



TIBCO® Order Management - Long Running

Administration

Version 5.0.1

April 2022

Document Updated: August 2023



Contents

Figures	7
About this Product	8
Deployment	9
Recommended Setup for a TIBCO Order Management - Long Running Development Environment	9
Microservices	10
Node Discovery	11
Deployment Topologies	13
Single Node Single Instance Topology	13
Single Node Multi-Instance Topology	14
Multi-Node Multi-Instance Topology	19
Predeployment Configuration	21
Components Deployment	21
Configurator Deployment	22
Automated Order Plan Development Deployment	22
Order Capture System Deployment	22
Order Management Server Deployment	22
Configuring SSL for TIBCO Order Management - Long Running	23
Connecting TIBCO Order Management - Long Running to TIBCO® EMS Server with SSL Enabled	23
Jeopardy Deployment	24
Order Management Server UI Deployment	24
Demo Subscriber Directory Toggling	25
Colocation	25
Configuration	26
Queue Management	26
Data Models	26
Model Loading Process	27
Online Model Loading	27
Offline Model Loading	28
Reading Offline XML Files	29
Model Processing	30
Enabling and Disabling Model Cache Persistence	32
Enabling and Disabling Posting Models on Enterprise Message Service topics	32
Jeopardy Management System	32
Jeopardy Management System Configuration	33
Order Management Server Components	34
Order Management Server Configuration	35

Messaging Configuration	35
User Interface Configuration	36
Plan Preview Integration with Order Management Server UI	38
Implementing OMSUIClient.jar	38
URL to Access Order Management Server UI Component	38
Target Parameters for Order Management Server UI	39
Additional Parameters for Order Management Server UI	39
Router Configuration	39
Routing Orders to Other Engines	42
Configuring Router Context	42
Multi-Tenancy Configuration	44
Managing Application Security	44
Managing Users and Roles	47
Creating User	49
Reading User Details	49
Deleting User	49
Resetting Password	49
Load Balancing	50
Order Management Server Web Service	50
Order Management Server Web Service Authentication	52
Collecting Order Summary Data	52
Audit Trail	53
Custom Audit Trail	53
Enabling Manual Order Plan Development Support	54
Manual Order Plan Development Orders Identification	54
Additional Manual Order Plan Development Configuration	55
Server Side Validation of Manual Order Plan Development	55
Enabling Internal Error Handler Support	56
State Machine Pagination	57
State Machine Configuration	57
State Machine Context Checkpoint	57
Order Capture System Configuration	57
Order Capture System External Component Configuration	57
Order Service Configuration	58
Subscriber Inventory Web Service Configuration	58
Eligibility and Pricing Web Service Configuration	59
Order Plan Preview Configuration	61
Pooled Database Source Configuration	62
PostgreSQL Database Configuration	65

Creating PostgreSQL Tables	66
Multi-Tenancy Configuration	66
Security Configuration	67
Catalog Directory Configuration	67
Access Points Common Configuration	68
Log File Configuration	68
Logging	68
How Logging Works	68
Contents of the Log Message	69
Controlling Log Levels	70
Configuring Logging for Java Components	70
Administration Tasks	72
Docker	72
Building a Docker Image Without an Internet Connection	72
Copying Files to Docker Context	73
Building Docker Images	73
ocs-util Docker Image	74
Preparing Docker Volumes	74
Setting Up the .env File	74
Configuring for Order Management Server Docker Containers	76
Running the Docker Containers	77
Enabling the RMI Port for the Order Management Server Container	78
Running the Order Management Server Container on a Predefined Port	78
Scaling the Docker Container	79
Extend Docker-Compose Files	79
Adding Environment Variables	79
Modifying a Container Time-Zone	80
Reading Container Logs	81
Troubleshooting Error from Building Docker Images	81
Order Sequencing	82
Enabling or Disabling Order Sequencing	83
Database Partitioning	83
Installing a Fresh Copy of a Database	83
Adding Future Partitions to an Existing Database	84
Truncating Old Partitions	84
Dropping Old Partitions	84
Database Partitioning with an Oracle Database	85
Database Partitioning with a Postgres Database	86
Migrating From a Non-Partitioned AUDIT_TRAIL Table	86

Product Model Purge	87
Purge Model Server Cache Impact	87
Bulk Order Actions	88
Bulk Order Actions	88
WSDL Location	88
Error Codes	88
Invoke Bulk Order Operation	89
Tracking the Request Status	89
Logging	89
Schema	90
Bulk Order Schema	90
Bulk Orders Operation Request Schema	90
Bulk Orders Operation Response Schema	90
Sample Request	90
Sample Response	91
Resource Failure Handling	91
Resource Failure Scenarios	91
Detection of Resource Failure	92
Action on Resource Failure	92
Action on Resource Recovery	93
Configuration of Resource Failure Handling	93
Advisory Messages and Impact on Fault Tolerance Processing	95
Event Processing on Orchestrator when it is Not Started	96
Multitenancy	96
Adding a New Tenant	96
Removing a Tenant	97
Manage Tenant Script Options	97
Cloning a Pluggable Database (PDB)	98
Creating a Source Pluggable Database	98
Preparing the Cloning Script	99
Order Capture System User Interface Tasks	99
Configuring the System	100
User Management	100
Entering Users	100
Editing Users	100
Deleting Users	101
Demo Subscriber Directory	101
Accessing the List of Subscribers	102
Entering Subscribers	102

- Editing Subscribers 102
- Deleting Subscribers 103
- Accessing the List of Stores103
 - Entering Stores 103
 - Editing Stores 104
 - Deleting Stores 104
- Reloading the Catalog104
- Back Office 104
- Search Syntax 105
- Managing Health Check Endpoint 105
- Tuning Performance 106**
 - Order Management Server Performance Tuning 106
 - Status Listener Queues 107
 - Changing Transient Data Store Operation Messages to Non-Persistent 109
- TIBCO Order Management - Long Running Disaster Recovery 111**
 - TIBCO Order Management - Long Running Topology for Disaster Recovery111
 - Storage and Volumes 111
 - Network 112
 - Database113
 - Messaging Server 113
- Schema References 115**
 - Plan Item 115
 - Result Status 117
 - Message 118
 - Order Request118
 - Order Request Header119
 - Order Request Line 122
 - Process Component Model 126
- TIBCO Documentation and Support Services 128**
- Legal and Third-Party Notices 129**

Figures

Single Node Single Instance Topology	14
Creating a New Cluster Member	15
Clone Member	16
JMX RMI Port	16
User Interface	17
Multi-node Multi-instance Topology	20
Offline Catalog Configuration	29
Poller Process	29
Jeopardy Management System Component Architecture	33
Time Window Value/Time Window Unit	34
Messaging Configuration	35
User Interface Configuration	36
Router Configuration	40
Router Types and Properties	40
Router	41
Order Management Server Application Security	45
Lightweight Directory Access Protocol Authentication Properties	46
Order Management Server Web Service	51
Central Logging	69
Bulk order action operation added to the Order Management Server order service	90
Request schema for bulk operation	90
Response schema for bulk operation	90
Listener Queues	106
AFI, Transient Data Store, AFS Configuration	107
Listener Queues	108
Plan Item	115
Result Status	117
Message	118
Order Request	118
Order Request Header	119
Invoice Address	120
Delivery Address	120
Order Request Line	122
Order Line Characteristics	123
Process Component Model	126

About this Product

TIBCO Order Management is an elastic, catalog-driven order management system for digital service providers. It accepts orders from any customer engagement system and orchestrates the tasks required for fulfilling the orders.

TIBCO Order Management is the next generation of TIBCO® Fulfillment Order Management and partially replaces the old product. To better align TIBCO Fulfillment Order Management with market demand, the product's capabilities have been reorganized into two new products: TIBCO® Order Management and TIBCO® Offer and Price Engine.

TIBCO Order Management is further divided into variant products:

- **TIBCO® Order Management - Low Latency:** Use this new product for scalable processing of low-latency orders
- **TIBCO Order Management - Long Running:** This product continues to support processing of long-running orders

Deployment

This section provides details about application deployment best practices and options.

Recommended Setup for a TIBCO Order Management - Long Running Development Environment

The following details are the recommended setup for a TIBCO Order Management - Long Running Development environment:

Component	Type	Instances	Primary Server	Memory
Order Management Server (Automated Order Plan Development + Orchestrator + Jeopardy Management System + UI)	Java	1	M1	4 GB
Oracle	Database	1	M2	8 GB
TIBCO Enterprise Message Service™	Messaging	1	M1	Default
Process Component (TIBCO BusinessWorks, TIBCO BusinessWorks Container Edition, or TIBCO BusinessEvent)	Java	1	M1	2 GB

Hardware

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

Disk Space

To install TIBCO Order Management - Long Running 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 might 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:

```
install_package_name.bin -is:tempdir /new_tmp where /new_tmp has sufficient free disk space.
```

Jeopardy Deployment Configuration

In the collocated mode, jeopardy deployment is a single instance or multiple instance cluster setup. Deploy it with the following configuration:



Fields	Colocated configurations
Profiles.properties	<code>com.tibco.fom.jeoms.deployMode = JEOMS_colocated</code>
ConfigValues_JeOMS.xml	<code>Operational Datastore = cache</code>

- Local and TIBCO ActiveSpaces® data stores are not supported in this version.
- If Jeopardy is enabled, which means that the Jeopardy Management System deploy mode is set to `JEOMS_colocated`, every TIBCO Order Management - Long Running cluster instance has to run Jeopardy Management System in the colocated mode. This means that if some server instances are set to `JEOMS_disabled` and some server instances are set to `JEOMS_colocated`, such configurations are not supported.

Microservices

Each TIBCO Order Management - Long Running component, or *microservices* with the new architecture, has its embedded Tomcat container. The `roles` folder available in the `$OM_HOME` directory houses all the microservices.

The following table lists the microservices for TIBCO Order Management - Long Running:

Microservice	Description	Default Port
configurator	Microservice for Configurator.	9090
omsserver  By default, Order Management Server runs Automated Order Plan Development in colocated mode.	Microservice for the Order Management Server.	9091
omsui	Microservice for the Order Management Server UI.	9092
aopd  By default, this service runs Automated Order Plan Development in standalone mode.	Microservice for Automated Order Plan Development.	9093
ocs	Microservice for Order Capture System.	9095

Each microservice under the `$OM_HOME/roles/<service name>/standalone` directory has the following directory structure:

- `bin`
This directory contains shell and power shell scripts to start and stop the service. It also contains a `copyLib` script, which is a utility script that can be used to copy hibernate, JDBC, JMS, and other essential dependencies.
- `config`
This directory contains the service's set of configuration files. Initially, each service has the following files:

- `application.properties`
- `configDBrepo.properties`
- `profiles.properties` (except for Configurator and Order Capture System)

When the service starts, it downloads its required config files from the database.

- `lib`

This directory holds all external and internal dependency jar files.

- `logs`

This directory is created when the service starts and contains all the logs for that service.

- `services`

This directory holds the service jar file, which is launched by the start script

Apart from these directories, some services like Order Capture System and Order Management Server UI, have extra directories for the UI.

Node Discovery

Automatic node discovery lets TIBCO Order Management - Long Running services to start without specifying the `NODE_ID` system property. This enables the services to scale without any manual intervention and provides the integration expected with a container technology like Docker.

In case of static assignment, for each microservice, the system properties that have to be set are `NODE_ID` and `DOMAIN_ID`.

The following services use the Node Discovery mechanism:

- Order Management Server
- Automated Order Plan Development
- Order Management Server UI
- Order Capture System

Static Assignment

This option lets a particular application be statically bound to a fixed node name. This is possible by providing a system property corresponding to the application at the JVM startup and marking the particular node as static in the domain members table.

The following are the system properties you can use to bind a node to a node name:

- Order Management Server (`NODE_ID`)
- Automated Order Plan Development (`NODE_ID`)
- Order Management Server UI (`NODE_ID`)
- Order Capture System (`NODE_ID`)

Each service for this option has to belong to a domain defined in the domain table.

The following are the system properties for specifying domain names corresponding to the services:

Service	System Property	Default Value
Order Management Server	<code>DOMAIN_ID</code>	ORCH-DOMAIN

Service	System Property	Default Value
Automated Order Plan Development	DOMAIN_ID	DYNAMIC
Order Management Server UI	DOMAIN_ID	DYNAMIC
Order Capture System	DOMAIN_ID	OCS-DOMAIN

The entry for all members for a particular service must be present in the domain members table, and the domain being used might be present in the domain table. The members of the domain members table must have `is_static` set to 1 for members reserved for static allocation.

Dynamic Assignment

This option lets a particular service dynamically register itself as a member in the cluster. The node name might not be provided for this option to work.

Each service for this option has to belong to a domain defined in the domain table.

The following are the system properties for specifying the domain names corresponding to the services:

Service	System Property	Default Value
Order Management Server	DOMAIN_ID	ORCH-DOMAIN
Automated Order Plan Development	DOMAIN_ID	DYNAMIC
Order Management Server UI	DOMAIN_ID	DYNAMIC
Order Capture System	DOMAIN_ID	OCS-DOMAIN

The entry for all members for a particular service might be present in the domain members table and the domain being used might be present in the domain table. The members of the domain members table might have `is_static` set to 0 for members reserved for dynamic allocation.

Dynamic Domain

This option lets services start without a defined domain value. Services covered under this option are Automated Order Plan Development, and Order Management Server UI. The following services do not have to be defined through the system properties:

- Automated Order Plan Development (NODE_ID)
- Order Management Server UI (NODE_ID)

For the cases where the node ID is not provided, a random node is allocated to the member of the application.

The default domain for these services is set to DYNAMIC; therefore, by default, they start without entries present in the domain members table. However, it is possible to start these services in the non-dynamic modes by providing a domain name as system properties:

Service	System Property	Default Value
Automated Order Plan Development	DOMAIN_ID	DYNAMIC
Order Management Server UI	DOMAIN_ID	DYNAMIC

Deployment Topologies

TIBCO Order Management - Long Running server-side components (that is omsServer (including Orchestrator), Automated Order Plan Development, and Jeopardy Management System) and the client side user interface components (that is Configurator and Order Management Server UI with dashboard) must be deployed for leveraging their functionalities. Anyone of the three topologies explained below can be followed for the deployment.

Single Node Single Instance Topology

In Single Node Single Instance topology, a single instance of one or more than one service run on the default ports.

This is the default topology that is supported immediately after the installation and initial configurations of TIBCO Order Management - Long Running.

Single Node Multi-Instance (Vertical Scaling) Topology

In the Single Node Multi-Instance topology, the service instance under roles can be copied to create and start multiple instances. You might opt to create copies of all or a combination of the services and start them after changing their default port values. There might be exactly one instance of the Configurator service running in any kind of topology.

This topology is also referred to as *Vertical Scaling*. Through it, the components leverage the processing power efficiency of the machine. One microservice 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 microservice instances, multiple JVM processes are started and provide multiple thread pools. The server components utilize the maximum processing power of the machine.

Multi-Node Multi-Instance (Horizontal Scaling) Topology

In Multi-Node Multi-instance topology, multiple microservice instances are started on multiple nodes to have multiple instances running in the cluster. You can deploy and start the application services on multiple nodes. The Configurator service must be installed and run only on one node. The other services can have multiple instances on any number of nodes.

This topology is also referred to as *Horizontal Scaling*. Through it, the components leverage the processing power efficiency of multiple machines. By running additional microservice instances on multiple machines, multiple JVM processes are started and provide multiple thread pools. The server components use the maximum processing power of all the machines.

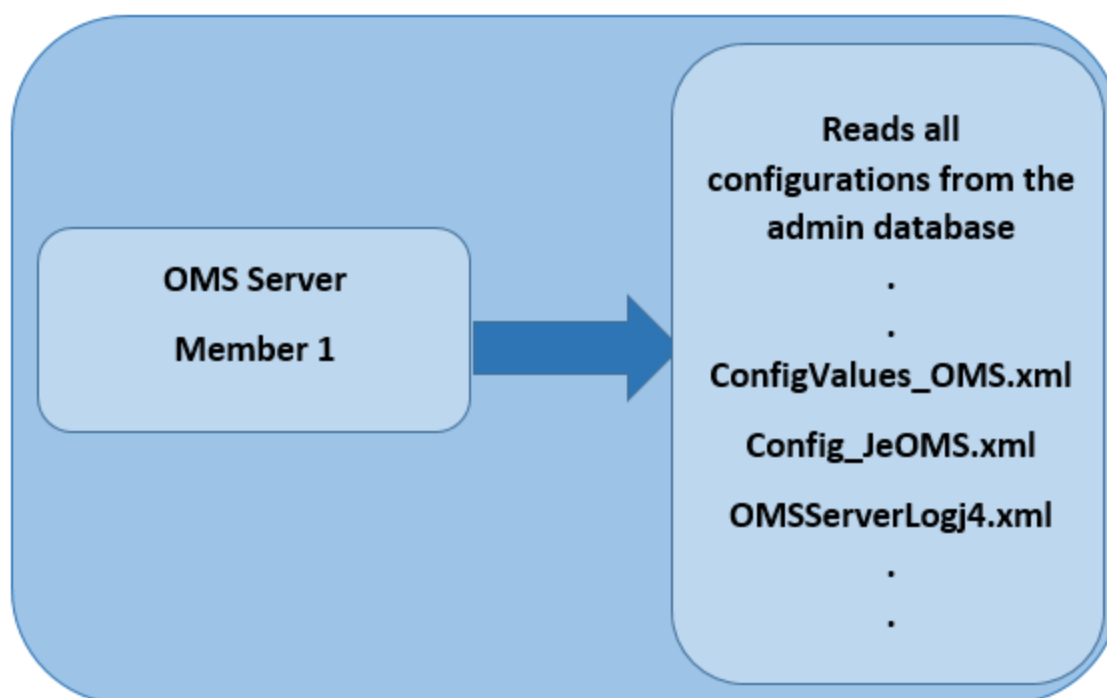
Single Node Single Instance Topology

This topology is supported by default, once the TIBCO Order Management - Long Running installation and configurations are done. The TIBCO Order Management - Long Running installer updates the values of some environment variables. For example, the \$OM_CONFIG_HOME variable is updated according to the installation directory path. The \$OM_CONFIG_HOME variable points to \$OM_HOME/roles/standalone/configurator/config directory containing configuration files used by the TIBCO Order Management - Long Running components.

For example, the omsServer component refers to some of the configuration files as mentioned below:

- `ConfigValues_OMS.xml` for all configuration properties such as JMS connection parameters, database connections, and other functional configuration properties.
- `OMSServerLog4j.xml` for Logging configurations.

Single Node Single Instance Topology



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 assigned `NODE_ID`. The default value of `NODE_ID` is `Member1`. The respective configuration files also contain the default member level configuration section (`<Server name="Member1">`).

Most of the required configurations such as JMS connection and database connections can be done through TIBCO Order Management - Long Running Configurator after the installation. Optionally the configurations can also be changed using the configurator application.

Single Node Multi-Instance Topology

The Single Node Multi-instance or Vertical Scaling deployment topology can be achieved by running multiple instances of the microservices on a single machine.

The Orchestrator component is an integral part of omsServer. By running multiple instances of the microservice, Orchestrator instances form a cluster. The name of the cluster is passed as `DOMAIN_ID` property with the default value as 'ORCH-DOMAIN' and must be the same in all the scripts.

The number of consumer counts (listeners) on JMS queues and topics used by different TIBCO Order Management - Long Running components are equal to the product of the count configured and the total number of instances running.

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

You can use the following steps for deployment topology.

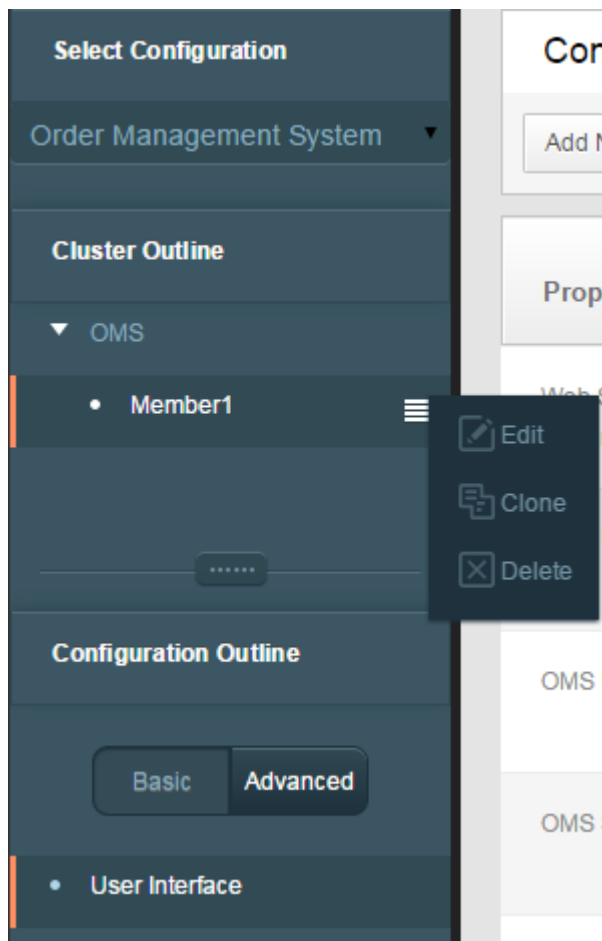
Prerequisites

1. TIBCO Order Management - Long Running is installed on top of all the underlying required products on the designated server node.
2. Correct configurations for the default topology, Single Node Single Instance, are in place.
3. The default microservice instance under \$OM_HOME/roles is up and running with all the server and client side components deployed successfully.

Creating New Cluster Members

1. Access TIBCO Order Management - Long Running configurator GUI application in a supported browser through the HTTP interface of the default microservice instance using the URL `http://<host>:<port>/#/login`.
2. Select **Order Management System** configuration.
3. Navigate to the existing member **Member1** in the cluster. Right-click **Member1** to pop up the menu. Select **Clone** as shown in the following figure:

Creating a New Cluster Member



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

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

Clone Member

Clone Member

Clone the Member

Name:

Member2

Save

Cancel

- Change the JMX RMI port for Member2 using the following configuration:

JMX RMI Port

Select Configuration

Order Management System

Cluster Outline

OMS

- Member1
- Member2

Configuration Outline

Configuration and Setup For Member2 - FOM JMX

Add New Property

Clone

Delete

Property	Value	Description
JMX RMI Port	10099	JMX RMI Port

- Save the configuration changes. The newly created Member2 is saved in the `$OM_HOME/roles/configurator/standalone/config/ConfigValues_OMS.xml` file.
- The user interface is at a cluster level, which means it is available for all the configured members; therefore, select **OMS** under Cluster Outline and select **User Interface** to configure the members and change the values for the following properties. Provide unique values as displayed:

User Interface

Configuration and Setup For OMS - OMS User Interface

Buttons: Add New Property, Clone, Delete

Property	Value	Description
FP Server node name	knode	Node name of the FP server
Web Service HTTP Transport Channel Type	http	
OMS UI HTTPS Port Number	8443	
OMS Server Host Name	cimlinxqa2.na.tibco.com	Host address of the OMS server
Owner for FP	FP	Owner for FP
OMS Server Port Number	9091	Port number of the OMS Server
OMS UI HTTP Port Number	9092	
FP Server Host Name	localhost	Host address of the FP server
FP Server Port Number	8080	Port number of the FP Server

8. Save the port configuration changes.
9. Perform steps [three](#) to [six](#) for creating additional members Member3, Member4, Member5, and so on in the cluster. This might be done according to the requirement.
10. Save the configuration changes. The *Configuration Saved* popup opens. At this step, the number of copies of the complete <Server name="Member1"> element are created as <Server name="Member2">, <Server name="Member3">.

Note that the deployment of multiple Order Management Server 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 Order Management Server. For this, no specific configuration is required. The only requirement is that the load balancer must have support for the sticky session. Sticky sessions mean 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 web applications using session cookies. It does not rely on the IP address. Typically, HLB inserts a cookie that the load balancer creates and manages automatically to remember, which back-end server a given HTTP connection might 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 NetScaler, Barracuda, and jetNexus.

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 is automatically multiplied by the total number of deployed instances.

Adding Cluster Members to a Database

1. An entry for the default member Member1 in the default cluster ORCH-DOMAIN is already present in the DOMAINMEMBERS table of the Order Management Server database.
2. To deploy and run additional members in the Orchestrator cluster, corresponding entries must be added. For this, refer to the following insert statement in the \$OM_HOME/db/oracle/oms/OMS_SeedData.sql script.

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

3. Replace the values for the memberid and description columns in the insert statement according to the names of the additional members created in previous steps. Prepare one insert statement for each new member to be added.
4. Run the insert statements prepared for all the new members on the Order Management Server database schema. Also, run the commit statement to commit the insert changes.
5. Run select * query on DOMAINMEMBERS table and verify the newly added members.



Always add only the required number of member entries to the DOMAINMEMBERS table. For example, if 10 instances are required to be run in the ORCH-DOMAIN cluster, add only the corresponding 10 entries in the 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', ...);
```

Creating Additional microservice Instances

1. Stop the running instance of the default microservice.
2. Create as many copies of the roles directory under the \$OM_HOME directory as the number of members configured previously to be run in the cluster.
3. Enter the admin database details in the configDBrepo.properties file, and change the port number accordingly. The port numbers in \$OM_HOME/roles/omsServer/standalone/config/application.properties must be changed for added members.
4. In case of static allocation of member IDs to nodes, set the system properties NODE_ID and DOMAIN_ID. If a dynamic allocation is required, it is not required to set these variables.

The cluster deployment is primarily done to scale the server-side TIBCO Order Management - Long Running components that are omsServer (Orchestrator and Automated Order Plan Development included) and jeoms.

The server components are scaled through Enterprise Message Service as more number of listeners and processors are activated on inbound destinations by deploying them in multiple microservices. This also creates additional thread pools to increase processing capabilities.

Sanity Test

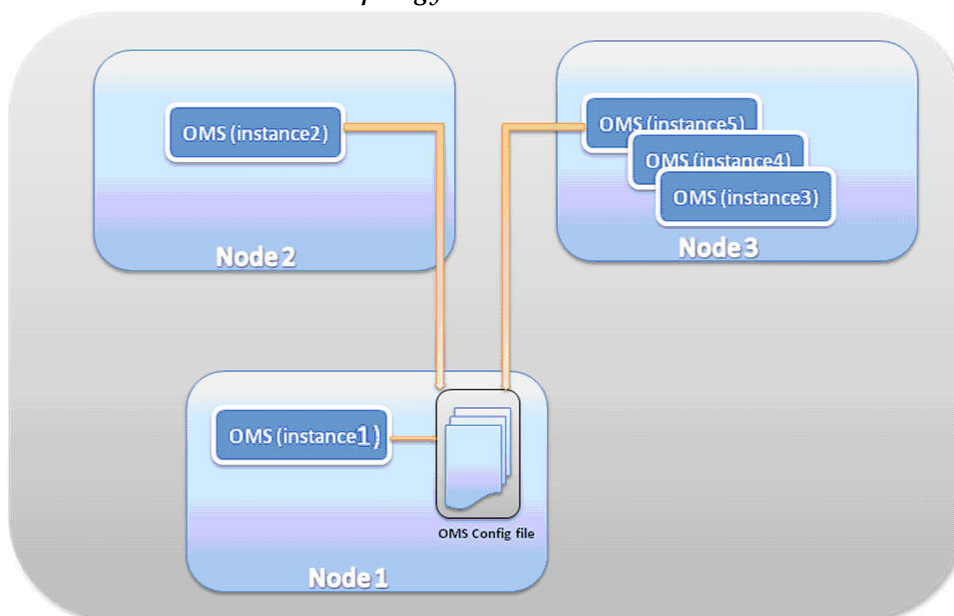
1. Start all microservice instances to start the deployed TIBCO Order Management - Long Running components.
2. Monitor the logs of each microservice to make sure that the server and all the deployed components have started correctly without any errors.
3. Start the Orchestrator member as a Cluster Manager for the ORCH-DOMAIN cluster. Start the Orchestrator member in the same cluster. Verify this according to the sample log statements shown below in the log file.

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

The multi-node multi-instance, or horizontal scaling deployment topology, can be achieved by running multiple instances of microservices on multiple nodes (machines). This is mainly done to increase the overall processing capacity of the TIBCO Order Management - Long Running server-side components namely omsServer and jeoms. The default microservice instance is duplicated on the main and other nodes to run multiple instances.

Once a node is configured, other nodes download the configuration from the database. However, this feature is only available if the database configuration is selected; it is selected by default. If disabled, the configuration has to be manually copied.

Multi-node Multi-instance Topology



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

The following steps help to use this deployment topology.

Prerequisites

- The setup for Single-node Multi-instance deployment topology is already in place and working on the main node on which TIBCO Order Management - Long Running is installed.
- All the required underlying software such as Java and the libraries such as Oracle JDBC driver, Enterprise Message Service libraries, which are being referred to in the microservices are already available on the additional nodes. All the corresponding environment variables are set and exported.

Creating New Cluster Members

1. Access TIBCO Order Management - Long Running Configurator GUI application in a supported browser through the HTTP interface of the default Configurator microservice instance on the main node using the URL `http://<HOST>:<PORT>`.
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 change the member name. As the new member needs to be run on a different node, the ports can be kept as-is.

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 runs on different node, the ports, although same, does not generate conflicts with Member1. Follow the same procedure for the remaining instances.
4. After the required number of additional members to be run on other nodes is created in the configuration files, other nodes download the configuration from the database if the database configuration is selected.
 - a. Make as many copies of the services (`$OM_HOME/roles/omsServer`, or `$OM_HOME/roles/oep`, or `$OM_HOME/roles/aopd`, and so on) needed to configure the number of instances at any location, mention the admin database details in the `configDBrepo.properties` file, and change the port number accordingly.

- b. Set the `$OM_CONFIG_HOME` environment variable to point to the directory path of the copied config directory `OM_HOME/roles/configurator/standalone/config`. This step is only required for the configurator service. The other services do not need this variable set.

Adding Cluster Members to the Database

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

Creating Additional microservice Instances

1. Create additional microservice instances on other nodes by copying the existing microservice directories from the main node.

The easiest way is to copy each directory and just change the member's suffix number in the directory name.

2. The port numbers in `$OM_HOME/roles/omsServer/standalone/config/application.properties` do not have to be changed for Member1. As it runs on a different node, the ports, although same, do not conflict with Member1. Follow the same procedure for the remaining instances.
3. In case of static allocation of member IDs to nodes, set the system properties `NODE_ID` and `DOMAIN_ID`. If a dynamic allocation is required, it is not required to set these variables.

Sanity Test

The sanity test for Multi-node Multi-instance topology is the same as explained in Single-node Multi-instance topology. Here the additional number of Orchestrator instances running on other nodes joins the ORCH-DOMAIN cluster. Anyone instance among all acts as the Cluster Manager and all others act as Workers.

Predeployment Configuration

Configure TIBCO Order Management - Long Running Configurator and other microservices before deploying them.

Edit the property file `$OM_HOME/roles/configurator/standalone/config/configDBrepo.properties` to configure the database details for the admin schema.

Configurator makes changes to these files and uploads them to the admin schema database. When a service starts up, it downloads all the required configuration files from the admin schema. Similar to Configurator, each service has a `configDBrepo.properties` file, where you must configure details of the admin schema. Configurator detects any changes done directly to the config files under the configurator service file system and uploads them to the database. You must restart the service to download the changes.

If you prefer to configure each service manually and not depend on the database-based approach, you can disable the feature for each service in `configDBrepo.properties` and manually copy the required files from Configurator to each service and edit the files. The property to disable this is `configurator.db.enabled`.

Components Deployment

This section provides details on deploying all the TIBCO Order Management - Long Running components.

- [Configurator Deployment](#)
- [Automated Order Plan Development Deployment](#)
- [Order Capture System Deployment](#)
- [Order Management Server Deployment](#)
- [Configuring SSL for TIBCO Order Management - Long Running](#)
- [Jeopardy Deployment](#)

- [Order Management Server UI Deployment](#)

Configurator Deployment

Deploy TIBCO Order Management - Long Running Configurator to configure the rest of the microservices.

After you have made the appropriate configurations for the [Predeployment Configuration](#), you can deploy Configurator.

To start and stop Configurator, use the following scripts:

- To start: `$OM_HOME/roles/configurator/standalone/bin/start.sh` (for Unix) or `start.ps1` (for Windows)
- To stop: `$OM_HOME/roles/configurator/standalone/bin/stop.sh` (for Unix)

Automated Order Plan Development Deployment

The Automated Order Plan Development component can be deployed either collocated with Order Management Server (that is default mode) or it can be deployed in standalone mode, in its application server.



The property file `$OM_HOME/roles/aopd/standalone/config/configDBRepo.properties` must be configured for this service.

Automated Order Plan Development deployment can be permanently set in the `$OM_HOME/roles/aopd/standalone/config/profiles.properties` and `$OM_HOME/roles/omsServer/standalone/config/profiles.properties` file:

- `com.tibco.fom.aopd.deployMode=AOPD_standalone`

The Automated Order Plan Development component can also be disabled to use the custom implemented Automated Order Plan Development. To do this, set the following property in the `$OM_HOME/roles/aopd/standalone/config/profiles.properties` file:

- `com.tibco.fom.aopd.deployMode=AOPD_custom`

To start and stop Automated Order Plan Development, use the following scripts:

- To start: `$OM_HOME/roles/aopd/standalone/bin/start.sh` (for Unix) or `start.ps1` (for Windows)
- To stop: `$OM_HOME/roles/aopd/standalone/bin/stop.sh` (for Unix)

Order Capture System Deployment

Order Capture System (OCS) can only be deployed in standalone mode.



The property file `$OM_HOME/roles/ocs/standalone/config/configDBRepo.properties` must be configured for this service.

To start and stop the Order Capture System, use the following scripts:

- To start: `$OM_HOME/roles/ocs/standalone/bin/start.sh` (for Unix) or `start.ps1` (for Windows)
- To stop: `$OM_HOME/roles/ocs/standalone/bin/stop.sh` (for Unix)

Order Management Server Deployment

The functionality and interfaces of the default Java based Orchestrator component are enabled only through the collocated deployment with omsServer, which is the default mode. Orchestrator cannot be deployed in a standalone mode like Automated Order Plan Development because 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 the `$OM_HOME/roles/omsServer/standalone/config/profiles.properties` file.

Configuring SSL for TIBCO Order Management - Long Running

This section describes how to configure SSL for Order Management Server UI and Order Management Server, the web-based application components of TIBCO Order Management - Long Running.

Configure SSL for Order Management Server UI and Order Management Server by using the following steps:

1. Edit the application.properties files in the following locations:

- `<OM_HOME>/roles/configurator/standalone/config/application.properties`
- `<OM_HOME>/roles/omsServer/standalone/config/application.properties`
- `<OM_HOME>/roles/omsui/standalone/config/application.properties`
- `<OM_HOME>/roles/aopd/standalone/config/application.properties`
- `<OM_HOME>/roles/ocs/standalone/config/application.properties`

Add the following parameters to each application.properties file:

- `server.ssl.key-alias=<key-alias>`
- `server.ssl.key-password=<key-password>`
- `server.ssl.key-store=classpath:<ssl-key-store-fileName>`
- `trust-store=classpath:<ssl-key-store-fileName>`
- `trust-store-password=<key-password>`

2. Keep the keystore files in each directory or as a classpath resource.

- `<OM_HOME>/roles/configurator/standalone/config`
- `<OM_HOME>/roles/omsServer/standalone/config`
- `<OM_HOME>/roles/omsui/standalone/config`
- `<OM_HOME>/roles/aopd/standalone/config`
- `<OM_HOME>/roles/ocs/standalone/config`

3. Edit and save the files and then start the Configurator.

- a. In a browser, open the following URL: `https://host:configuration_port`.
- b. Log in to Configurator
- c. In Order Management Server **basic configuration** > **OMS User Interface** change the **Web Service HTTP Transport Channel Type** property to `https` and **HTTP Port number** to `omsServer port number`.
- d. In Order Management Server **Advance** > **Web Service Configuration**, change **HTTP Channel type** to `https` and **HTTP Port number** to `omsServer port number`

4. Start (or restart) Order Management Server and Order Management Server UI.

Connecting TIBCO Order Management - Long Running to TIBCO® EMS Server with SSL Enabled

Procedure

1. In Configurator or the XML files, set the following properties as specified:

- In Configurator:
 - `com.tibco.tibjms.naming.security_protocol = ssl`
 - `com.tibco.tibjms.naming.ssl_enable_verify_host = false`
 - `TopicConnectionFactory = SSLTopicConnectionFactory`
 - `QueueConnectionFactory = SSLQueueConnectionFactory`
 - `GenericConnectionFactory = SSLConnectionFactory`
- In the xml files:
 - In `<OM_HOME>/roles/configurator/standalone/config/ConfigValues_OMS.xml`
 - `com.tibco.tibjms.naming.security_protocol = ssl`
 - `com.tibco.tibjms.naming.ssl_enable_verify_host = false`
 - `TopicConnectionFactory = SSLTopicConnectionFactory`
 - `QueueConnectionFactory = SSLQueueConnectionFactory`
 - `GenericConnectionFactory = SSLConnectionFactory`
 - In `<OM_HOME>/roles/configurator/standalone/config/OMSServerLog4j.xml`
 - `TopicConnectionFactory = SSLTopicConnectionFactory`
 - In `<OM_HOME>/roles/configurator/standalone/config/AOPDLog4j.xml`
 - `TopicConnectionFactory = SSLTopicConnectionFactory`
 - In `<OM_HOME>/roles/configurator/standalone/config/ConfigValues_AOPD.xml`
 - `GenericConnectionFactory = SSLConnectionFactory`

2. Start (restart) the server.

Jeopardy Deployment

The Jeopardy component can be deployed only in collocated mode with Order Management Server in its application server.

The deployment mode for Jeopardy can be permanently set in the `$OM_HOME/roles/OMS/standalone/config/profiles.properties` file:

- `com.tibco.fom.jeoms.deployMode=JEOMS_collocated`

The Jeopardy component can also be disabled if the Jeopardy Management features are not required. To do this, set the following property in the `$OM_HOME/roles/omsServer/standalone/config/profiles.properties` and the `$OM_HOME/roles/omsui/standalone/config/profiles.properties` files:

- `com.tibco.fom.jeoms.deployMode=JEOMS_disabled`

To start and stop Jeopardy, use the following scripts:

- To start: `$OM_HOME/roles/oms/standalone/bin/start.sh` (for Unix) or `start.ps1` (for Windows)
- To stop: `$OM_HOME/roles/oms/standalone/bin/stop.sh` (for Unix)

Order Management Server UI Deployment

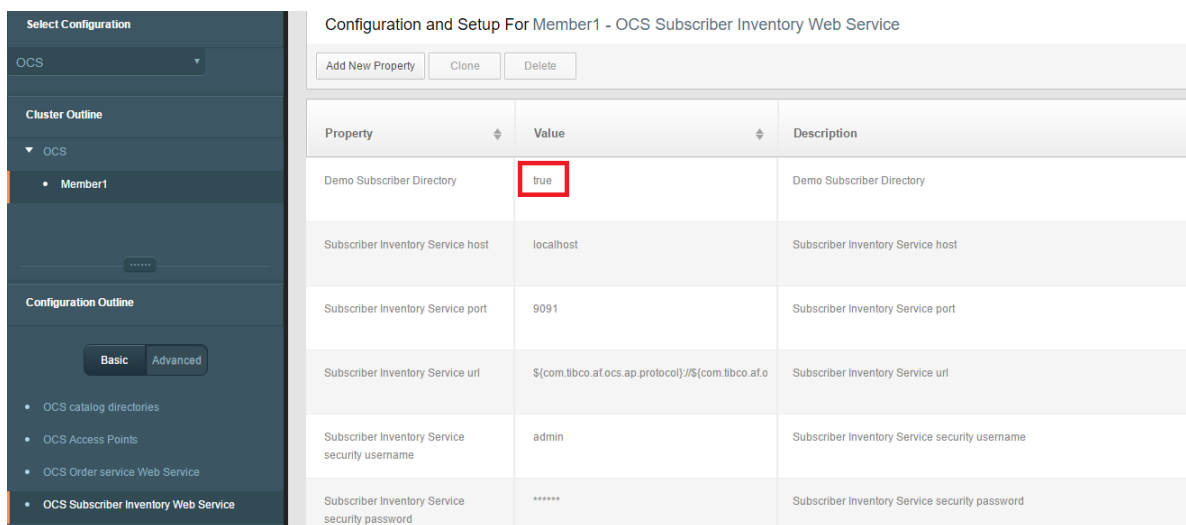
To start and stop Order Management Server UI, use the following scripts:

- To start: `$OM_HOME/roles/omsui/standalone/bin/start.sh` (for Unix) or `start.ps1` (for Windows)
- To stop: `$OM_HOME/roles/omsui/standalone/bin/stop.sh` (for Unix)

Demo Subscriber Directory Toggling

Enable or disable Demo Subscriber Directory in TIBCO Master Data Management Configurator to demonstrate subscriber and store management in Order Capture System.

From the Order Capture System Subscriber Inventory Web Service Configuration Outline in TIBCO Master Data Management Configurator, the first line of configuration values for Demo Subscriber must be switched to `true` to enable and `false` to disable. If Demo Subscriber Directory is disabled, Order Capture System accesses the Subscriber Inventory Web Service configured in Order Capture System Subscriber Inventory Web Service Configuration Outline.



The screenshot shows the 'Configuration and Setup For Member1 - OCS Subscriber Inventory Web Service' window. On the left, the 'Cluster Outline' shows 'OCS' expanded with 'Member1' selected. Below it, the 'Configuration Outline' shows 'Basic' and 'Advanced' tabs, with 'OCS Subscriber Inventory Web Service' selected. The main area displays a table of configuration properties:

Property	Value	Description
Demo Subscriber Directory	true	Demo Subscriber Directory
Subscriber Inventory Service host	localhost	Subscriber Inventory Service host
Subscriber Inventory Service port	9091	Subscriber Inventory Service port
Subscriber Inventory Service url	\$(com.tibco.af.ocs.ap.protocol)/\$(com.tibco.af.o	Subscriber Inventory Service url
Subscriber Inventory Service security username	admin	Subscriber Inventory Service security username
Subscriber Inventory Service security password	*****	Subscriber Inventory Service security password

Colocation

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

See the [Configuration](#) section for more details on how to configure the different deployments.

Configuration

This section covers all the configuration details for TIBCO Order Management - Long Running.

Queue Management

TIBCO Order Management - Long Running communicates with the external systems through the JMS messaging capability provided by TIBCO Enterprise Message Service. It has many inbound queues to receive the messages from external systems and also for inter-component communication in some cases. The number of listeners on these queues can be configured using Configurator. By default, the listener count on each queue is set to a minimal value. The queue configurations are available under different categories distributed component wise.

Data Models

TIBCO Order Management - Long Running requires a variety of data models (catalogs) for its different functionalities.

TIBCO Order Management - Long Running uses the following data models:

Data Models	Description
Product Model	It is used by the Automated Order Plan Development component for generating the execution plan for the newly submitted orders and the amendments. It is also used by the Order Capture System, which is an optional component, in TIBCO Order Management - Long Running.
Action Model	It is optionally used by the Automated Order Plan Development component when generating the execution plans specifically for the ProductDependsOn feature.
Plan Fragment Model	It is used by the Orchestrator component for executing the plan for a particular order. It is also used by the optional component – Jeopardy Management System for supporting the jeopardy management functionalities.



The optional Order Capture System component also uses the Segment Model along with Product Model. However, since it is not used by any other core components such as Automated Order Plan Development or Orchestrator, its specific details are not covered here. For details, refer to the Order Capture System documentation in *TIBCO Order Management - Long Running User's Guide*.

The following table summarizes the models required by components in TIBCO Order Management - Long Running:

Components	Data Models			
	Product	Action	Plan Fragment	Segment
Automated Order Plan Development	Required	Optional	Not Required	Not Required
Orchestrator	Not Required	Not Required	Required	Not Required

Jeopardy Management System	Not Required	Not Required	Required	Not Required
Order Capture System	Required	Not Required	Not Required	Required

Model Loading Process

The models mentioned in [Data Models](#) must be loaded into TIBCO Order Management - Long Running database so that they can be used by different components. These data models are modeled as catalogs using repositories and relationships readily available in TIBCO Fulfillment Subscriber Inventory. After modeling the catalogs in TIBCO Fulfillment Subscriber Inventory, they can be made available to TIBCO Order Management - Long Running and swagger REST API catalog service, either by publishing them on JMS topics, or by writing them into offline XML files, which can be used by TIBCO Order Management - Long Running at a later point in time.



When model loading, Orchestrator does not process any incoming order events for orchestration when the existing flag `com.tibco.fom.oms.modelLoadingMaxIdle` is set to `true`. When the flag is set to `false`, Orchestrator processes the order when the model loading is in progress.

Use the following ways to load the models into TIBCO Order Management - Long Running:

- [Online Model Loading](#)
- [Offline Model Loading](#)

After the models are received in TIBCO Order Management - Long Running, one member is assigned a task of loading models. This member is identified by the designated model loading member instance property `com.tibco.af.oms.model.loading.member` in the configuration file. This member is responsible for completing the model loading process and coordination with the rest of the members in the cluster.

The following steps are performed on receiving the model:

1. A designated model loading member starts sending an advisory message on the topic `tibco.aff.orchestrator.cluster.advisory.heartbeat` indicating that the model loading has started. All the members listen to this message and set the `modelLoadingStarted` flag to `true`. This member keeps sending an advisory message until the loading is complete.
2. If no advisory message is received by other members in the cluster, the members set the `modelLoadingStarted` flag to `false` after the threshold idle time.
3. If there are any incoming requests to Orchestrator, it checks the `modelLoadingStarted` flag. If this flag is `true`, Orchestrator keeps waiting until the flag is set to `false` to proceed further. Once this flag becomes `false`, the Orchestrator request is processed.
4. Catalog Web Service Model Loading is used for Product and Planfragment Model loading.

Online Model Loading

Online model loading requires the invoking of the catalog publish workflow in TIBCO Product and Service Catalog using the exposed SOAP service.

You can invoke the catalog publish workflow in TIBCO Product and Service Catalog directly by using the sample SOAP web service requests available in the `$OM_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.

Invoke the request against the running instance of TIBCO Product and Service Catalog on the URL, which typically looks like `http://<HOST>:<PORT>/em1/services/router/MasterCatalogRecordAction` where

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

Refer to the TIBCO Product and Service Catalog documentation for more details.

The sample request to invoke TIBCO Product and Service Catalog workflow for publishing product catalogs are available in `$OM_HOME/samples/Online_ProductModel_Publish_Request.xml`.

TIBCO Product and Service Catalog publishes the models on respective topics as mentioned in the following table:

Model (Catalog)	TIBCO Product and Service Catalog JMS Topic
Product	tibco.ac.productmodel.topic
Action	tibco.ac.actionmodel.topic
Plan Fragment	tibco.ac.planfragmentmodel.topic

To make these models available to TIBCO Order Management - Long Running, the following JMS bridges must be created between the TIBCO Product and Service Catalog topics and the corresponding TIBCO Order Management - Long Running queues as mentioned in the following table:

TIBCO Product and Service Catalog Source Topic	TIBCO Order Management - Long Running Target Queue
tibco.ac.productmodel.topic	tibco.aff.catalog.product.request
tibco.ac.actionmodel.topic	tibco.aff.catalog.action.request
tibco.ac.planfragmentmodel. topic	tibco.aff.catalog.planfragment.request



A bridge does not have to be created for the Segment Model because it is not required by the core components. Order Capture System loads the segment model differently. See Order Capture System details in *TIBCO Order Management - Long Running User's Guide* for more details.

This software has separate listeners on each queue mentioned in the earlier table. On receiving the JMS message, corresponding to a particular model, TIBCO Order Management - Long Running processes it using [Model Processing](#).

Offline Model Loading

Offline models are available as flat files containing the model data in the form of the schema compliant XML payload. These files must be generated first by invoking the catalog publish workflow in TIBCO Product and Service Catalog. It is a one time activity unless the modeling is changed in TIBCO Product and Service Catalog.

This feature enables TIBCO Product and Service Catalog to have no dependency on TIBCO Product and Service Catalog to be online all the time for the data models.

The sample offline XML files for all models are available in `$OM_HOME/samples/Models/offline` directory.

Offline catalog mode must be enabled for each catalog (model) separately using TIBCO Product and Service Catalog Configurator.

The following figure shows the offline catalog configuration for the PRODUCT catalog. The configurations for other catalogs can be done similarly.

Offline Catalog Configuration

The screenshot shows the TIBCO FOM Configurator interface. The left sidebar has a 'Select Configuration' section with 'Order Management System' selected. Below it is a 'Cluster Outline' section with 'OMS' selected. The main area is titled 'Configuration and Setup For OMS - Offline Catalog Configuration - Product Catalog Configuration'. It contains a table with the following properties:

Property	Value	Description
Use offline product	true	Use offline product
Product catalog poller and WS directory	/usr/tibco/product	Product catalog poller and WS directory
Product catalog import success poller and WS directory	/usr/tibco/product-success	Product catalog import success poller and WS directory
Product catalog import failure poller and WS directory	/usr/tibco/product-failure	Product catalog import failure poller and WS directory

From the screenshot you can see that three directories must be configured for loading the offline catalog:

1. Catalog poller and WS directory: The main directory, which contains the offline .xml files to be read by TIBCO Product and Service Catalog. These files can be read by TIBCO Product and Service Catalog in either of the following ways:
 - a. By a thread, which polls the directory periodically.
 - b. On demand by invoking the offline catalog SOAP web service externally.
2. Catalog import success poller and WS directory: The directory into which the offline .xml files might be moved if the file is successfully read and processed.
3. Catalog import failure poller and WS directory: The directory in which the offline .xml files might be moved if the file could not be read or processed successfully.

Reading Offline XML Files

TIBCO Order Management - Long Running can read the offline XML files in two ways: Poller process and SOAP web service.

1. Poller Process

A separate poller thread is started for loading each offline catalog. This thread keeps polling the main catalog directory to read the catalog files that are present in the directory. The poller polls the directory according to the polling interval configured in the property shown in the following figure:

Poller Process

The screenshot shows the TIBCO FOM Configurator interface. The left sidebar has a 'Select Configuration' section with 'Order Management System' selected. Below it is a 'Cluster Outline' section with 'OMS' selected. The main area is titled 'Configuration and Setup For OMS - Offline Catalog Configuration - Common Configuration'. It contains a table with the following properties:

Property	Value	Description
Polling interval in seconds	60	Polling interval in seconds



Only one member instance configured by the property "Designated model loading member instance" (com.tibco.af.oms.model.loading.member) is responsible for polling the offline folder and processing models.

2. SOAP Web Service

A SOAP/ HTTP web service is available to read the offline model files on demand without waiting for the poller process to trigger.

Offline catalog invocation web service WSDL is `$OM_HOME/schemas/wsd1/http/OfflineCatalogue.wsdl`.

The web service is available on the following URL: `http://<HOST>:<PORT>/api/offlineCatalogueWS`: where the HOST and PORT are the hostname and the port on which the omsServer microservice is running.

The sample web service request for reading offline product models is available at `$OM_HOME/samples/OfflineCatalogRequest.xml`.

Regardless of the way the offline model files are read, the model payload is processed in exactly the same way as mentioned in [Model Processing](#) in the online model loading section.



For reading the offline catalog files using web service, the flag to enable offline mode for each catalog does not have to be set. It can be kept as `false`.

Model Processing

This topic explains model processing details such as model storage, data extraction, server cache population, cache validity, repository cache rebuild, and parallel processing of models.

Model Storage and Publishing

1. The product, offer ID mapping, and action are stored in the DATA_MODEL table in the database so that they can be loaded from the database on the next engine restart.

The plan fragment model is stored in the PLAN_FRAGMENT table so that a particular plan fragment model can be accessed by Orchestrator anytime during the plan execution.

2. The plan fragment model payload is converted into the process component model and published on a designated topic mentioned in the table below. The Jeopardy Management System component subscribes to this topic. On receiving the message, Jeopardy Management System creates in-memory objects for using the models readily.

The DATA_MODEL table is the primary storage location for all loaded models.

Model	Published on JMS Topic
Product	tibco.aff.ocv.events.products.publish
Action	tibco.aff.ocv.events.actions.publish
Offer ID Mapping	tibco.aff.ocv.events.products.publish
Plan Fragment (Process Component)	tibco.aff.jm.model.processComponent.pub

Data Extraction

The process of extracting model information is the same regardless of the online or offline model processing mode. The XML data models information is extracted from the files and the in-memory cache objects are populated with that information.

Server Cache Population

As part of performance improvements in the model loading process, the member instance designated to process models writes in-memory cache repository information generated by reading the models to the database SERVER_CACHE table.

1. The model cache is broken down into multiple model types. These types are PRODUCT, ACTION, and OFFERID. Based on the published models, cache information for each category is populated in the SERVER_CACHE table.
2. Based on the maximum size of the in-memory model information for each model cache, this information is broken down into multiple rows.
3. When Automated Order Plan Development (deployed either as a standalone or collocated instance) starts up, embedded model load engine components first check to see if valid cache information exist in the SERVER_CACHE table. If cache information does not exist in the table or is not valid, the XML models are read from the DATA_MODEL table and the in-memory server cache information is regenerated.
4. If valid cache information exists in the SERVER_CACHE table, this information is read instead of processing the XML models. This step saves a considerable amount of processing time as SERVER_CACHE information stores the pre-processed XML information.
5. When a designated member instance processes all the models and writes information to the SERVER_CACHE table, this instance sends out a notification on the TIBCO EMS topic (tibco.aff.oms.events.refresh.cache) for all Automated Order Plan Development instances to read the updated cache information from the SERVER_CACHE table. Until this notification is processed, and Automated Order Plan Development continues to hold the existing in-memory information.
6. While processing models for creating in-memory repository information, all the models are processed. The existing configuration parameter "Max No of Model cached" is still honored by Automated Order Plan Development.

Cache Validity

- The SERVER_CACHE table maintains cache validity that signifies if the cache information stored in the table is valid or not. The value one (1) signifies a VALID cache and value zero (0) signifies an INVALID cache.
- All the offline or online models are posted by the application and application cache is written to the database after all the models are processed. Cache validity is switched to valid one (1) after all the database write operations are performed.

Repository Cache Rebuild

When the SERVER_CACHE table is empty or has an invalid cache validity, the restarting application server processes the corresponding XML models from the DATA_MODEL table and repopulates the cache information in the SERVER_CACHE table. It also sets the validity of all the rows for the model type to a VALID (value of 1) status.

Parallel Processing of Models

As part of speeding up model processing time, designated member instance processes all models (PRODUCT, ACTION, and OFFERID Mapping) in parallel.

Enabling and Disabling Model Cache Persistence

The functionality of writing in-memory repository information to the SERVER_CACHE table can be enabled or disabled by the configuration parameter "Enable Model Cache Persistence" (com.tibco.af.oms.model.cache.enabled) in the ConfigValues_OMS file.

```
<ConfValue description="Enable Model Cache Persistence" name="Model Cache Persistence"
propname="com.tibco.af.oms.model.cache.enable" sinceVersion="3.0" visibility="Basic">
  <ConfString default="true" value="true"/>
</ConfValue>
```

The value for this flag can either be TRUE or FALSE. The default value for this flag is TRUE. Set the Enable Model Cache Persistence parameter to TRUE if you want the application's in-memory cache repository information written to the SERVER_CACHE table in the serialized object format. Set the Enable Model Cache Persistence parameter to FALSE if you want the application to not write any cache repository information to SERVER_CACHE table; the SERVER_CACHE table remains empty if it is set to FALSE.

Enabling and Disabling Posting Models on Enterprise Message Service topics

The functionality of posting models on TIBCO Enterprise Message Service can be enabled or disabled using the configuration parameter "Post Models on Enterprise Message Service for Automated Order Plan Development" (com.tibco.af.oms.model.ems.post.disabled) in the ConfigValues_OMS file.

When the server cache information is read by Automated Order Plan Development, they do not have to process model XMLs again, therefore model XMLs are not required to be posted on the TIBCO Enterprise Message Service topic when the cache persistence is enabled.

```
<ConfValue description="Post Models on EMS for AOPD and OPE Disabled" name="Post
Models on EMS for AOPD and OPE Disabled"
propname="com.tibco.af.oms.model.ems.post.disabled" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="true" value="true"/>
</ConfValue>
```

The value for this flag can either be TRUE or FALSE. The default value for this flag is TRUE. Set the Post Models on Enterprise Message Service Topic for Automated Order Plan Development parameter to FALSE if you want XML models posted on the TIBCO Enterprise Message Service topic, which is an existing functionality before TIBCO Order Management - Long Running version 3.0.2. Set the Post Models on Enterprise Message Service Topic for Automated Order Plan Development parameter to TRUE if you do not want XML models posted on the TIBCO Enterprise Message Service topic to save model processing time.



It is good practice to set the value of this flag to TRUE to disable posting XML models on the Enterprise Message Service topic when cache persistence is enabled.



In an unlikely scenario where in-memory cache repository information is not written to the SERVER_CACHE table, either due to an exception or due to invalid cache validity, this flag is overwritten by the application and models are still posted to the TIBCO Enterprise Message Service topic for Automated Order Plan Development to consume.

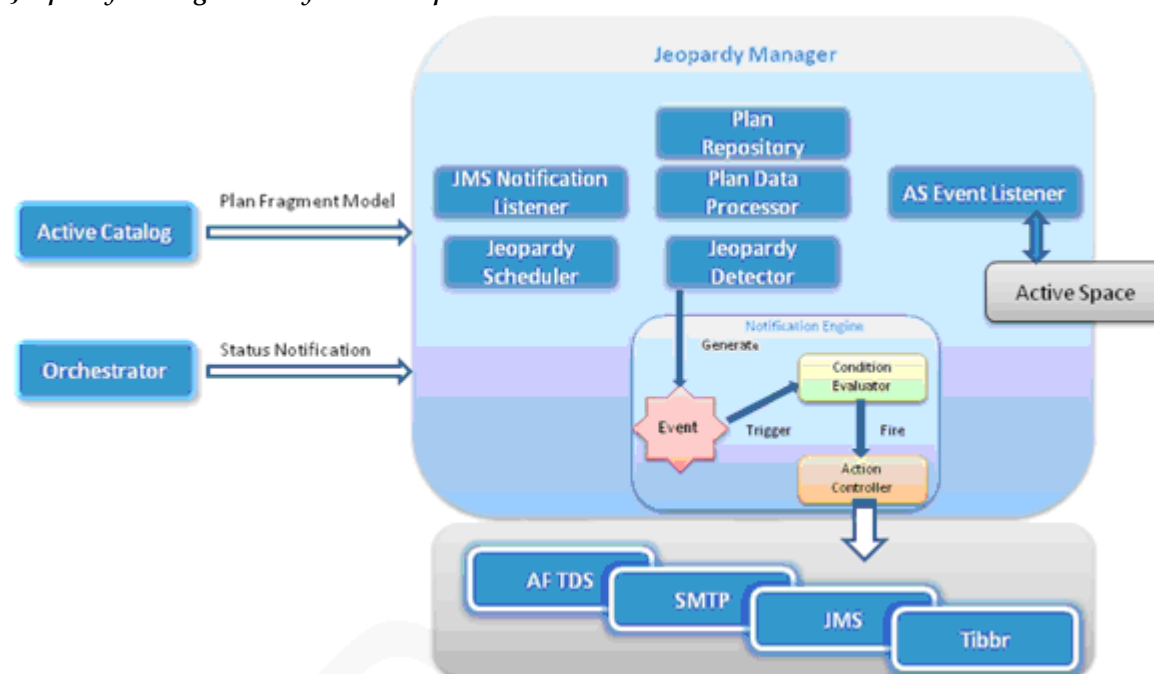
Jeopardy Management System

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:

Jeopardy Management System Component Architecture



Jeopardy Management System Configuration

All the parameters of the Jeopardy Management System have default values and no parameter value required to deploy and run Jeopardy Management System in basic mode. You might 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 a percentile increase over typical duration beyond which a task can be considered to be running in a hazard region. For example, if a typical duration of a process component is 60 minutes and if risk the threshold is 50, any plan item that is executing for more than 90 minutes is considered to be running in a hazard region. The default value is 25 percentile points.

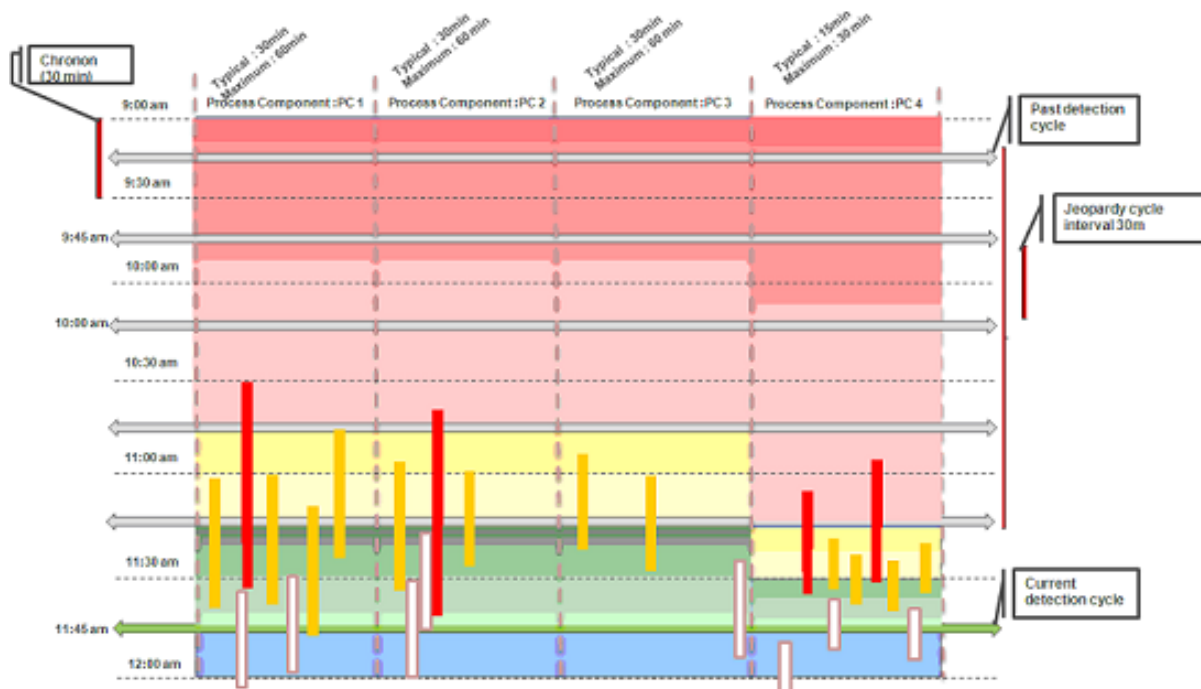
Out of Scope Threshold

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 the Jeopardy Management System stops monitoring the plan.

Time Window Value/Time Window Unit

The Configuration parameter Time window value and Time Window Unit are combined to determine the Time window or Chronon to which plan items of execution plans belong in the Jeopardy Management System. 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 the execution characteristics of process components in the fulfillment ecosystem. This value is used by Jeopardy Management System to cluster plan items in execution to optimize jeopardy detection cycle.

Time Window Value/Time Window Unit



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

Messaging Configuration

Jeopardy Management System piggyback on message configuration parameters of Order Management Server and no additional configuration required.

Datasource Configuration

Jeopardy Management System piggyback on data source configuration parameters of Order Management Server and no additional configuration required.

Colocated Jeopardy Mode

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

Two tables are responsible for the cache data store. The tables are:

1. TIME_WINDOW
2. DAMPENING_CRITERIA

Change in Jeopardy Management System

If the node status has not been set to STARTED, none of the Jeopardy Management System requests are processed. Currently, all Jeopardy Management System requests are processed by the state machine. Jeopardy Management System waits for the node to achieve the STARTED state to begin jeopardy monitoring.

Order Management Server Components

The Order Management System consists of the following application components:

- **Order Management Server** - Core Order Management 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 Order Management Server UI. This component also provides plan generation, orchestration, and jeopardy management capabilities.
- **Order Management Server UI** - provides Web-based interface to browse orders and execution plans and perform actions on the orders.
- **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.
- **Automated Order Plan Development** - This component provides plan generation capability in standalone mode.

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

Order Management Server Configuration

Messaging Configuration

All the application components of TIBCO Order Management - Long Running use JMS as one of the conduits for passing a message within TIBCO Order Management - Long Running and with the external applications.

The following properties are required to connect to the JMS server:

Messaging Configuration

The screenshot displays the 'Configuration and Setup For OMS - Messaging Configuration' window. On the left is a sidebar with a 'Select Configuration' dropdown set to 'Order Management System'. Below this is a 'Cluster Outline' with 'OMS' selected. Further down is a 'Configuration Outline' with tabs for 'Basic' and 'Advanced', and a list of configuration categories including 'Messaging Configura...'. The main panel shows a table of properties for the selected configuration.

Property	Value	Description
JNDI Connection factory JNDI Name	GenericConnectionFactory	JNDI Connection factory JNDI Name
Topic Connection factory JNDI Name	TopicConnectionFactory	Topic Connection factory JNDI Name
JNDI URL	tibjmsnaming://[redacted]:7333	JNDI URL for JMS Service
JNDI Username	admin	JNDI Username
JNDI Password	*****	JNDI Password
Orchestrator Connection factory JNDI Name	QueueConnectionFactory	Orchestrator Queue Connection factory JNDI Name
Orchestrator JMS Delivery Mode	2	Orchestrator JMS Delivery Mode

Parameters	Description
Host	JMS provider hostname
Port	Port number of the JMS Provider

Parameters	Description
Username	Username
Password	Password

Jeopardy deployment in the collocated mode does not require jeopardy queues and bridges for communication. The collocated 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 set up as a part of an earlier version, then you can use the *JEOMS_DeleteEMSChannel.txt* to remove the queues and bridges.

User Interface Configuration

Order Management Server provides a web user interface to browse and perform actions on the orders and execution plans. Order Management Server UI is deployed as a separate application, and it requires parameters to connect to the Order Management Server application.



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

User Interface Configuration

Configuration and Setup For OMS - OMS User Interface

Buttons: Add New Property, Clone, Delete

Property	Value	Description
Web Service HTTP Transport Channel Type	http	
OMS UI HTTP Port Number	9092	
OMS UI HTTPS Port Number	8443	
OMS Server Host Name	localhost	Host address of the OMS server
Owner for FP	FP	Owner for FP
OMS Server Port Number	9091	Port number of the OMS Server
FP Server node name	knode	Node name of the FP server

The following ports must be configured in case of plan preview with standalone Automated Order Plan Development:

- **localhost:** host for standalone Automated Order Plan Development used for plan preview from Order Management Server UI
- **port:** port for standalone Automated Order Plan Development used for planpreview from Order Management Server UI.

<div> <div>Select Configuration</div> <div>Order Management System</div> <div>Cluster Outline</div> <div>OMS</div> <div>Configuration Outline</div> <div>Basic Advanced</div> <ul style="list-style-type: none"> messaging Configur... Data Source Configur... Data Source Configur... Persistence Catalog Publishing JM... Offline Catalog Config... Standalone AOPD Int... Data Interfaces Confi... Application Security Web Service Configu... </div>		
<div>Configuration and Setup For OMS - Web Service Configuration</div> <div> <div>Add New Property</div> <div>Clone</div> <div>Delete</div> </div>		
Property	Value	Description
Enable User Name token based Security	true	Enable User Name token based Security
Enable Schema validation	true	Enable Schema validation
Enable Order Receiver Idempotency	true	Enable Submit Order Web Service Idempotency
HTTP Channel type	http	HTTP Channel type
HTTP Port Number	9091	
HTTPS Port Number	8443	
Use external business transactionid as business transaction id within	true	Use external business transactionid as business transaction id within OMS
Standalone/Custom AOPD host name	localhost	Standalone/Custom AOPD host name

The following table shows configurable parameters for the UI.

Parameters	Description
Maximum Session Per User	Number of sessions allowed per user. By default, Order Management Server creates only one session 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.
Error If Maximum Session Expired	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) cannot log in again until the original session expires.
Http Session Fixation Protection	<p>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:</p> <ul style="list-style-type: none"> 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 is retained. newSession - Create a new, clean session without copying the existing session data.

Plan Preview Integration with Order Management Server UI

Order Management ServerUIClient.jar is provided to let users to directly access Order Management Server UI components like Orders, Plan, Rule, and activityLog through third-party apps.

It must be placed in your library or classmate. Implementing OMSUIClient.jar in an app introduces Oauth2 authorization with Order Management Server UI, which helps the third-party apps to internally login to Order Management Server UI. It can also directly access Order Management Server UI components based on the target.

Implementing OMSUIClient.jar

Procedure

1. The following dependencies can be added to access omsuiClient.jar:

```
<!-- TIBCO Dependencies -->

<dependency>
  <groupId>com.tibco.aff</groupId>
  <artifactID>logClient</artifactID>
</dependency>

<dependency>
  <groupId>com.tibco.aff</groupId>
  <artifactID>omsuiClient</artifactID>
</dependency>

<!-- End: TIBCO dependencies -->
```
2. Get access token through OMSUIClient for Order Management Server UI. The steps to access the token are as follows:
 - a) **Access Token URL:** http://[hostname]:[PortName]/omsui/oauth/token.
 - b) **ClientID:** Provided by Order Management Server UI (for example: my-trusted-client-with-secret).
 - c) **UserName:** User Name for Order Management Server UI authorization.
 - d) **Password:** Password for Order Management Server UI authorization.
 - e) **clientSecret:** A secret key provided by Order Management Server UI.
3. OMSUIClient fetches access tokens for Oauth2 authorization; the user can then add the token in target URL to access the Order Management Server UI.

URL to Access Order Management Server UI Component

Users can directly access Order Management Server UI by providing target components for each order, dashboard, plan, ruleconfig, catalog, and activitylog.

The target parameter in URL redirects to specific component of Order Management Server UI, for example http://localhost:9092/omsui/OTS/main?target=order redirects to the Order's tab of Order Management Server UI.

Observe the following two scenarios:

Using Order Management Server UI Client	The URL is redirected to a specific component, based on the target and the search parameters.
Directly Accessing Order Management Server UI URL with the target specified	Initially the user is redirected to the login page and once the authentication completes, the user is redirected to the specific component based on the target and the search parameter.

Target Parameters for Order Management Server UI

The target parameters for Order Management Server UI are as follows:

- Order
- Dashboard
- catalog
- Ruleconfig
- ActivityLog

Additional Parameters for Order Management Server UI

Apart from target parameters users can add other parameters for order, plan, and activitylog for specific search.

The parameters for search are as follows:

Order	orderRef orderID status
Plan	orderRef planID
ActivityLog	Type (orderRef/plan)

Router Configuration

The *Content-based router* in Order Management Server lets 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 where an order might be routed for fulfillment, content-based routing provides a high degree of flexibility to configure multiple types of Orchestration engines.

Order Management Server supports two types of routers:

1. **Pass-through router (passthroughRouter)**, and
2. **Filter based router (filteringRouter)**.

To select router type, you must be at the Order management System level configuration:

1. Go to **Settings** under "Hi, admin!"
2. Edit router configuration.
3. Select the type of router you want to use. Default router type is `passthroughRouter`

Router Configuration

TIBCO® FOM Configurator
Home Save Tools
Hi, admin!

Settings
Deployment Targets for Cluster: InitialConfig

Router Configuration	passthroughRouter <input checked="" type="radio"/> passthroughRouter <input type="radio"/> filteringRouter Save Changes Cancel
Application Security	Default Authentication Provider Edit
Order Summary Data Collection Method	onStatusChange Edit

Visibility Options

Display Hidden Configurations	Hidden configurations are displayed. Edit
Display Hidden Properties	Hidden properties are displayed. Edit

Router Types and Properties

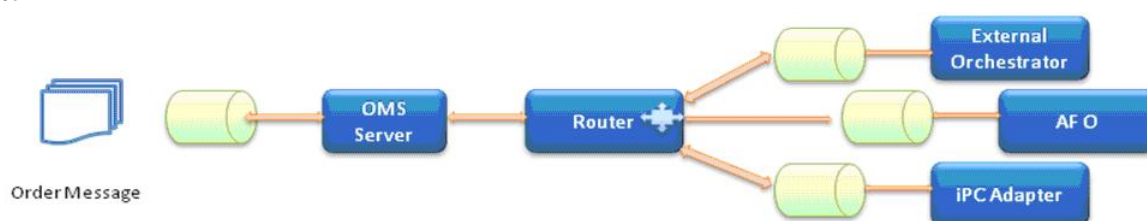
Configuration and Setup For OMS - Router Configuration - passthroughRouter		
Add New Property Clone Delete		
Property	Value	Description
Orchestrator Cancel Order Queue	tibco.aff.orchestrator.order.submit	Cancel Order Queue for Orchestrator
Router Type	passthroughRouter	Router Type
Orchestrator Submit Order Reply Queue	tibco.aff.orchestrator.order.submitResponse	Submit Order Queue for Orchestrator
Request timeout for synchronous order submission	30000	Request time out for synchronous order submission
Orchestrator Submit Order Queue	tibco.aff.orchestrator.order.submit	Submit Order Queue for Orchestrator
Orchestrator Amend Order Queue	tibco.aff.orchestrator.order.submit	Amend Order Queue for Orchestrator
Orchestrator Withdraw Order Queue	tibco.aff.orchestrator.order.withdraw	Withdraw Order Queue for Orchestrator
Orchestrator Resume Order Queue	tibco.aff.orchestrator.order.activate	Resume Order Queue for Orchestrator

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.

Parameters	Description
Router Condition	XPath filter condition to be applied on the incoming order message. If the XPath condition is not satisfied, the message is routed to the default Orchestrator or else the message is routed to the iProcess Conductor destination.
Orchestrator Submit Order Queue	Destination queue name of Orchestrator for submit order request.
Orchestrator Amend Order Queue	Destination queue name of Orchestrator for amend order request.
Orchestrator Suspend Order Queue	Destination queue name of Orchestrator for suspend order request.
Orchestrator Activate Order Queue	Destination queue name of Orchestrator for activate order request.
iProcess Conductor Orchestrator Submit Order Queue	Destination queue name of iProcess Conductor for submit order request.
iProcess Conductor Orchestrator Amend Order Queue	Destination queue name of iProcess Conductor for amend order request.
iProcess Conductor Orchestrator Suspend Order Queue	Destination queue name of iProcess Conductor for suspend order request.
iProcess Conductor Orchestrator Activate Order Queue	Destination queue name of iProcess Conductor for activate order request.

Router



The functional support for TIBCO iProcess Conductor (iPC) has been deprecated in TIBCO Order Management - Long Running version 2.0.0. However, orders to be sent to iProcess Conductor are still routed by Order Management Server to the above-listed iProcess Conductor 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 is always 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 with name value pair 'Orchestrator' and iProcess Conductor' to TIBCO iProcess Conductor. It is not required to be based on the User Defined Field element. The only requirement is that it might be a valid XPath condition on the order schema. TIBCO Order Management - Long Running 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 TIBCO Order Management - Long Running Configurator.



1. Any XPath filter condition results in the Orchestrator name.
2. Orchestrator names are case-sensitive. The Order Management Server router matches the text value to the Orchestrator name (case-sensitive). If the text value is `iProcess`, 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 Order Management Server component. This file is available inside `$OM_HOME/roles/omsServer/standalone/services/omsServer/omsServer-5.0.0-SNAPSHOT.jar`.

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

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



You must restart the Order Management Server to apply the updated XPath filter condition after updating the router information in Configurator.

Routing Orders to Other Engines

Order Management Server routes orders to Advanced Fulfillment Orchestration and iProcess Conductor.

To enable Order Management Server to route an order to any other engine, perform the following steps:

1. Configure the `router-context.xml` file.
2. Change ConfigValues using Advanced Fulfillment Configurator.

Configuring Router Context

Router uses XML configuration to configure routing and mediation rules, which are added to a `router-context.xml` of the Order Management Server. This file is available inside `$OM_HOME/roles/omsServer/standalone/services/omsServer/omsServer-5.0.0-SNAPSHOT.jar`.

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

The following sections take 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>
                <to uri="bean:ipcSubmitOrderRouter" pattern="InOnly" />
                <transform>
                    <simple> IPC</simple>
                </transform>

                <to uri="bean:routerDataProcessor"/>
            </when>
            <when>
                <xpath>${orchestrator} = 'KPSA'</xpath>
                <to uri="bean:kpsaSubmitOrderRouter" pattern="InOnly" />
                <transform>
                    <simple>KPSA</simple>
                </transform>

                <to uri="bean:routerDataProcessor"/>
            </when>
            <otherwise>
                <to uri="bean:beoSubmitOrderRouter" pattern="InOnly" />
                <transform>
                    <simple>AFO</simple>
                </transform>

                <to uri="bean:routerDataProcessor"/>
            </otherwise>
        </choice>
    </route>

```

2. Changes to ConfigValues using Advanced Fulfillment Configurator.

- a. Add a new property in the config values for the submit order destination according to 1.b (*Assign a destination queue*).

Multi-Tenancy Configuration

To use the multi-tenancy feature for Order Management Server, configure the appropriate properties as necessary.



When there are multiple tenants in TIBCO Order Management - Long Running, then you must specify the TENANTID in the TDS requests so that the order management system knows which tenant the request is related to.

Database details for the admin schema are provided in ConfigValues_OMS.xml under the category "Data Source Configuration" and the database details for the default tenant are provided in ConfigValues_OMS.xml under the category "Data Source Configuration Default Tenant".

You can use the following property under the category "Orchestrator Configuration" to determine messages sent to the process component is sent to a new JMS destination or not:

```
<ConfValue description="Flag to enable or disable using the tenant specific
destination for process component" isHotDeployable="true" name="Tenant specific
destination for process component destinations"
propname="com.tibco.fom.orch.tenantSpecificDestination" readonly="false"
sinceVersion="3.1" visibility="Basic">
  <ConfBool default="false" value="false"/>
</ConfValue>
```

If this property is set to true, the messages are sent to a JMS destination prefixed with the tenant ID to the existing destination as follows:

```
<TENANTID>.tibco.aff.orchestrator.planItem.execute.request
```

Along with the existing property com.tibco.fom.orch.overridePlanfragmentDestination, the new flag com.tibco.fom.orch.tenantSpecificDestination works in the following manner:

- If com.tibco.fom.orch.overridePlanfragmentDestination is set to true and com.tibco.fom.orch.tenantSpecificDestination is set to false, the messages are sent to the configured destination for the respective process component.
- If com.tibco.fom.orch.overridePlanfragmentDestination is set to false and if com.tibco.fom.orch.tenantSpecificDestination is true, then
 - if the owner is defined for this process component, the JMS destination is tibco.aff.orchestrator.planItem.<planFragment-owner>.execute.request.
 - if the owner is not defined for this process component, the JMS destination is <TENANTID>.tibco.aff.orchestrator.planItem.execute.request.
- If com.tibco.fom.orch.overridePlanfragmentDestination is set to false and if com.tibco.fom.orch.tenantSpecificDestination is false, then
 - if the owner is defined for this process component, the JMS destination is tibco.aff.orchestrator.planItem.<planFragment-owner>.execute.request.
 - if the owner is not defined for this process component, the JMS destination is tibco.aff.orchestrator.planItem.execute.request.



By default, both properties are set to false.

Managing Application Security

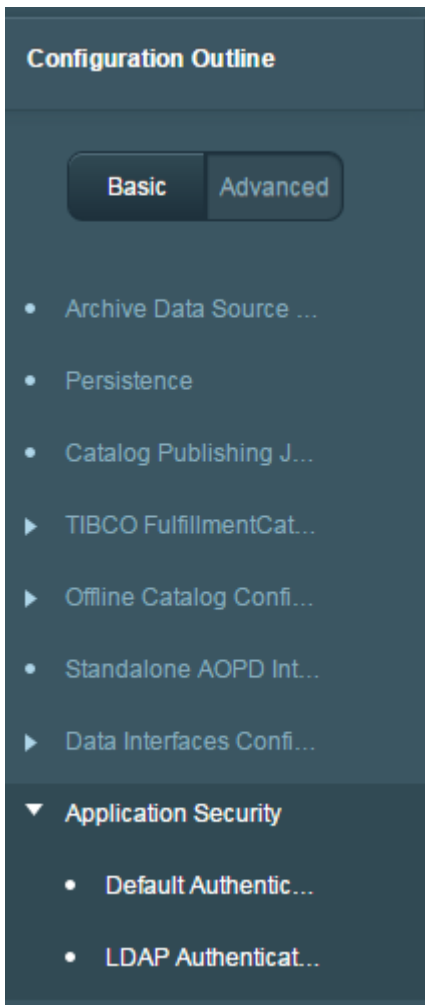
Order Management Server provides two application-level security options.

- Default Authentication Provider
- Lightweight Directory Access Protocol (LDAP) Authentication Provider

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

Order Management Server Application Security

The screenshot shows the TIBCO FOM Configurator interface. The top navigation bar includes 'Home', 'Save', 'Tools', 'Search Property', and a user profile 'Hi, admin!'. The 'Settings' menu is open, showing 'Settings', 'Help', and 'Logout'. The main content area is titled 'Settings' and 'Deployment Targets for Cluster: InitialConfig'. Under the 'Router Configuration' section, the 'Application Security' tab is selected. The 'LDAP Authentication Provider' is chosen, with options for 'Default Authentication Provider' and 'LDAP Authentication Provider'. The 'LDAP Authentication Provider' is selected with a radio button. Below the selection, there are 'Save Changes' and 'Cancel' buttons. The 'Order Summary Data Collection Method' is set to 'onStatusChange'. The 'Visibility Options' section includes 'Display Hidden Configurations' and 'Display Hidden Properties', both with 'Edit' links.



Order Management Server also supports Lightweight Directory Access Protocol-based authentication.

Lightweight Directory Access Protocol Authentication Properties

Configuration and Setup For OMS - Application Security - LDAP Authentication Provider		
<input type="button" value="Add New Property"/> <input type="button" value="Clone"/> <input type="button" value="Delete"/>		
Property	Value	Description
User Search Filter	(uid={0})	User Search Filter
Ldap Authentication Provider	IdapAuthenticationProvider	Ldap Authentication Provider
LDAP User Manager ID	uid=admin,ou=system	LDAP User Manager ID
LDAP User Manager password	*****	LDAP User Manager Password
User SearchBase	ou=people	User SearchBase
LDAP Server URL	ldap://localhost:389/dc=oms,dc=org	LDAP Server URL
Group Search Base	ou=groups	Group Search Base
Group Role Attribute	cn	Group Role Attribute
Group Search Filter	uniqueMember={0}	Group Search Filter

The following properties are required to configure the Order Management Server to use external Lightweight Directory Access Protocol server authentication.

Parameters	Description
Lightweight Directory Access Protocol Server URL	Lightweight Directory Access Protocol Server URL ldap://<hostname>:port/<root context>. Many Lightweight Directory Access Protocol servers also support SSL-encrypted Lightweight Directory Access Protocols, preferred for security purposes and to configure Order Management Server to use SSL Lightweight Directory Access Protocol to connect to server use ldaps:// at the beginning of the Lightweight Directory Access Protocol server URL.
Lightweight Directory Access Protocol User Manager DN	User Manager Distinguished Name to be used to connect to Lightweight Directory Access Protocol Server.
Lightweight Directory Access Protocol User manager Password	Password of the user manager to be used for authentication.

Parameters	Description
User Search Base	A <i>search base</i> (the distinguished name of the search base object) defines the location in the directory from which the Lightweight Directory Access Protocol user search begins.
User Search Filter	Search filter to be used to locate the user. For example, use the following filter to substitute the login name with value for the uid (filter) in the directory: filter (uid={0})
Search Subtree	Flag to enable deep search through the subtree of the Lightweight Directory Access Protocol Server URL + User Search Base. True by default.
Group Search Base	It defines the base DN under which the Lightweight Directory Access Protocol integration might look for one or more matches for the users' DN. The default value performs a search from the Lightweight Directory Access Protocol root.
Group Search Filter	It defines the Lightweight Directory Access Protocol search filter used to match user's DN to an attribute of an entry located under Group Search Base. The default value is (uniqueMember={0}).
Group Role Attribute	It defines the attribute of the matching entries, which is used to compose the user's role in Order Management Server. The default value is <i>cn</i> . Attribute must have either admin or user as the value for the role attribute. Role-based authorization provided by Order Management Server depends on the value specified in this attribute to provide appropriate permission for the user.

Managing Users and Roles

Order Management Server 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.

Order Management Server Interface	Function	Roles
Order Management Server UI	Dashboard	ROLE_USER, ROLE_ADMIN
	Search Order	ROLE_USER, ROLE_ADMIN
	Order Detail	ROLE_USER, ROLE_ADMIN
	Suspend Order	ROLE_ADMIN
	Cancel Order	ROLE_ADMIN
	Resume Order	ROLE_ADMIN
	Amend Order	ROLE_ADMIN

Order Management Server Interface	Function	Roles
	Withdraw Order	ROLE_ADMIN
	Search Execution Plan	ROLE_USER, ROLE_ADMIN
	View Execution Plan	ROLE_USER, ROLE_ADMIN
	Add Order	ROLE_ADMIN
Order Management Server Web Service	Submit Order	ROLE_ADMIN
	Synchronous Submit Order	ROLE_ADMIN
	Amend Order	ROLE_ADMIN
	Get Orders	ROLE_USER, ROLE_ADMIN
	Get Order Detail	ROLE_USER, ROLE_ADMIN
	Get Execution Plan	ROLE_USER, ROLE_ADMIN
	Get Enriched Execution Plan	ROLE_USER, ROLE_ADMIN
	Cancel Order	ROLE_ADMIN
	Activate Order	ROLE_ADMIN
	Perform Bulk Order Action	ROLE_ADMIN

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

The following table shows the default user id and password provided by the Order Management Server.

User Name	Password	Role
admin	admin	ROLE_ADMIN
affadmin	affadmin	ROLE_ADMIN
afuser	afuser	ROLE_USER

Mentioned below is the UserManagement application available with TIBCO Order Management - Long Running.

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

`http://<host>:<port>`

Where:

- `host` is the computer where you installed the Fulfillment Order Management.
- `port` number is the port number of the machine where the Order Management Server Web server is listening to requests. The default port number is **9091**.
- `omsServer` is the application context.

The User Management application is used to manage users in the TIBCO Order Management - Long Running solution. The application has a *userservice* command line interface that helps 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 a role assigned as `ROLE_ADMIN` 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. Set the Valid Context URL in the following format:
`export OM_OMS_CONTEXT_URL=http://localhost:port`
2. On the command prompt, access the `$OM_HOME/roles/userClient/standalone/bin` directory.
3. Run the following command:

```
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 `$OM_HOME/bin` directory.
2. Run the following command:

```
$OM_HOME/roles/userClient/standalone/bin/userservice.sh -action read
-adminusername <adminusername> -adminuserpassword <adminpassword>
-username <user>
```

Deleting User

1. On the command prompt, access the `$OM_HOME/roles/userClient/standalone/bin` directory.
2. Run the following command:

```
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 `$OM_HOME/roles/userClient/standalone/bin` directory.

2. Run the following command:

```
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", and so on.

Load Balancing

Order Management Server itself does not provide any load balancing capability, but any third-party load balancer can be used to load balance across multiple instances of Order Management Server. For this, no specific configuration is required. The only requirement is that the load balancer must have support for sticky sessions. Sticky sessions mean 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 traffic and maintain session persistence for Web applications without relying on the user's IP address with session cookies. Typically, Hardware Load Balancer inserts a cookie that the load balancer creates and manages automatically to remember, which back-end server a given HTTP connection must 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:

- NetScaler
- Barracuda
- jetNEXUS

Order Management Server Web Service

Order Management Server supports both HTTP and JMS as transport protocols for invoking SOAP-based web services. Order services in Order Management Server can be secured by enabling the user name token-based security. Order Management Server supports the WS-Security UserName Token mechanism, which lets 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 user name 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/oasis-wss-saml-token-profile-1.0.pdf">
admin</wsse:Password>
      <wsse:Nonce>WScqanjCEAC4mQoBE07sAQ==</wsse:Nonce>
      <wsu:Created>2010-05-11T01:24:32Z</wsu:Created>
    </wsse:UsernameToken>
  </wsse:Security>
</soapenv:Header>
```

Order Management Server Web Service

Configuration and Setup For OMS - Web Service Configuration		
<input type="button" value="Add New Property"/> <input type="button" value="Clone"/> <input type="button" value="Delete"/>		
Property	Value	Description
HTTPS Port Number	8443	
Enable User Name token based Security	true	Enable User Name token based Security
Enable Order Receiver Idempotency	true	Enable Submit Order Web Service Idempotency
HTTP Channel type	http	HTTP Channel type
HTTP Port Number	8080	
Enable Schema validation	true	Enable Schema validation
Use external business transactionId as business transaction id within	false	Use external business transactionId as business transaction id within OMS
Standalone/Custom AOPD host name	localhost	Standalone/Custom AOPD host name

The following table shows configurable properties for order related WebServices in Order Management Server.

Parameters	Description
Enable User Token based Security	The Order service provided by Order Management Server can be secured by enabling username token-based security.
Enable Schema Validation	Defines a flag to specify if schema validation is required on the order requests submitted to Order Management Server.
Enable Order Receiver Idempotency	Making the order Web services idem potent lets 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).
HTTP Channel Type	Defines channel type to be used for the transport. Specifying channel type to be HTTPS lets the client and server to use mutual authentication and encrypts the communication.
HTTP Port Number	Port number of HTTP Channel. This port number must match the port number specified for the HTTP port transport.

Parameters	Description
HTTPS Port Number	Port number of HTTPS Channel. This port number must match the port number specified for the HTTPS transport.

Order Management Server Web Service Authentication

A new token-based authentication has been introduced for the Order Management Server order soap web service using JSON Web Token (JWT). The operation `FetchAuthenticationToken` has been added to fetch the token-based on the existing username and password. This token can be used to invoke other operations of the order service.

To use this form of authentication, the client can keep the password empty and send the fetched token as the username in the SOAP request.

Sample Request for `FetchAuthenticationToken`

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ord="http://www.tibco.com/aff/orderservice" xmlns:aut="http://www.tibco.com/aff/
authentication">
  <soapenv:Header/>
  <soapenv:Body>
    <ord:AuthenticateRequest>
      <aut:username>admin</aut:username>
      <aut:password>admin</aut:password>
    </ord:AuthenticateRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample Response for `FetchAuthenticationToken`

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AuthenticateReply xmlns:ns8="http://www.tibco.com/aff/enrichedPlan"
xmlns:ns7="http://www.tibco.com/aff/planfragments" xmlns:ns6="http://www.tibco.com/aff/
plan" xmlns:ns5="http://www.tibco.com/aff/authentication" xmlns:ns4="http://
www.tibco.com/aff/commontypes" xmlns:ns3="http://www.tibco.com/aff/order"
xmlns:ns2="http://www.tibco.com/aff/orderservice/result" xmlns="http://
www.tibco.com/aff/orderservice">

<ns5:token>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3IiOiJhZG1pbiIsImZcyI6Ik9NUyIsInJ
scyI6W3siYXV0aG9yaXR5IjoIuk9MRV9BRE1JTjI9XSwidG50IjoIvDEiLCJleHAiOiJlE00TQyMTcyNjAsImldhC
I6MTQ5NDIxMzY2MHO.sW6zyVrPOV4g8hE-dItzriShWiT9XCVcDk0PMopm89g</ns5:token>

    </AuthenticateReply>
  </soap:Body>
</soap:Envelope>
```

Sample Security Header with JWT

```
<wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/
2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <wsse:UsernameToken wsu:Id="UsernameToken-F5E29770329D29B85614793195129081">

<wsse:Username>eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c3IiOiJhZG1pbiIsImZcyI6Ik9NUyI
sInJscyI6W3siYXV0aG9yaXR5IjoIuk9MRV9BRE1JTjI9XSwidG50IjoIvDEiLCJleHAiOiJlE00TQyMTcyNjAsIm
ldhCI6MTQ5NDIxMzY2MHO.sW6zyVrPOV4g8hE-dItzriShWiT9XCVcDk0PMopm89g</wsse:Username>

    <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText"></wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
```

Collecting Order Summary Data

Order Management Server provides the Dashboard component to view summary information about the fulfillment engine. This requires the 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 intervals rather than updating summary data on every status change. *Cron* Expression is used to define the time interval for summary collection.

For example, the default Cron Expression "0 0/10 * * * ?" defines an interval of every 10 minutes for the summary collection.

Order Summary Collection can be scheduled to run in off-peak hours and not affect the order processing. Look at highlighted values to be changed in ConfigValues_OMS.xml. The following configuration Order Summary runs every 10 minutes. Change it appropriately according to your environment and requirement.

```
Category description="Order Summary Data Collection" name="Order Summary Data
Collection" visibility="Basic">
  <ConfValue description="schedule Interval for order summary data collection."
name="Cron Expression"
propname="com.tibco.af.oms.summaryDataCollection.scheduled.cronExpress"
sinceVersion="1.1" visibility="Basic">
    <ConfString default="0 0/10 * * * ?" value="0 0/10 * * * ?"/>
  </ConfValue>
</Category>
```

By default the Cron Expression is set to get the data from server every 10 minutes. It is kept at 10 minutes so it does not interfere with the performance of the Order Management Server.

Audit Trail

Audit trail can be enabled or disabled by using the configuration parameter in ConfigValues_OMS.xml (Miscellaneous section):

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

Custom Audit Trail

It is now possible to write custom Audit Trail messages corresponding to an order present in Order Management Server. The request can be sent to TIBCO Order Management - Long Running using JMS through one of the three interfaces in the format defined in \$OM_HOME/schemas/schema/oms/CustomAuditTrail.xsd. This feature is dependent on whether audit trail is enabled or disabled.

The following are the three interfaces:

- RequestReply - provide requestReply=true in the request header to use this mode.
This sends the response to the destination specified by JMSReplyTo.
- Request - provide requestReply=false in the request header to use this mode.
This sends the response to the response queue.
- Notification - provide mode=notification in the request to use this mode.
This processes the request but does not send any response.

The following is a sample request:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:AuditTrailRequest xmlns:ns0="http://www.tibco.com/fom/customAuditTrail">
  <ns0:AuditTrailMessage>
    <ns0:orderRef>orderref_1</ns0:orderRef>
    <ns0:order>
      <ns0:auditMessage>message from PC</ns0:auditMessage>
      <ns0:messageType>INFO</ns0:messageType>
      <ns0:origin>BWSTUB</ns0:origin>
    </ns0:order>
  </ns0:AuditTrailMessage>
</ns0:AuditTrailRequest>
```

The following ConfigValues_OMS.xml properties are used for this feature:

```
<ConfValue description="Custom Audit Trail Request queue" isHotDeployable="true"
name="Custom Audit Trail Request queue"
propname="com.tibco.fom.oms.orch.customAudit.sender.queue" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="tibco.aff.orchestrator.customAudit.request"
value="tibco.aff.orchestrator.customAudit.request"/>
</ConfValue>

<ConfValue description="Custom Audit Trail Response queue" isHotDeployable="true"
name="Custom Audit Trail Response queue"
propname="com.tibco.fom.oms.orch.customAudit.response.queue" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="tibco.aff.orchestrator.customAudit.response"
value="tibco.aff.orchestrator.customAudit.response"/>
</ConfValue>

<ConfValue description="Custom Audit Trail Response queue receiver count"
isHotDeployable="true" name="Custom Audit Trail Response queue receiver count"
propname="com.tibco.fom.oms.orch.customAudit.receiver.count" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="2" value="2"/>
</ConfValue>
```

Enabling Manual Order Plan Development Support

To enable Manual Order Plan Development support, set the value of property `com.tibco.fom.orch.mopdSupported` to true. The default value of this property is set to false.

```
<ConfValue description="Enable MOPD Support" name="Enable MOPD Support"
propname="com.tibco.fom.orch.mopdSupported" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfBool default="true" value="true" />
</ConfValue>
```

Manual Order Plan Development Orders Identification

The orders eligible for Manual Order Plan Development can be identified on the basis of `orderRef` or the order User Defined Field value. The possible values for property `com.tibco.fom.orch.mopdIndicator` are `orderRef` and `User Defined Field`. The following properties determine the orders that can be processed for Manual Order Plan Development.

```
<ConfValue description="MOPD order Indicator Flag" isHotDeployable="true" name="MOPD
order Indicator Flag" propname="com.tibco.fom.orch.mopdIndicator" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="udf" value="udf" />
</ConfValue>

<ConfValue description="orderRef string for MOPD order Indicator"
isHotDeployable="true" name="orderRef string for MOPD order Indicator"
propname="com.tibco.fom.orch.mopdIndicator.orderRef.string" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="Manual_" value="Manual_" />
</ConfValue>

<ConfValue description="UDF key string for MOPD order Indicator"
isHotDeployable="true" name="UDF key string for MOPD order Indicator"
propname="com.tibco.fom.orch.mopdIndicator.udf.key.string" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="MOPD" value="MOPD" />
</ConfValue>

<ConfValue description="UDF value string for MOPD order Indicator"
isHotDeployable="true" name="UDF value string for MOPD order Indicator"
propname="com.tibco.fom.orch.mopdIndicator.udf.value.string" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="yes" value="yes" />
</ConfValue>
```

The following table explains the properties:

com.tibco.fom.orch.mopdIndicator	User Defined Field	A User Defined Field property of order is used to identify an Manual Order Plan Development order.
	orderRef	orderRef of an order is used to identify Manual Order Plan Development order.
com.tibco.fom.orch.mopdIndicator.orderRef.string	Any valid string value	An orderRef with this specified value as prefix string is eligible for Manual Order Plan Development. This property is used if the property com.tibco.fom.orch.mopdIndicator is set to orderRef.
com.tibco.fom.orch.mopdIndicator.udf.key.string	Any valid string value	A User Defined Field (UDF) with key as this specified value is eligible for Manual Order Plan Development. This property is used if property com.tibco.fom.orch.mopdIndicator is set to udf.
com.tibco.fom.orch.mopdIndicator.udf.value.string	Any valid string value	A User Defined Field with this value as the specified value is eligible for Manual Order Plan Development. This property is used if the property com.tibco.fom.orch.mopdIndicator is set to User Defined Field.

Additional Manual Order Plan Development Configuration

```
<ConfValue description="MOPD draft lock timeout in minute" isHotDeployable="true"
name="MOPD draft lock timeout in minute"
propname="com.tibco.fom.orch.mopd.draftLockTimeout" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfNum default="10" value="10" />
</ConfValue>
```

Server Side Validation of Manual Order Plan Development

Rules can be configured for server-side validation of the manually created plan. When the plan is executed then it is validated for the rules configured on the server-side. And if the validation fails the user needs to update the plan and execute it again.

All the rules for the server-side validations can be configured in file: \$OM_HOME/roles/omsServer/standalone/config/mopd_validate_config.xml including the system and custom rules.

Server side validations include:

- System rules implemented by the application - Circular dependency in plan is checked by application. If circular dependency is found in the manually edited plan, plan cannot be executed further.

- Third party rules - User can configure custom rules for validating the manually created plan. The following steps enable custom rules:
 - Define the classes implementing the custom rule. Build the code and copy the .jar file to \$OM_HOME/roles/omsServer/standalone/lib.
 - All the classes configured in step mentioned earlier must implement the interface `com.tibco.aff.eca.base.Action` and must implement a method with name `EXECUTE`. The method signature looks like the following code:

```
@Override
public void execute(MopdContext mopdContext)
{
    String currentMethod = "execute";
    if(log.isDebugEnabled())
        log.debug(null,currentMethod,null,"Entering Method {}", currentMethod);
    Plan plan = mopdContext.getPlan();
    mopdContext.setPlanValid(true);
    try
    {
        isCircularDependency(plan);
    }
    catch (Exception e)
    {
        mopdContext.setPlanValid(false);
    }
    if(log.isDebugEnabled())
        log.debug(null,currentMethod,null,"Exiting Method {} --> Validating Plan
with planId: [{]} and orderId: [{]} ", currentMethod, plan.getPlanID(),
plan.getOrderID());
    return;
}
```

- Build the code and copy the .jar file to \$OM_HOME/roles/omsServer/standalone/lib.
- Configure the rule for plan validation in file: \$OM_HOME/roles/config/standalone/omsServer/mopd_validate_config.xml.
- Restart the Order Management Server.
- Custom rules work the same way as the system rule. The bean definition for classes involved in custom rule is defined in application context file and for custom rule classes are defined and loaded.

Enabling Internal Error Handler Support

You can enable Internal Error Handler by configuring the `ConfigValues_OMS`.

```
<ConfValue description="The Error Handler component to be used in case of failed plan
item" isHotDeployable="true" name="Plan Item Error Handler Type"
propname="com.tibco.fom.orch.pcErrorHandlerType" sinceVersion="3.0" visibility="Basic">
  <ConfEnum>
    <EnumValue default="true" selected="false" value="ExternalErrorHandler" />
    <EnumValue selected="true" value="InternalErrorHandler" />
  </ConfEnum>
</ConfValue>
```

This is a new property introduced for configuring Internal Error Handler. We can have two values:

- `ExternalErrorHandler` (default)
- `InternalErrorHandler`

When it is configured as `ExternalErrorHandler` the user's implementation of error handler is considered, which means the on plan-item failure is handled by the error handler defined by user.

When the property is configured as `InternalErrorHandler`, it invokes the plan-item failure response and newly created error handler in Order Management Server.

State Machine Pagination

The state machine is initiated for each order in the heap memory. The memory grows with the number of orders that the engine is processing, Orchestrator must page these objects when it reaches the threshold.

When the number of orders hits the threshold configured, the least used objects are displaced from the memory and moved to the backing store for later use. The displaced object is saved in backing store using the State Machine Context checkpoint. When the information related to an order is required and not available in-memory, then the needed information is fetched from the backing store using the checkpoint data.

State Machine Configuration

The following configurations are required to enable pagination of state machines in the application:

Threshold can be configured using property `com.tibco.fom.orch.maxNoMilestonesLoadedinMemory` under category Orchestrator Configuration in `ConfigValues_OMS.xml`.

```
<ConfValue description="Maximum number of StateMachines instances to be kept in Heap
Memory" isHotDeployable="true" name="Max No of StateMachines in Heap Memory"
propname="com.tibco.fom.orch.maxNoMilestonesLoadedinMemory" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="0" value="0" />
</ConfValue>
```

This property indicates the number of state machine kept in heap memory. Once this threshold is exceeded eviction of state machine starts and state machines are moved to backing store. The default value of '0' indicates no eviction of state machine from heap memory and all the state machine is in heap memory.

State Machine Context Checkpoint

When a state machine is evicted from the memory, the object is saved in backing store as State Machine Context checkpoint.

State Machine Context checkpoint stores following information:

- Order ID
- Node ID
- State Machine

The checkpoint data along with resourceUpdate checkpoint data are deleted from the backing store after a regular interval when the order is completed or canceled, and are not used by Jeopardy Management System at the time it completed or canceled.

Order Capture System Configuration

Order Capture System External Component Configuration

The configuration information required from other components is gathered in `ConfigValues_OCS.xml`, which is configured through TIBCO Order Management - Long Running Configurator or TIBCO Configuration Tool.



You must ensure that all systems in the TIBCO Fulfillment Orchestration Suite (Order Capture System, Order Management Server, Offer and Price Engine, and so on) share the same version of the catalog data. This is particularly important because the Order Capture System and Order Management Server can use catalog files from different locations. By using out-of-sync catalog data in Order Management Server, Offer and Price Engine, and Order Capture System results in failures in Order Capture System. Ensure that procedures are in place to guarantee that the same catalog data are used in all systems.

Order Service Configuration


These configuration values are used to point Order Capture System to the TIBCO Order Management - Long Running Order service endpoint. Order Capture System uses these configuration values to submit the orders to TIBCO Order Management - Long Running. They also provide security credentials.


Name	Description	Default Value
<code>com.tibco.af.ocs.orderService.host</code>	The host machine where the order service endpoint is exposed.	localhost
<code>com.tibco.af.ocs.orderService.port</code>	The port where the order service endpoint is exposed.	9095
<code>com.tibco.af.ocs.orderService.url</code>	The complete URL for the order service endpoint. This is constructed using the values given for host and port.	<code>http://\${com.tibco.af.ocs.orderService.host}:\${com.tibco.af.ocs.orderService.port}/api/orderService</code>
<code>com.tibco.af.ocs.orderService.username</code>	The order service security user name.	admin
<code>com.tibco.af.ocs.orderService.password</code>	The order service security password	admin
<code>com.tibco.af.ocs.orderService.trust.store</code>	The order Service SSL trust store location.	cert/trust.jks
<code>com.tibco.af.ocs.orderService.trust.password</code>	The order Service JKS trust store password.	changeit

Subscriber Inventory Web Service Configuration

These configuration values are used to point Order Capture System to the subscriber inventory web service endpoint. Order Capture System uses these values to retrieve subscribers and subscriber data based on a query string. They also provide security credentials.

For demonstration purposes, you can use Demo Subscriber Directory in Order Capture System to manage a list of subscribers and stores. For more information, see [Demo Subscriber Directory](#).

Name	Description	Default Value
<code>com.tibco.af.ocs.demo.subscriberInventory</code>	Demo Subscriber Directory demonstrates subscriber and store management in Order Capture System.	True  Switch the value to false to turn off this feature.

Name	Description	Default Value
<code>com.tibco.af.ocs.subscriberService.host</code>	The host machine where inventory service endpoint is exposed.	localhost
<code>com.tibco.af.ocs.SubscriberInventoryService.port</code>	The port where inventory service endpoint is exposed.	9091
<code>com.tibco.af.ocs.SubscriberInventoryService.url</code>	<p>The complete URL for subscriber inventory service endpoint.</p> <p>This is constructed using the values given for host and port.</p>	<p><code>http://\$ {com.tibco.af.ocs.subscriberService.host} }\$ {com.tibco.af.ocs.subscriberService.port} } /fos/ocs/subscriber/inventory</code></p> <p> Do not change this value because the Order Capture System needs the complete and correct URL to access the subscriber inventory service endpoint.</p>
<code>com.tibco.af.ocs.subscriberService.username</code>	The subscriber inventory service security user name.	admin
<code>com.tibco.af.ocs.subscriberService.password</code>	The subscriber inventory service security password.	admin
<code>com.tibco.af.ocs.subscriberService.trust.store</code>	The subscriber inventory service SSL trust store location.	cert/trust.jks
<code>com.tibco.af.ocs.subscriberService.trust.password</code>	The subscriber inventory service JKS trust store password.	changeit

Eligibility and Pricing Web Service Configuration

These configuration values are used to configure the offer and price interface used by the Order Capture System. In a typical TIBCO Order Management - Long Running deployment, these values point to the pricing service from Offer and Price Engine. They also provide security credentials.

Eligibility Web Service Configuration

Name	Description	Default Value
<code>com.tibco.af.ocs.eligibilityService.host</code>	The host machine where the eligibility web service endpoint is exposed.	localhost
<code>com.tibco.af.ocs.eligibilityService.port</code>	The port where eligibility web service endpoint is exposed.	9094
<code>com.tibco.af.ocs.eligibilityService.url</code>	The complete URL for eligibility service endpoint. This is constructed using the values given for host and port.	<code>\${com.tibco.af.ocs.ap.protocol}://\${com.tibco.af.ocs.eligibilityService.host}:\${com.tibco.af.ocs.eligibilityService.port}/opes/api/v1/offers/eligible</code>
<code>com.tibco.af.ocs.eligibilityService.userName</code>	The eligibility web service security user name.	admin
<code>com.tibco.af.ocs.eligibilityService.password</code>	The eligibility web service security password.	admin
<code>com.tibco.af.ocs.eligibilityService.trust.store</code>	The eligibility service JKS trust store for the HTTP location.	cert/trust.jks
<code>com.tibco.af.ocs.eligibilityService.trust.password</code>	The eligibility web service JKS trust store password.	changeit


Pricing Web Service Configuration




Name	Description	Default Value
<code>com.tibco.af.ocs.pricingService.host</code>	The host machine where the pricing web service endpoint is exposed.	localhost
<code>com.tibco.af.ocs.pricingService.port</code>	The port where pricing web service endpoint is exposed	9094

Name	Description	Default Value
<code>com.tibco.af.ocs.pricingService.url</code>	The complete URL for pricing web service endpoint. This is constructed using the values given for host and port.	<code>\${com.tibco.af.ocs.ap.protocol}://\${com.tibco.af.ocs.pricingService.host}:\${com.tibco.af.ocs.pricingService.port}/opes/api/v1/offers/prices</code>
<code>com.tibco.af.ocs.pricingService.username</code>	The pricing web service security user name.	admin
<code>com.tibco.af.ocs.pricingService.password</code>	The pricing web service security password.	admin
<code>com.tibco.af.ocs.pricingService.trust.store</code>	The pricing web service JKS trust store for the HTTPS location.	cert/trust.jks
<code>com.tibco.af.ocs.pricingService.trust.password</code>	The pricing web service JKS trust store password.	changeit

Order Plan Preview Configuration

These configuration values are used by the Order Capture System to access Order Management Server UI to display an order plan preview for orders. They also provide security credentials.

Name	Description	Default Value
<code>com.tibco.af.ocs.oauthTokenService.host</code>	The host machine where order plan preview service endpoint is exposed.	localhost
<code>com.tibco.af.ocs.oauthTokenService.port</code>	The port where order plan preview service endpoint is exposed.	8080
<code>com.tibco.af.ocs.oauthTokenService.url</code>	The complete URL for order plan preview service endpoint. This is constructed using the values given for host and port.	<code>http://\${com.tibco.af.ocs.oauthTokenService.host}:\${com.tibco.af.ocs.oauthTokenService.port}/omsui/oauth/token</code>  Do not change this value because the Order Capture System needs the complete and correct URL to access the order plan preview service endpoint.

Name	Description	Default Value
<code>com.tibco.af.ocs.oauthTokenService.username</code>	The order plan preview service security user name	admin
<code>com.tibco.af.ocs.oauthTokenService.password</code>	The order plan preview service security password.	admin
<code>com.tibco.af.ocs.oauthTokenService.clientId</code>	The security client ID for the order plan preview service.	my-trusted-client-with-secret  Do not change this value.
<code>com.tibco.af.ocs.oauthTokenService.clientSecret</code>	The security secret for the order plan preview service.	somesecret  Do not change this value.
<code>com.tibco.af.ocs.planPreview.url</code>	The complete URL for the order plan preview service endpoint. This is constructed using the values given for host and port.	<code>http://\$ {com.tibco.af.ocs.oauthTokenService. host}\$ {com.tibco.af.ocs.oauthTokenService. port} /omsui/OTS/main?target=order</code>  Do not change this value because the Order Capture System needs the complete and correct URL to access the order plan preview service endpoint.
<code>com.tibco.af.ocs.oauthTokenService.trust.store</code>	The order plan preview service JKS trust store for the HTTP location.	cert/trust.jks
<code>com.tibco.af.ocs.oauthTokenService.trust.password</code>	The order plan preview service JKS trust store password.	changeit


Pooled Database Source Configuration

Order Capture System uses connection pooling along with Hibernate for connecting to the Oracle database. These configurations are used to modify the behavior of the connection pool and change Hibernate settings.

To switch the database from Oracle to PostgreSQL, see [PostgreSQL Database Configuration](#).

Name	Description	Default Value
<code>com.tibco.af.ocs.pooledDataSource.driverClassName</code>	The JDBC driver class to use	<code>oracle.jdbc.driver.OracleDriver</code>
<code>com.tibco.af.ocs.pooledDataSource.host</code>	The host machine where the database is available	localhost

Name	Description	Default Value
<code>com.tibco.af.ocs.pooledDataSource.port</code>	The database port	1521
<code>com.tibco.af.ocs.pooledDataSource.database</code>	The name of the database	orcl
<code>com.tibco.af.ocs.pooledDataSource.username</code>	The user name that is given when connecting to the database	aff_ocs
<code>com.tibco.af.ocs.pooledDataSource.password</code>	The password to be given when connecting to the database	aff_ocs
<code>com.tibco.af.ocs.pooledDataSource.url</code>	The URL connection string for the database This is constructed using the values given for the host, port, and database.	<code>jdbc:oracle:thin:@/\${com.tibco.af.ocs.pooledDataSource.host}:\${com.tibco.af.ocs.pooledDataSource.port}/ \${com.tibco.af.ocs.pooledDataSource.database}</code>
<code>com.tibco.af.ocs.pooledDataSource.initializeSize</code>	The initial size of the database connection pool	2
<code>com.tibco.af.ocs.pooledDataSource.maxIdle</code>	The maximum number of connections that might be kept in the idle pool if the pool sweeper is enabled	11
<code>com.tibco.af.ocs.pooledDataSource.maxActive</code>	The maximum number of active connections that can be allocated from this pool at the same time	12
<code>com.tibco.af.ocs.pooledDataSource.maxWait</code>	The maximum number of milliseconds that the pool waits for a connection to be returned before throwing an exception	10000
<code>com.tibco.af.ocs.pooledDataSource.validationQuery</code>	The SQL query that is used to validate connections from this pool before returning them to the caller or pool	Select 1 from dual

Name	Description	Default Value
<code>com.tibco.af.ocs.pooledDataSource.testOnBorrow</code>	The indication of whether objects are validated before being borrowed from the pool.	False
<code>com.tibco.af.ocs.pooledDataSource.testWhileIdle</code>	Set to true if query validation might take place when the connection is idle.	True
<code>com.tibco.af.ocs.pooledDataSource.timeBetweenEvictionRunsMillis</code>	The number of milliseconds to sleep between runs of the idle connection validation, abandoned cleaner, and idle pool resizing	5000
<code>com.tibco.af.ocs.pooledDataSource.minEvictableIdleTimeMillis</code>	The minimum amount of time an object must sit idle in the pool before it is eligible for eviction	5000
<code>com.tibco.af.ocs.hibernate.dialect</code>	The hibernate dialect to use.	<code>org.hibernate.dialect.Oracle10gDialect</code>  Do not change this value.
<code>com.tibco.af.ocs.hibernate.cache.use_second_level_cache</code>	Indicates if second level cache is used	false
<code>com.tibco.af.ocs.hibernate.cache.provider_class</code>	The hibernate Cache Provider	<code>org.hibernate.cache.NoCacheProvider</code>
<code>com.tibco.af.ocs.hibernate.transaction.factory_class</code>	The transaction management factory	<code>org.hibernate.transaction.JDBCTransactionFactory</code>
<code>com.tibco.af.ocs.hibernate.current_session_context_class</code>	The hibernate current session context manager	<code>org.hibernate.context.ManagedSessionContext</code>
<code>com.tibco.af.ocs.hibernate.jdbc.batch_size</code>	Enables JDBC batching and sets the batch size to a reasonable value	30
<code>com.tibco.af.ocs.hibernate.show_sql</code>	Indicates if hibernate must log generated SQL	false

Name	Description	Default Value
<code>com.tibco.af.ocs.hibernate.default_catalog</code>	The hibernate default catalog. This value might be the same as the database user name value.	<code>aff_ocs</code>

PostgreSQL Database Configuration

With the release of TIBCO Order Management - Long Running 5.0, Order Capture System now supports a PostgreSQL database instead of only an Oracle database. PostgreSQL is a powerful, open source object-relational database system, which has support in Hibernate, which is used by the Order Capture System to talk to the database. These configuration values have been updated to support switching from an Oracle database to a PostgreSQL database.

The following values can be configured in TIBCO FOM Configurator or in the `ConfigValues_OCS.xml` file.

Name	Description
<code>com.tibco.af.ocs.pooledDataSource.driverClassName</code>	The pooled data source driver name. To switch from Oracle to PostgreSQL, move <code>selected=true</code> from the Oracle line to the Postgres line: <pre><EnumValue value="org.postgresql.Driver"/></pre>
<code>com.tibco.af.ocs.pooledDataSource.host</code>	The hostname for the Postgres database.
<code>com.tibco.af.ocs.pooledDataSource.port</code>	The port number for the Postgres database.
<code>com.tibco.af.ocs.pooledDataSource.database</code>	The pooled data source database. The value for this is the name of the database you create in postgres. To create a database, see Creating PostgreSQL Tables . The default database name in postgres is postgres.
<code>com.tibco.af.ocs.pooledDataSource.username</code>	The pooled data source database user name.
<code>com.tibco.af.ocs.pooledDataSource.password</code>	The pooled data source database password.
<code>com.tibco.af.ocs.pooledDataSource.postgres.schema</code>	This is a Postgres specific property that indicates the schema name. The default schema for Postgres is public.

Name	Description
<code>com.tibco.af.ocs.pooledDataSource.url</code>	<p>The pooled data source URL. To switch from Oracle to PostgreSQL, move <code>selected=true</code> from the Oracle line to the Postgres line:</p> <pre><EnumValue value="jdbc:postgresql://\$ {com.tibco.af.ocs.pooledDataSource.host}: \$ {com.tibco.af.ocs.pooledDataSource.port}/ \$ {com.tibco.af.ocs.pooledDataSource.databa se}?currentSchema=\$ {com.tibco.af.ocs.pooledDataSource.postgr es.schema}" /></pre>
<code>com.tibco.af.ocs.pooledDataSource.validationQuery</code>	<p>The pooled data source validation query. To switch from Oracle to PostgreSQL, move <code>selected=true</code> from the Oracle line to the Postgres line:</p> <pre><EnumValue value="SELECT 1" /></pre>
<code>com.tibco.af.ocs.hibernate.dialect</code>	<p>Hibernate dialect. To switch from an Oracle database to a PostgreSQL database, move <code>selected="true"</code> from the Oracle line to the PostgreSQL line.</p>

Creating PostgreSQL Tables

A script has been added for creating Postgres tables.

Procedure

1. Create a tablespace for the Order Capture System database by running the following script:
\$OM_HOME/db/postgreSQL/oms/dbscripts/create_tablespace.sql.
2. Create the database user for Order Capture System by running the following script: \$OM_HOME/db/postgreSQL/oms/dbscripts/create_user.sql.
3. Run the following script: \$OM_HOME/db/PostgreSQL/ocs/ocs_schema_ddl.sql
Or you can copy `ocs_schema_ddl.sql` to \$OM_HOME/db/postgreSQL/oms/dbscripts and run `setup.sh` or `setup.cmd` from \$OM_HOME/db/postgreSQL/oms/.

Multi-Tenancy Configuration

Order Capture System requires a separate instance for each tenant. By default, Order Capture System integrates with Order Management Server using the default tenant. To use any other tenant than the default tenant, the following properties in `ConfigValues_OCS.xml` must be configured to provide the tenant ID and password:

- ```
<ConfValue description="Order Service security username" name="Order Service
security username" propname="com.tibco.af.ocs.orderService.username"
sinceVersion="3.0" visibility="Advanced">
 <ConfString default="admin" value="admin"/>
</ConfValue>>
```
- ```
<ConfValue description="Order Service security password" name="Order Service
security password" propname="com.tibco.af.ocs.orderService.password"
sinceVersion="3.0" visibility="Advanced">
```

- ```
<ConfString default="admin" isPassword="true" value="admin"/>
</ConfValue>
```
- ```
<ConfValue description="OCS Pricing username" name="OCS Pricing username"
propname="com.tibco.af.ocs.pricingService.username" sinceVersion="3.0"
visibility="Advanced">
  <ConfString default="admin" value="admin"/>
</ConfValue>

<ConfValue description="OCS Pricing security password" name="OCS Pricing security
password" propname="com.tibco.af.ocs.pricingService.password" sinceVersion="3.0"
visibility="Advanced">
  <ConfString default="admin" isPassword="true" value="admin"/>
</ConfValue>
```
 - ```
<ConfValue description="OCS Eligibility username" name="OCS Eligibility username"
propname="com.tibco.af.ocs.eligibilityService.username" sinceVersion="3.0"
visibility="Advanced">
 <ConfString default="admin" value="admin"/>
</ConfValue>

<ConfValue description="OCS Eligibility security password" name="OCS Eligibility
security password" propname="com.tibco.af.ocs.eligibilityService.password"
sinceVersion="3.0" visibility="Advanced">
 <ConfString default="admin" isPassword="true" value="admin"/>
</ConfValue>
```
  - ```
<ConfValue description="OMS UI REST Security username" name="OMS UI REST Security
username" propname="com.tibco.af.ocs.oauthTokenService.username" sinceVersion="3.0"
visibility="Advanced">
  <ConfString default="admin" value="admin"/>
</ConfValue>

<ConfValue description="OMS UI REST Security password" name="OMS UI REST Security
password" propname="com.tibco.af.ocs.oauthTokenService.password" sinceVersion="3.0"
visibility="Advanced">
  <ConfString default="admin" isPassword="true" value="admin"/>
</ConfValue>
```

The value for the above properties must be in the format of username@tenantid (for example, admin@tenant1).

Security Configuration

Order Capture System implements OAuth to provide server-side security. These configurations are used to modify the security settings.

| Name | Description | Default Value |
|--|---|---------------|
| com.tibco.af.ocs.noonce.validity.seconds | Time in seconds for which the OAuth nonce tokens generated by client are kept in-memory. This interval helps optimize nonce storage and must not be more than the session timeout interval. | 900 |
| com.tibco.af.ocs.session.timeout.seconds | Time in seconds for which the HTTP session is valid | 900 |

Catalog Directory Configuration

These configurations are used to tell Order Capture System where the catalog files from TIBCO Product and Service Catalog are available. At startup, Order Capture System loads data from the given locations.



The directories for segment and products must be configured. If these do not exist on the machine Order Capture System is running on, Order Capture System does not start at all.

| Name | Description | Default Value |
|---|--|-----------------------------------|
| <code>com.tibco.af.ocs.pmcatalog.dir</code> | Directory location for the product model catalog. Order Capture System needs the product model data exported from TIBCO Product and Service Catalog. This is an absolute path. | <code>/ocs/catalog/product</code> |
| <code>com.tibco.af.ocs.smcatalog.dir</code> | Directory location for the segment model catalog. Order Capture System needs the segment model data exported from TIBCO Product and Service Catalog. This is an absolute path. | <code>/ocs/catalog/segment</code> |

Access Points Common Configuration

This configuration is used so Order Capture System can identify where the web services within and outside of TIBCO Order Management - Long Running are hosted.

| Name | Description | Default Value |
|---|--|-------------------|
| <code>com.tibco.af.ocs.ap.protocol</code> | Order Capture System Access Points for the HTTP Protocol | <code>true</code> |

Log File Configuration

To have full logs during development or debugging, you can change INFO to ALL. This might slow down the system and is not a good practice for production.

Configure the log file: `$OM_HOME/roles/configurator/standalone/config/OCSLog4j.xml` if you are using the database mode of configuration. For the non-database mode, copy and edit the `$OM_HOME/roles/ocs/standalone/config/OCSLog4j.xml` file.

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

These 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.

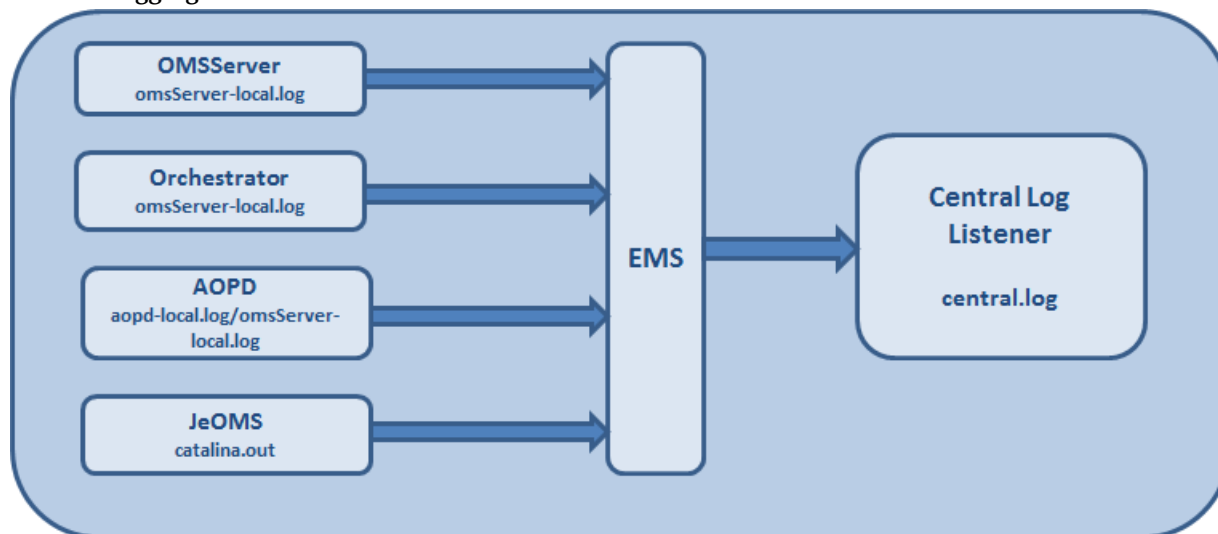
The individual components publish log messages to a central queue and then the central log server picks up the log messages from this queue.

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 components.
- Enables filtering of messages for effective analysis of the logs.

Central Logging



Contents of the Log Message

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

| Log Message Component | Description |
|-----------------------|--|
| Log level | One of the levels - DEBUG, INFO, WARN, ERROR, OFF. |
| BusinessTransactionId | Unique identifier for tracing purposes across function calls. |
| OrderRef | An Identifier to identify the Order for which this log message is written. |
| Component | Context information about the origin of the log (typically the engine name). |
| Service | Context information about the origin of the log (Typically the class name). |
| Operation | Context information about the origin of the log (Typically the method name). |
| StackTrace | Entire stack trace of the activity. |

| Log Message Component | Description |
|-----------------------|--|
| TimeStamp | Indicates when the message was logged. |

Controlling Log Levels

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

Configuring Logging for Java Components

Logging configuration for Java components applies to omsServer, Orchestrator, Automated Order Plan Development, Order Management Server UI, and Jeopardy Management System.

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

Order Management Server and Orchestrator: \$OM_HOME/roles/configurator/standalone/config/OMSServerLog4j.xml is used to configure logging in database mode. For the non-database mode configure \$OM_HOME/roles/omsServer/standalone/config/OMSServerLog4j.xml.

- The default logLevel is:
 - INFO for com.tibco.aff package and its sub package
 - 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 go into the logs folder for each service, the same as for the other components.

Order Management Server UI: \$OM_HOME/roles/configurator/standalone/config/OMSUILog4j.xml is used to configure logging in database mode. For the non-database mode configure \$OM_HOME/roles/omsui/standalone/config/OMSUILog4j.xml.

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

Jeopardy Management System: \$OM_HOME/roles/configurator/standalone/config/OMSServerLog4j.xml is used to configure logging in database mode. For the non-database mode configure \$OM_HOME/roles/omsServer/standalone/config/OMSServerLog4j.xml.

- The default logLevel is:
 - INFO for com.tibco.aff.jeoms and com.tibco.aff.eca.engine packages and their sub packages
 - 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 Jeopardy Management System. The console logs go under the logs folder for each service, the same as for the other components.

Automated Order Plan Development: \$OM_HOME/roles/configurator/standalone/config/AOPDLog4j.xml is used to configure logging in database mode. For the non-database mode configure \$OM_HOME/roles/aopd/standalone/config/AOPDLog4j.xml.

- The default logLevel is:
 - DEBUG for com.tibco.aff package and its sub packages
 - 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 Automated Order Plan Development in case of standalone deployment is aopd-local.log and in case of collocated deployment mode is omsServer-local.log. Published logs go into the logs folder for each service, the same as for the other components.

Administration Tasks

This section covers all the administration tasks for TIBCO Order Management - Long Running.

Docker

You can containerize TIBCO Order Management - Long Running components and run them on hosts that support the Docker environment. The Docker files are delivered as part of the TIBCO Order Management - Long Running installer. You can build images using those Docker files and then run them as containers.

This feature of TIBCO Order Management - Long Running requires Docker version 1.13 (or later) and Docker-Compose version 1.10 (or later).

It is required to have an internet connection on the machine where you install and run Docker.

You must have an Internet connection to download the base Docker image. For more information, see [Building a Docker Image Without an Internet Connection](#).



The term *Docker Context* refers to the directory where the Docker file is available. For example, Docker context for Orchestrator is `$OM_HOME/docker/oms/5.0.0-LR`.

Depending on your system configuration, you might need the following Docker containers:

- Order Management Server Container
- Order Management Server UI Container
- Automated Order Plan Development Container
- Configurator Container
- Order Capture System Container

After installation, all Docker related files are located in the `$OM_HOME/docker` directory when the user's PWD is `$OM_HOME/docker`.

Before you start using the Docker feature in TIBCO Order Management - Long Running, you must be familiar with the following Docker concepts:

- Docker architecture
- Using Docker in production
- Using Docker Volumes
- Docker commands
- Docker-Compose
- Using Docker-Compose in production

Building a Docker Image Without an Internet Connection

Download and install `wget` and `unzip` utilities to build a Docker image without an internet connection. In `$OM_HOME/docker/base/1.0/Dockerfile`, the "FROM CentOS" instruction initializes a new build stage and sets the base image for subsequent instructions. You can accept the default base image, which is the CentOS image from the Docker's public repository, or you can change the instruction and provide a valid source for a different base image. You can pull a valid base image from the Docker's public repository or you can create your base image, push it to a public or private Docker registry, and then use the newly created image as a base image. For more information about creating your base Docker image, see the [Docker documentation](#) related to Creating a Base Image.

In `$OM_HOME/docker/base/1.0/Dockerfile`, you can find instructions for downloading `wget` and `unzip` utilities. These instructions can be modified to pick up the installers of the utilities from the Docker context and install them in the image.



In this case, the Docker context for \$OM_HOME/docker/base/1.0/Dockerfile is \$OM_HOME/docker/base/1.0.

Perform the following steps to download the wget and unzip utilities:

Procedure

1. Download and install the wget and unzip utilities.
2. Copy the installer (.rpm) files for the wget and unzip utilities to the Docker context directory.
3. Modify \$OM_HOME/docker/base/1.0/Dockerfile to include the instruction to copy from the Docker context to run the installers:
 - a) Add a **COPY** command for both the utility installers.
 - b) Change the command **yum -y install wget unzip** to **yum localinstall -y wget*.rpm unzip*.rpm**.

Copying Files to Docker Context

It is required to copy files to the Docker context before building the required Docker images.

Procedure

1. Install Docker 1.13 (or later), Docker-Compose 1.10 (or later), and TIBCO Order Management - Long Running on the host machine.
2. Copy the JRE 11 64-bit RPM file, for example `jre-11*-linux-x64.rpm`, to the \$OM_HOME/docker/base/1.0 directory.
3. Run the \$OM_HOME/roles/copyLib.sh script. This script copies `ojdbc7.jar`, `jms-2.0.jar`, `tibjms.jar` and other required .jar files. For more information, refer to the "Post-Installation Task: Copying Dependencies" topic in the *TIBCO Order Management - Long Running and Configuration* guide.
4. Go to the \$OM_HOME/docker directory and execute `copy-required-files.sh`. This shell script copies all the required directories from OM_HOME to a specific Docker context. These files are required to build Docker images. This script works on OM_HOME set in the environment or takes the value of OM_HOME as user input.

Building Docker Images

After the required files are copied at the Docker context, you can build the Docker images. Use Docker-Compose for building the Docker images.

Procedure

1. Go to the \$OM_HOME/docker directory and execute the following command:


```
docker-compose --file docker-compose-build-complete.yml build
```
2. Execute the following command to check the images that are created:

```
$] docker images
```

You might get the following output:

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|--------------------|----------|--------------|--------------|--------|
| tibco/ocs-util | 5.0.0-LR | 1a27d000a07e | 5 weeks ago | 898MB |
| tibco/ocs | 5.0.0-LR | a73f62977959 | 5 weeks ago | 1.09GB |
| tibco/aopd | 5.0.0-LR | bad7075c78c3 | 6 weeks ago | 1.15GB |
| tibco/omsui | 5.0.0-LR | 8f13d5107ed5 | 6 weeks ago | 1.33GB |
| tibco/oms | 5.0.0-LR | 886cbc34af15 | 6 weeks ago | 1.19GB |
| tibco/configurator | 5.0.0-LR | 523371e6360a | 6 weeks ago | 1.04GB |
| tibco/base | 1.0 | 45c6f2ff14c3 | 22 hours ago | 630 MB |
| centos | latest | 67591570dd29 | 3 months ago | 192 M |

ocs-util Docker Image

You must load the models into the system to configure Order Capture System applications.

The `ocs-util` Docker image is used to load the required models onto a specific volume. After starting the utility application, the models can be copied onto the volume associated with the container. Once the models are loaded, the same volume is mounted on the Order Capture System container for processing.

Preparing Docker Volumes

A Docker volume (commonly referred as data volume) is a specially-designated directory within one or more containers that bypasses the Union File System. Docker volumes provide several useful features for persistent and shared data. Docker volumes can be shared and reused among containers. Docker volumes persist even if the container itself is deleted. You can create a Docker volume container from an existing image. The volume container has the mapping of the host's directory to the container. For further information, refer to the Docker documentation on volume.

Before you start running the Docker containers, you are required to have the following directories on the host machine:

- The `$OM_HOME/roles/configurator/standalone/config` directory. This directory contains all the configuration files required by the Configurator server; this directory is mapped as a data volume inside the `fom-configurator` Docker containers.



All the other containers refer to the database to access any of the property files.

- The model loading directories. These are the directories setup to load the product and planFragment, action. These models are loaded by the Order Management Server and no other container needs this volume mapping other than the Order Management Server container.
- A logs directory that you must create.

Procedure

- Create a logs directory where logs from all the containers are available on your host machine. For example, `$OM_HOME/logs`.



Since the Docker container reads and writes in the three directories mentioned, it is mandatory to give read and write privilege to others on the host machine on the directories where you are creating volumes. Since Docker is writing to the existing files in the volume, each file in the volume must have read and write privileges. For example:

```
$] chmod o+rw -R
$OM_HOME/roles/configurator/standalone/config
```

Setting Up the .env File

Set the required variables, which varies according to the user's environment. All of these variables are required to be changed according to the user's environment.

All of these variables are in the `.env` file located in the `$OM_HOME/docker` directory.

Procedure

1. Set the port number for which you want to start the TIBCO Order Management - Long Running Configurator. When you want to access the Configurator UI, you use this port.

```
# Port exposed for fom-configurator on docker host machine
#
HOST_CONFIGURATOR_PORT=8083
```

2. Set the port number for which you want to start the Order Management Server UI. When you want to access the Order Management Server UI, you use this port.

```
# Port exposed for oms ui on docker host machine
#
HOST_OMS_UI_PORT=8081
```

3. Set the host directory path to mount a volume from the host machine to the Docker container for the TIBCO Order Management - Long Running config directory.

```
# Path till fom's config directory, this directory will
# be mapped as a data volume inside all the docker containers
#
HOST_CONFIG_DIR_PATH=<path-to-fom-config-dir-on-host>
```

4. Set the path for the Order Management Server product model, Order Management Server product model success, and Order Management Server product model failure directory on the host machine.

```
# Path till product model, product-success and product-failure
# directory on host machine, this directory will be used by
# oms at startup time and this directory will be mapped as
# a data volume inside oms docker containers
#
HOST_OMS_PRODUCT_MODEL_DIR_PATH=<path-to-oms-product-dir-on-host>
HOST_OMS_PRODUCT_MODEL_SUCCESS_DIR_PATH=<path-to-oms-product-success-dir-on-host>
HOST_OMS_PRODUCT_MODEL_FAILURE_DIR_PATH=<path-to-oms-product-failure-dir-on-host>
```

5. Set the path for the Order Management Server plan fragment model, Order Management Server plan fragment success, and Order Management Server plan fragment failure directory on the host machine.

```
# Path till planfragment model, planfragment-success and
# planfragment-failure directory on host machine, this
# directory will be used by oms at startup time and this
# directory will be mapped as a data volume inside oms
# docker containers
#
HOST_OMS_PLANFRAGMENT_MODEL_DIR_PATH=<path-to-oms-planfragment-dir-on-host>
HOST_OMS_PLANFRAGMENT_MODEL_SUCCESS_DIR_PATH=<path-to-oms-planfragment-success-dir-on-host>
HOST_OMS_PLANFRAGMENT_MODEL_FAILURE_DIR_PATH=<path-to-oms-planfragment-failure-dir-on-host>
```

6. Set the path for the Order Management Server action model, Order Management Server action success, and Order Management Server action failure directory on the host machine.

```
# Path till action model, action-success and action-failure
# directory on host machine, this directory will be used by
# oms at startup time and this directory will be mapped as
# a data volume inside oms docker containers
#
HOST_OMS_ACTION_MODEL_DIR_PATH=<path-to-oms-action-dir-on-host>
HOST_OMS_ACTION_MODEL_SUCCESS_DIR_PATH=<path-to-oms-action-success-dir-on-host>
HOST_OMS_ACTION_MODEL_FAILURE_DIR_PATH=<path-to-oms-action-failure-dir-on-host>
```

7. Set the cluster member ID for the Order Capture System server.

```
# Cluster member id for ocs server
#
OCS_NODE_ID=Member1
```

8. Set the path for the Order Capture System segment model directory on the host machine.

```
# Path till segment model directory on host machine,
# this directory will be used by ocs at startup time and
# this directory will be mapped as a data volume
# inside ocs docker containers
#
HOST_OCS_SEGMENT_MODEL_DIR_PATH=<path-to-ocs-segment-dir-on-host>
```

9. Set the path for the Order Capture System product model directory on the host machine.

```
# Path till product model directory on host machine,
# this directory will be used by ocs at startup time and
# this directory will be mapped as a data volume
# inside ocs docker containers
#
HOST_OCS_PRODUCT_MODEL_DIR_PATH=<path-to-ocs-product-dir-on-host>
```

10. Set the path for the log file created in [Preparing Docker Volumes](#).

```
#Location of Logfile
HOST_LOG_ROOT_LOCATION_DIR_PATH=<path-to-logfile>
```



When you are loading models through the containerized Order Management Server service, you must not make any changes to the paths of "Offline Catalog Configuration" in the `ConfigValues_OMS.xml` file.

Configuring for Order Management Server Docker Containers

When you run the Order Management Server or the Order Management Server UI as Docker containers, you have to make configuration changes using the Configurator UI and in the `ConfigValues_OMS.xml` file.

Procedure

1. Start TIBCO Order Management - Long Running Configurator as a Docker container by running the following command:

```
$] docker-compose --file docker-compose-run-configurator.yml up -d
```

You can access the Configurator UI on the port, which you had set for the `HOST_CONFIGURATOR_PORT` variable in the `.env` file.

2. On the TIBCO Order Management - Long Running Configurator UI, make the configuration changes according to your environment. For example, make related configuration for the database, TIBCO EMS, Order Management Server, Automated Order Plan Development, Order Capture System, Jeopardy Management System, and other related configurations.

All the changes that you do in the Configurator UI are reflected in the host machine's directory, and the changes are uploaded to the database so other containers can read from it.

3. When you are going to use the Order Management Server UI as a Docker container, configure the value of the Order Management Server UI port through TIBCO Order Management - Long Running Configurator:

```
name="OMS UI HTTP Port Number" propname="com.tibco.af.omsui.http.port"
```

The property value for the Order Management Server UI port is the same as the value of the `HOST_OMS_UI_PORT` variable in the `.env` file.

4. When you are going to use the Order Management Server UI as a Docker container, configure the Order Management Server address and port number in the Configuration UI:

- Server IP: description="Host address of the OMS server" name="OMS Server Host Name" propname="com.tibco.af.omsServer.proxyHost"
- Port number: description="Port number of the OMS Server" name="OMS Server Port Number" propname="com.tibco.af.omsServer.proxyPort"

This port is used by the Order Management Server UI to send the user requests to the Order Management Server container.

5. Do the required setup related changes in the `profiles.properties` file located in the `$OM_HOME/roles/omsServer/standalone/config` directory. You can enable or disable different components. For example, Jeopardy Management System can be disabled. For some components you specify if you want to use that specific component co-located or standalone, for example, Automated Order Plan Development, and Orchestrator.

Running the Docker Containers

After building the Docker images, you can run the images as containers to start containerized TIBCO Order Management - Long Running.

Based on the environment and setup you need, you can at-max start the following images as a Docker containers:


- Automated Order Plan Development image used to start as a Docker container: `$OM_HOME/docker/aopd/5.0.0-LR`
- TIBCO Order Management - Long Running Configurator image used to start as a Docker container: `$OM_HOME/docker/configurator/5.0.0-LR`
- Order Capture System image used to start as a Docker container: `$OM_HOME/docker/ocs/5.0.0-LR`
- Order Management Server image used to start as a Docker container: `$OM_HOME/docker/oms/5.0.0-LR`
- Order Management Server UI image used to start as a Docker container: `$OM_HOME/docker/omsui/5.0.0-LR`

Running Different Containers for TIBCO Order Management - Long Running Components

Prerequisites

Before you start the Order Management Server as a Docker container, you must complete all the steps in [Configuring for Order Management Server Docker Containers](#).

Procedure

1.  Ignore this step if you are going to use Automated Order Plan Development in co-located mode.

Start `tibco/aopd:5.0.0-LR` as a container using the following command:

```
$] docker-compose --file docker-compose-run-aopd.yml up -d
```

- a) Check that the container has started using the following command:

```
$] docker ps -a
```

This command gives all the information about the started container. You can also see the port on which Automated Order Plan Development has started as a container. Use this port to configure Automated Order Plan Development configurations for Plan-Preview through Configurator UI.

2. Start `tibco/oms:5.0.0-LR` as a container using the following command:

```
$] docker-compose --file docker-compose-run-oms.yml up -d
```

- a) Check the details for the Order Management Server container by executing the following command:

```
$] docker ps -a
```

The output also gives you the port number, which you must use for accessing the Order Management Server.

- b) Configure the server information (address and port) using the Configurator UI. These properties are used by Automated Order Plan Development UI at the startup time, so after making these changes, start the Automated Order Plan Development UI container.

3. Start the `tibco/omsui:5.0.0-LR` as a container using the following command:

```
$] docker-compose --file docker-compose-run-omsui.yml up -d
```

- a) Check the details for the Automated Order Plan Development UI container by executing the following command:

```
$] docker ps -a
```

You can access the Automated Order Plan Development UI on the port, which you had set for the `HOST_OMS_UI_PORT` variable in the `.env` file.

Enabling the RMI Port for the Order Management Server Container

Complete the following steps to enable the RMI port for the Order Management Server Docker container.

Prerequisites

You must have already executed `$OM_HOME/docker/copy-required-files.sh`. The following steps talk about the files kept at that specific docker context.

Procedure

1. Open the `$OM_HOME/docker/oms/5.0.0-LR/omsServer/standalone/bin/start.sh` file in edit mode and create the following new variable `JMX_RMI_ARGS` in the script:

```
JMX_RMI_ARGS="-Dcom.sun.management.jmxremote
-Djava.rmi.server.hostname=<host-ip-address>
-Dcom.sun.management.jmxremote.port=<port-number>
-Dcom.sun.management.jmxremote.rmi.port=<port-number>
-Dcom.sun.management.jmxremote.local.only=false
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false"
```

`<host-ip-address>` must be updated with the IP address of your host machine and `<port-number>` with the port number where you want to access the RMI services exposed by Order Management Server.

2. Use the created variable `JMX_RMI_ARGS` and add it to the line where the application invokes Java and passes the required arguments. For example:

```
java $JMX_RMI_ARGS -cp
$CLASS_PATH $JVM_OPTIONS $CLASS_NAME ${ARGUMENTS[@]}
```

In case the port number you are using is behind the firewall, make sure to make the exception for this port number. In case you have IP tables, make a new rule in IP tables for this port.

3. Expose the port number you have mentioned in [step 1](#) in the Order Management Server Docker file `$OM_HOME/docker/oms/5.0.0-LR/Dockerfile`.
4. In `$OM_HOME/docker/docker-compose-run-oms.yml` map the port, which you have exposed for JMX under ports (maintain the white-spaces). For more information on how to map the port, refer to [this step](#) in the "Running the Order Management Server Container on a Predefined Port" topic.
5. Build the oms docker image as mentioned in [Building Docker Images](#).

Running the Order Management Server Container on a Predefined Port

Since Order Management Server UI requires an address and port information of Order Management Server and because you start the Order Management Server container with no default hard-coded port, in some cases, it might become difficult to update the server information repetitively through the Configuration UI. Due to the dynamic port assignment, Order Management Server ports might change whenever the server is restarted. To avoid this issue, you can start one of the members of the Order Management Server with a predefined port and use the same port to configure through the Configurator UI.

Procedure

- You can start Order Management Server on a predefined port with either of the following two ways:
 - You can make changes in Docker compose file.

Open the `$OM_HOME/docker/docker-compose-run-oms.yml` file with a suitable editor and locate the section with the port defined as 9091 and change it in the following way: `<user-specified-port-number>:9091`. Here `<user-specified-port-number>` is the port number, which you want Order Management Server/service to start on. Save the YAML file and now execute the YAML to start the Order Management Server container as mentioned in [Running the Docker Containers](#).

- You can skip Docker-Compose commands and use Docker commands directly to start the Order Management Server container on a specific port. Please refer to some of the examples that are mentioned in the `$OM_HOME/docker/ReadMe.txt` file.

After starting the Order Management Server container on a specific port and configuring its address and port in the Configurator, all of Order Management Server UI's requests are routed through this new Order Management Server.

Scaling the Docker Container

You can scale your Docker containers with Docker-Compose.

Start the Docker containers without specifying the port. In case you specify the port, then you are not able to scale the Docker container using the Docker-Compose's scale command. Docker compose's scale command provides flexibility to start desired containers with the same configuration by just specifying a number. So when you are scaling the Docker container, you are letting the Docker daemon assign a free port for the container you are starting.



When scaling any service, it is mandatory to [Run the OMS Server Container on a Predefined Port](#).

Procedure

1. Change PWD to `$OM_HOME/docker`.
2. Use the following command to scale the Order Management Server container and start, for example, three instances of the Order Management Server:

```
$] docker-compose --file docker-compose-run-oms.yml scale tibco-fom-oms=3
```

In the above command, you have specified the service name `tibco-fom-oms` defined in `docker-compose-run-oms.yml` and are scaling it three times. In a similar way, you can scale the service for Automated Order Plan Development.

3. After scaling the Docker container, check the container details by executing the following command:

```
$] docker ps -a
```

Extend Docker-Compose Files

You can extend the Docker-Compose files provided as part of the TIBCO Order Management - Long Running installation.

This is mainly done to handle different environments. For example, this is done in case you required a separate parameters for the containers based on your environment, such as a testing or production environment.

The suggested way to do this is to have multiple Docker compose files for each environment. For more information, see the [Docker documentation](#) on Multiple Compose Files.

Adding Environment Variables

There can be some scenarios where you want to add environment variables to a container on startup. This can be done in Docker-Compose files, which you are using to start the containers.

You can add a `NODE_ID` environment variable at startup of an Order Management Server docker container. After the NodeFinder library is added to Order Management Server, you are not required to specify the `NODE_ID` and `DOMAIN_ID` environment variables.

1. In case you want to specify these environment variables on a container startup, add environment variables to the `$OM_HOME/docker/docker-compose-run-oms.yml` file:

```
"NODE_ID=Member1"
```

```
"DOMAIN_ID=ORCH-DOMAIN"
```

The following code snippet looks like the modified part of the `$OM_HOME/docker/docker-compose-run-oms.yml` file:

```
version: "3"

services:
  tibco-oms:
    image: "tibco/oms:${FOM_VERSION_TAG}"
    ports:
      - "9091"
    volumes:
      - "${HOST_LOG_ROOT_LOCATION_DIR_PATH}:/home/tibuser/tibco/om-lr/5.0/omsServer/standalone/logs"
```

Modifying a Container Time-Zone

The default time-zone for any Docker container is UTC. In the case where you want the Docker container's time-zone to be in sync with the host machine's time-zone, you can apply these changes either in the Docker file or in the Docker-Compose YAML file.

Docker containers always use the system clock of the host machine but it sets its time-zone as UTC.

The following steps are an example of changing time zone for an Order Management Server container.

Procedure

- You can modify a container's time zone with either of the following two ways:
 - This approach can be applied when you have not created any images. Open the `$OM_HOME/docker/oms/5.0.0-LR/Dockerfile` in a suitable editor and modify the file as shown:

```
FROM tibco/base:1.0

COPY omsServer $OM_HOME/omsServer

RUN chown -R tibuser:tibuser /home/tibuser/ \
&& chmod -R 755 /home/tibuser \
&& chmod a+w $OM_HOME/omsServer/standalone/logs \
&& chmod a+w $OM_HOME/omsServer/standalone/config

USER tibuser

ENTRYPOINT ["/home/tibuser/tibco/om-lr/5.0/omsServer/standalone/bin/start.sh",
"--run=FG"]

EXPOSE 9091
```

In this example, the following has been modified:

```
ENV TZ=Asia/Kolkata
```

```
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ /etc/timezone
```

Here you have to change the value of the TZ variable as per your time zone (in the example, the time zone is Asia/Kolkata).

- This approach can be applied if your images are already created and now you want to change the container time zone at runtime. Open `$OM_HOME/docker/docker-compose-run-oms.yml` in a suitable editor and modify the file as shown:

```
version: "3"

services:
  tibco-fom-oms:
    image: "tibco/fom-oms:${FAF_VERSION_TAG}"
    ports:
      - "9091"
    volumes:
      - "${HOST_LOG_ROOT_LOCATION_DIR_PATH}:/home/tibuser/tibco/af/5.0/
```



```
omsServer/standalone/logs"
  environment:
    - "TZ=Asia/Kolkata"
  command: >
    sh -c "ln -snf /user/share/zoneinfo/$TZ /etc/localtime &&
    echo $TZ > /etc/timezone"
```

In this example, the following has been modified:

```
environment:
- "TZ=Asia/Kolkata"

command: >
  sh -c "ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/
  timezone"
```

Here you have to change the value of the TZ variable as per your time zone (in the example, the time zone is Asia/Kolkata).

Reading Container Logs

When all the desired containers are up and running, it is best practice to check the logs for all the running services.

Logs for all the started and exited containers are available at the path you have mentioned for the HOST_LOG_ROOT_LOCATION_DIR_PATH variable in the .env file. So all the logs are preserved on your host machine.

Procedure

- When you are starting multiple containers of the same service, for example, when you are scaling the Order Management Server or Automated Order Plan Development service, use the system generated prefix `${sys:uuid}` to `fileName` and `filePattern` properties of the `RollingFile` element of the Log4j XML file of the service you are scaling.

For example, when you are trying to scale the Order Management Server service, open `OMSServerLog4j.xml` and locate the element `RollingFile` and add the prefix `${sys:uuid}` to the `fileName` and `filePattern` properties.

The changed properties look like this:

```
<RollingFile name="LocalLogFileAppender"
fileName="logs/${sys:uuid}omsServer-local.log"
filePattern="logs/${sys:uuid}omsServer-local-%i.log">
```

This ensures that scaling multiple times does not overwrite the log files as the file names are appended with a randomly generated text. The Docker daemon also logs all the activities of the container right from the moment you start it. For more information, see the Docker documentation on Docker logs.

Troubleshooting Error from Building Docker Images

Troubleshoot an error when building Docker images with the following steps.

1. Complete the following steps if the following error occurs when building Docker images:

```
rm: cannot remove '/home/tibuser/tibco/om-lr/5.0/configurator/standalone/config/
backup': Directory not empty
```

Procedure

1. Run the following command on the host machine:

```
$] docker info | grep 'Storage Driver' | awk -F':' '{print $2}'
overlay
$]
```

2. If the output is overlay, then apply the following workaround:
 - a) Stop the Docker engine.
 - b) Changed DOCKER_OPTS to set storage-driver value to device mapper, edit `/etc/docker/daemon.json`, and add "storage-driver" : "devicemapper" at the end of existing keys.
 - c) Start the Docker engine.



You can lose the existing Docker images due to the above change.

- d) Verify the fix by running the following command:

```
$] docker info | grep 'Storage Driver' | awk -F':' '{print $2}'
devicemapper
$]
```

Order Sequencing

By default Order Sequencing feature is disabled. When Order Sequencing is disabled, all the incoming orders are processed in parallel. After the Order Sequencing is enabled, only a single order is processed at a time and any other incoming order by the same customer is stored in a queue (`tibco.aff.oms.ordersSequencer.submitOrder`) till the previous one is processed.

The order of a customer is processed in the sequence of order submission. Each order request (SOAP request) has an element or a tag (called as custom property) in the request body, which is common into all the requests for a customer. Orders from one customer are processed in the sequence of order submission.

The following configuration properties are related to Order Sequencing:

| Configuration Variable Name | Configuration Property Value | Description |
|---|--|---|
| Submit Order Sequencer Queue | <code>tibco.aff.oms.ordersSequencer.submitOrder</code> | Queue where an order is sent when other order is still in process |
| Order Sequencer Order Status Notification Queue | <code>tibco.aff.oms.ordersSequencer.notification.order</code> | Status notification for each order sequencer that are in queue |
| Order Sequencer Listener Trigger Status | <code>tibco.aff.oms.ordersSequencer.trigger.status</code> | Status of the order sequencer listener, indicating whether the order has been completed |
| Submit Order Sequencer Queue Receive TimeOut | <code>tibco.aff.oms.ordersSequencer.submitOrder.receive.timeout</code> | The order remains in sequencer queue for a specific time (millisesonds) and then it is suspended. |
| Custom property xPath expression for order sequencing | <code>tibco.aff.oms.ordersSequencer.customer.xPath</code> | Custom property xPath for order sequencing that points to a unique customer identifier |
| Flag to enable or disable the Order Sequencing | <code>tibco.aff.oms.ordersSequencer.enableOrderSequencing</code> | Options to enable (for all or with a udf) or disable order sequencing |

Enabling or Disabling Order Sequencing

Procedure

1. Define an XPath in `tibco.aff.oms.ordersSequencer.customer.xpath`, which points to a unique customer identifier in the SOAP request.
2. Set the below are the enum values for `tibco.aff.oms.ordersSequencer.enableOrderSequencing`:
 - `Disable`: To disable order sequencing for all the orders.
 - `EnableForAll`: To enable order sequencing for all the orders.
 - `EnableWithUdf`: For the selected user-defined fields for which you want to enable order sequencing.
3. In TIBCO Enterprise Message Service, create the following queues:
 - `tibco.aff.oms.ordersSequencer.submitOrder`
 - `tibco.aff.oms.ordersSequencer.notification.order`
4. Create a bridge with the following source and target configurations:
 - `source=tibco.aff.orchestrator.notification.order`
 - `target=tibco.aff.oms.ordersSequencer.notification.order`

Database Partitioning

TIBCO Order Management - Long Running maintains a large number of orders, models, plan fragments, and other business related data. To manage this growing data, the database is partitioned into logically independent units. The TIBCO Order Management - Long Running database supports two databases out of box: Oracle and PostgreSQL.

The database is partitioned using the `PARTITIONDATE` column. Implementation of the `PARTITIONDATE` column differs for Oracle and PostgreSQL databases.

For both database back-ends, scripts are provided to complete the following tasks:

- [Install a fresh copy of the database \(new users\)](#)
- [Add future partitions to the existing database \(existing users\)](#)
- [Truncate old partitions \(new and existing Oracle database users\)](#)
- [Drop old partitions \(Postgres database users\)](#)

Installing a Fresh Copy of a Database

Run the respective script to install a fresh copy of an Oracle or PostgreSQL database if you are a new user.

- **Oracle EE users:**

Run the `OMS_DDL.sql` file by running the file manually on the SQL prompt. This script automatically creates a new partitioned database.

- **Postgres users:**

With a Postgres database, only a partitioned database is supported. To create a partitioned PostgreSQL database instance, run the `setup.sh` file under `$OM_HOME/db/postgreSQL/oms`.

Adding Future Partitions to an Existing Database

The Oracle SE database does not support partitioning, this section assumes that the user has an Oracle EE or Postgres database.

- **Oracle EE users:**

Future partitions for an Oracle database are created automatically on the first of every month. For existing users, the create new partitioning procedures might already be in place.

For users creating a new instance of the database, run the following two scripts under \$OM_HOME/samples/db_scripts/oracle after the initial tablespace and user are setup:

- OMS_Create_Future_Partitions_SQL_Monthly.sql
- OMS_Future_Partitions_Monthly_JOB.sql

The OMS_Create_Future_Partitions_SQL_Monthly.sql script saves a procedure called ORS_FUT_PART_WEEKLY in the database. The OMS_Future_Partitions_Monthly_JOB.sql script creates a job that kicks off at 1:00 a.m. on the first of every month. This job calls the ORS_FUT_PART_WEEKLY procedure to create a one month worth of future partitions. Since partitions are divided weekly, this procedure creates five partitions for the 1st, 7th, 14th, 21st, and 28th day of every month.

- **Postgres users:**

For Postgres users, no additional steps are needed to create future partitions. The database creates a new partition automatically if one does not exist in the database before saving relevant information in that partition.

Truncating Old Partitions

Truncating old partitions is only supported for Oracle databases.

- **Oracle EE users:**

A truncate partition script is provided for an Oracle EE database. Truncate partitioning lets 24/7 up time of the database when partitions are being truncated and indexes are being updated. Refer to the Partition_Truncate_Readme.txt file under \$OM_HOME/db/oracle/purge for additional details on the setup and run procedure.



Even though partitions can be truncated when the database is serving OLTP requests, it is highly a good practice to run the truncation script when either the database is under a minimal load or disconnected from the front end for other maintenance.

Dropping Old Partitions

Dropping old partitions is only supported for a Postgres database.

- **Postgres Users:**

For a Postgres database, partitions can be dropped by running the drop_partition_by_name.sql file.

See the Partition_Drop_Readme.txt file under \$OM_HOME/db/postgreSQL/purge for additional details on the setup and run procedure.



Dropping a partition in a PostgreSQL database is allowed only when all of the orders in that partition are either COMPLETE or CANCELLED. If there are any open orders in the partition, the drop partition process aborts.

Database Partitioning with an Oracle Database

The Oracle database is partitioned using reference partitioning. A hierarchy tree of tables is created for existing TIBCO Order Management - Long Running tables using a foreign key relationship between tables. The table hierarchy is maintained using the ORDERS table as a primary (or a parent) table for all tables.

A column PARTITIONDATE in the ORDERS table is used for partitioning. The column PARTITIONDATE maintains the same date as SUBMITTEDDATE, and this column is populated when a new order is submitted in the system. The ORDERS table is partitioned weekly based on the PARTITIONDATE column.

The following are the advantages of a partitioned database:

- Data is logically separated

With partitioning in place, the database administrator has the option to keep each partition in its tablespace resulting in better space management and improving the total cost of ownership.

- Purge performance is improved

You can now truncate a partition instead of manually deleting entries from the ORDERS and other related tables for purging. Since the partition truncate operation is performed at a data dictionary level, this operation is very quick compared to manually deleting the rows.

TIBCO Order Management - Long Running comes with the ORDERS table in a weekly partitioned format based on the PARTITIONDATE column. The PARTITIONDATE column maintains a date when the order was submitted to the system. Each month is divided into five partitions for each week in the month.

The following is an example of weekly partitioning:

| Partition Number | Orders Submitted Between (Date and Time) |
|------------------|--|
| Partition 1 | FROM: 28th of the last month - 1 a.m.
TO: 1st of current month - 12:59 a.m. |
| Partition 2 | FROM: 1st of current month - 1 a.m.
TO: 7th of current month - 12:59 a.m. |
| Partition 3 | FROM: 7th of current month - 1 a.m.
TO: 14th of current month - 12:59 a.m. |
| Partition 4 | FROM: 14th of current month - 1 a.m.
TO: 21st of current month - 12:59 a.m. |
| Partition 5 | FROM: 21st of current month - 1 a.m.
TO: 28th of current month - 12:59 a.m. |

You can do the partition of the database monthly as well. Example scripts for partitioning monthly for the ORDERS table and related create future partitions can be found under the \$OM_HOME/samples/db_scripts/oracle folder.



The \$OM_HOME/db/oracle/oms/OMS_DDL.sql file has default partition entries until the date 12-28-2017:01:00:00 AM. If the default dates are about to be passed within a month or have already passed, you must manually change the date entries to the dates that you want to create and map the partitions.



Database partitioning does not require a maintenance window for the cleanup activity.

Database Partitioning with a Postgres Database

All order related tables in a Postgres database are partitioned individually.



Refer to the `Partition_Drop_Readme.txt` file under `$OM_HOME/db/postgreSQL/purge` for the list of partitioned tables.

A `PARTITIONDATE` column is added to each table that stores order related information. All these tables are partitioned monthly based on the `PARTITIONDATE` column.

The following are the advantages of a partitioned database:

- Data is logically separated

With partitioning in place, the database administrator has the option to keep each partition in its tablespace resulting in better space management and improving the total cost of ownership.

- Purge performance is improved

You can now drop a partition instead of manually deleting entries from the `ORDERS` and other related tables for purging. Since in Postgres each partition is represented by individual tables, the drop partition does not impact any other table or partition for the same table. This operation is also very quick compared to manually deleting the rows.

The following is an example of monthly partitioning:

| Partition Number | Orders Submitted Between (Date and Time) |
|------------------|--|
| Partition 1 | FROM: 1st of June - 1 a.m.
TO: 1st of July - 12:59 a.m. |
| Partition 2 | FROM: 1st of July - 1 a.m.
TO: 1st of August 12:59 a.m. |
| Partition 3 | FROM: 1st of September - 1 a.m.
TO: 1st of October - 12:59 a.m. |
| Partition 4 | FROM: 1st of October - 1AM
TO: 1st of November - 12:59 a.m. |
| Partition 5 | FROM: 1st of November - 1AM
TO: 1st of December - 12:59 a.m. |

Migrating From a Non-Partitioned AUDIT_TRAIL Table

A migration script is provided to migrate the TIBCO Order Management - Long Running `AUDIT_TRAIL` table from a non-partitioned state to a partitioned state.

The `Partition_Migration_Readme.txt` README file in `$OM_HOME/db/oracle/oms` specifies all the prerequisites and steps required to migrate the `AUDIT_TRAIL` table from a non-partitioned state to a partitioned state.

This migration is performed using the Oracle `DBMS_REDEFINITION` package without any downtime.

Although the migration script can run when the database is handling online transaction processing, it is a good practice to run this migration when there is no load or a minimal load on the database.

Any new grants required to operate are detailed in the README file.

Product Model Purge

The Purge web service makes it possible to remove existing products from model repository in Automated Order Plan Development and models persisted in the Order Management Server database.



When the flag `com.tibco.fom.oms.modelLoadingMaxIdle` is set to `true`, during models purging and model loading, Orchestrator does not process any incoming order events for orchestration. When the flag is set to `false`, Orchestrator processes the order when the model loading and model purging is in progress. For more information, see the [Model Processing](#) topic.

Purge Request

```
<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="?">
      <!--off:all>all</off:all-->
      <off:productId>Product_TV,PO_TV</off:productId>
    </off:PurgeRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Purge Response

```
<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/orderservice" xmlns:ns2="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.

- `$OM_HOME\schemas\wsdl\http\OfflineCatalogue.wsdl`

Purge Model Server Cache Impact

Writing all of the in-memory repository cache information in the `SERVER_CACHE` table is an input/output intensive operation. In a scenario where a single model is purged from the database also requires an almost identical task (with one less model) of all the in-memory information written to the database multiple times. To avoid this input/output intensive database write, in case of a model purge operation that does not involve purging all models, cache validity for the relevant cache in the `SERVER_CACHE` table is set to `INVALID` (value of 0). If all the models are purged, such as the products, the server cache information for the corresponding model type is completely removed and any subsequent model processing updates the `SERVER_CACHE` table.



Any model operations performed with invalid cache does not involve updating cache information in this table. This information is updated and cache validity is set to `VALID` (value of 1) with a subsequent application server restart as detailed in the "Repository Cache Rebuild" section in the [Model Processing](#) topic.

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 Order Management Service.

Bulk Order Actions

The bulk order actions let 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 Order Management Server order service. This operation is called **PerformBulkOrderAction**.

The existing Order Management Server order service is modified to include a new operation. You can use this operation 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 information about the status of the request.
- Order lists - show 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.

- \$OM_HOME\schemas\wsdl\http\OrderServiceHTTP.wsdl
- \$OM_HOME\schemas\wsdl\jms\OrderServiceJMS.wsdl

Error Codes

The following table lists the error codes:

| Error Code | Description |
|--------------------------------------|--|
| TIBCO-AFF-OMS-100046: INVALID_ACTION | Web service fault code for invalid values of action. |

| Error Code | Description |
|--|---|
| TIBCO-AFF-OMS-100047: NO ORDERS FOUND | Web service fault code when neither order id nor order reference is present in the request. |
| TIBCO-AFF-OMS-100048: BOTH ORDERID AND ORDERREF FOUND | Web service fault code when both order id and order reference are present in the request. |
| TIBCO-AFF-OMS-100020:
ORDER {ORDERREF} NOT FOUND / ORDER {ORDERID}
NOT FOUND | This exception is logged if an order to be canceled or withdrawn is not present in the Order Management Server component. |

Invoke 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 Order Management Server component. For tracking the status of individual orders, activity logs, or an order, use the browse windows of the TIBCO Order Management - Long Running 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 `Order Management Server` in Configurator must be set to `false` when invoking bulk cancel operation.

Tracking the Request Status

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

- TIBCO Order Management - Long Running UI (Dashboard, Order Screen and Activity logs)
- Logs in the Order Management Server and Orchestrator components

Logging

TIBCO Order Management - Long Running 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 `$OM_HOME/roles/omsServer/standalone/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 Order Management Server component, an 'ERROR' level log is generated indicating that the order was not found.

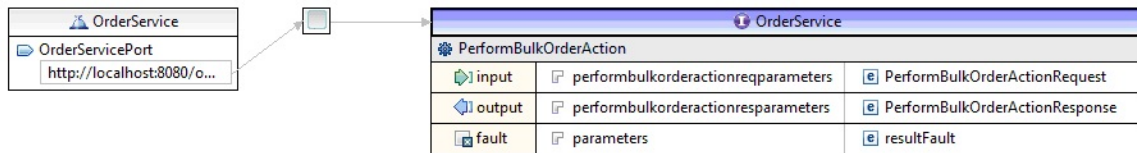
Schema

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 Order Management Server order service.

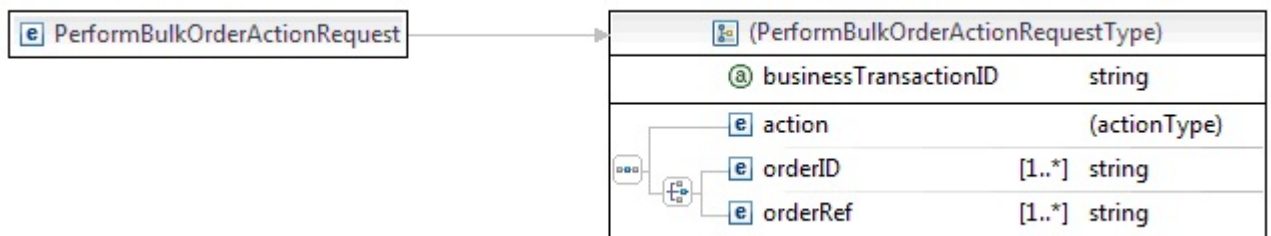
Bulk order action operation added to the Order Management Server order service



Bulk Orders Operation Request Schema

The following figure depicts the bulk operation request schema.

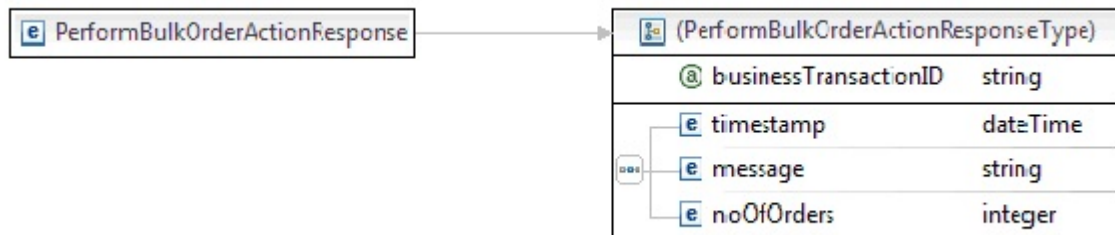
Request schema for bulk operation



Bulk Orders Operation Response Schema

The following figure depicts the bulk operation response schema.

Response schema for bulk operation



Sample Request

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

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ord="http://www.tibco.com/aff/orderservice">
  <soapenv:Header/>
  <soapenv:Body>
    <ord:PerformBulkOrderActionRequest businessTransactionID="bTranID">
      <ord:action>SUSPEND</ord:action>
      <ord:orderID>74</ord:orderID>
      <ord:orderID>56</ord:orderID>
    </ord:PerformBulkOrderActionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <ord:orderID>26</ord:orderID>
        <ord:orderID>30</ord:orderID>
        <ord:orderID>37</ord:orderID>
        <ord:orderID>88</ord:orderID>
        <ord:orderID>57</ord:orderID>
        <ord:orderID>27</ord:orderID>
        <ord:orderID>67</ord:orderID>
        <ord:orderID>35</ord:orderID>
    </ord:PerformBulkOrderActionRequest>
</soapenv:Body>
</soapenv:Envelope>

```

Sample Response

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

```

<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/commontypes" 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/
planfragments">
      <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>

```

Resource Failure Handling

The Resource Failure Handling feature identifies the failure or unavailability of resources as soon as possible and takes necessary actions. The Resource Failure Handling feature implements automatic reconnection feature for the key resources (JMS or DB) after failure or unavailability of these resources.

The Resource Failure Handling feature assists in the completion of the processing of previously submitted order without data loss, and in the suspending of the order processing after detecting resource failure.

Resource Failure Scenarios

The following are some resource failure scenarios:

JMS Failure

The JMS failure occurs when a connection to the Enterprise Message Service server is not available. The reason could be a connectivity issue with the Enterprise Message Service server, or the Enterprise Message Service server might have stopped. The Enterprise Message Service server can be configured as standalone, or in the fault tolerant mode. In the fault tolerant mode, if the application is not able to get an Enterprise Message Service connection, because of a network issue, it is identified as failure. If all the Enterprise Message Service servers in FT mode have stopped, and are not returning an Enterprise Message Service connection, a failure is triggered.

Additionally, for Enterprise Message Service standalone mode, if the application is not able to get an Enterprise Message Service connection, because of a network issue, it is identified as a failure.

Database Failure

The database failure occurs when a connection to the database server is not available. The reason could be a connectivity issue with database server or the database server might have stopped.

Batch Processor on Error

If the batching is configured, the database notifications are batched, executed and committed in batch. If the transaction fails, each notification from the batch is executed separately and is committed to the database. If

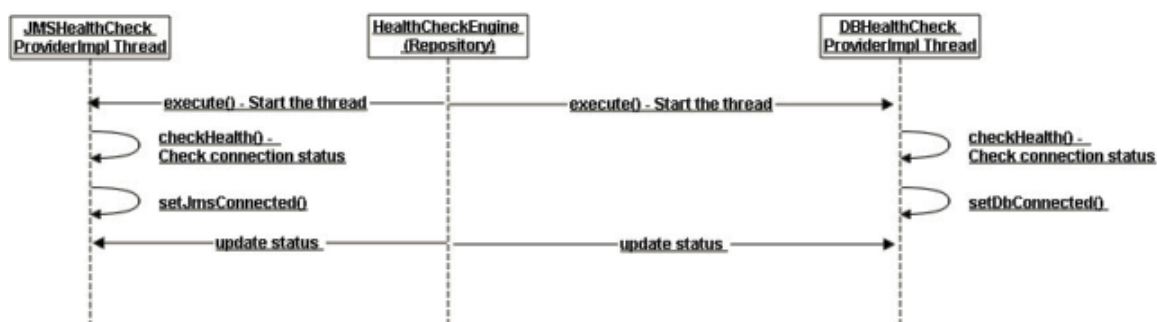
the transaction still fails, the batch processor marks the notification as a batch error and the notification is moved to dead letter table in the database. Any other notification for the same order is moved to the dead letter table. The batch processor is marked as an ERROR state if the database sub-batching, or batch task, fails after maximum retries.

Detection of Resource Failure

The detection of Resource Failure Handling has the following steps:

1. **Detecting the failure of Database or Enterprise Message Service:** A thread is configured, which checks the status of these resources at a predefined interval and sends the reports to a repository with the latest status. The repository can be accessed from other components and it can take the respective action. A timeout is configured when checking the status of the resources, and if a response is not received within the timeout period the failure of the particular resource is reported. The concerned thread also checks if any batch processor thread is in an error state. The properties or configuration used for failure detection is described in [Configuration of Resource Failure Handling](#).

The Fulfillment Order Management application runs another thread, which keeps looking at the repository data that is updated by failure detection thread, and identifies the resource failure when the status changes in the repository data.



2. **Using Spring Interceptor:** All the exceptions are thrown from the package `com.tibco.aff.oms.server` are intercepted by a class `LogException.java` that analyzes the thrown exception and identifies if the exceptions are thrown due to a resource failure. When intercepting the exception, if a resource failure issue is identified, an appropriate action on resource failure is taken. See [Action on Resource Failure](#) and [Action on Resource Recovery](#) for more details.
3. **JMS exception handler:** A JMS exception handler is registered to identify exception handling. Once an exception is detected by the exception handler the application verifies the JMS failure, and if a failure is found, an appropriate action is taken.
4. **Intercepting the exception thrown when sending the JMS message:** The Orchestrator uses `ResourceRouterImpl.dispatchMessage()` method to send any JMS messages. If an exception occurs when sending a message then the exception is sent to the `HealthCheck` thread and the resources are verified for resource failure.

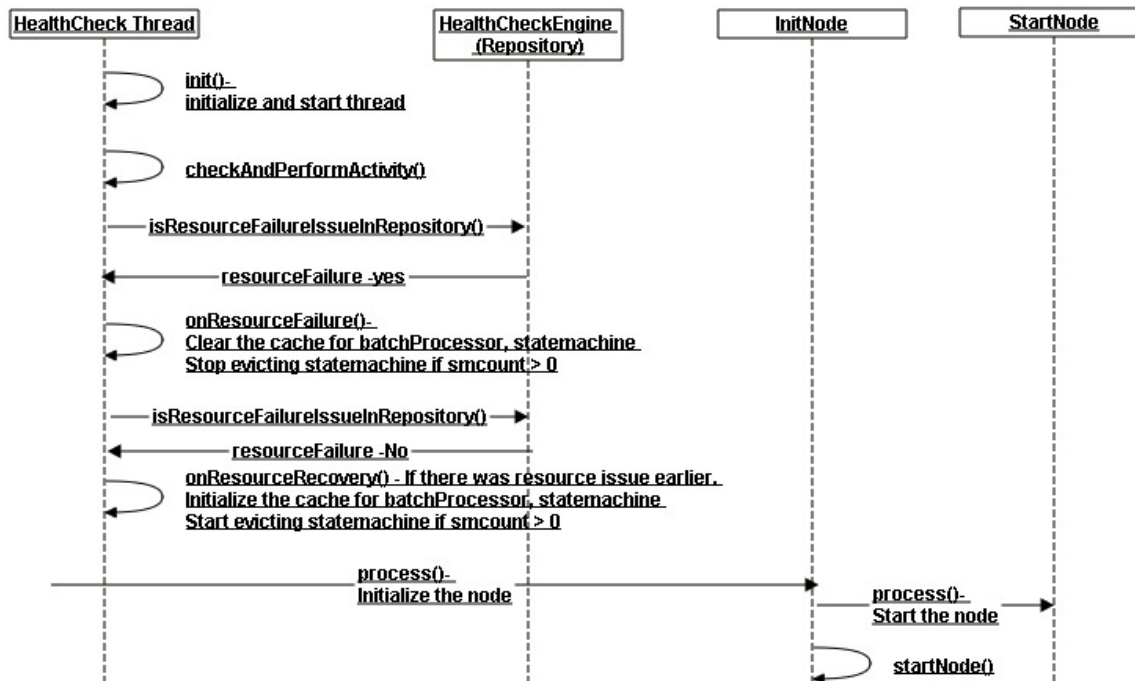
Action on Resource Failure

When a resource failure is identified the following activities are performed:

- Mark the cluster status as INIT and error state of cluster to true.
- Heartbeat processing of the node is stopped.
- Tasks related to finding cluster manager are paused.
- Stops the activity of monitoring the state machines for eviction.
- Start processing the Advisory message route. For additional details, see [Advisory Messages and Impact on Fault Tolerance Processing](#).

- Destroy the cache maintained by the Orchestrator. It includes stopping the processing and cleaning up of the batch processor threads maintained by the Orchestrator. All the JMS listeners run in the background.

All web services respond with the error code and message, indicating the resource failure. All JMS listeners in the Orchestrator remains started but messages are delivered back to the same source destination. Other JMS events respond indicating resource failure scenarios and the messages that are consumed.



Action on Resource Recovery

When a resource recovery is identified the following tasks are performed:


1. Initialize the node.
2. Mark the error state of the cluster as FALSE.
3. Heartbeat processing of the node is restarted.
4. Task-related to finding cluster manager is restarted.
5. Stop the processing of advisory message route. For additional details, see [Advisory Messages and Impact on Fault Tolerance Processing](#).
6. Initialize the batch processor threads.
7. Start the node.
8. Process cache cleanup messages.
9. Load pending batches.
10. Mark the node to STARTED.

Configuration of Resource Failure Handling

The following table describes the properties used for handling resource failure:

Resource Failure Handling Configuration

| SI No. | Name | Default Value | Used | Location |
|--------|--|---------------------|--|---|
| 1 | com.tibco.fom.orch.jms.socket.connection.timeout (JMS Socket connection timeout in milliseconds) | 500 milliseconds | Timeout interval for the socket, which is used to check the availability of the EMS server on the defined port. | \$OM_HOME/roles/configurator/standalone/config/ConfigValues_OMS.xml |
| 2 | com.tibco.fom.orch.jms.jndiLookup.connection.timeout (JMS JNDI connection check timeout in milliseconds) | 10000 milliseconds | A threshold interval for which the thread waits to get the connection factory during lookup as part of the EMS monitoring. | \$OM_HOME/roles/configurator/standalone/config/ConfigValues_OMS.xml |
| 3 | com.tibco.fom.orch.db.connection.timeout (DB connection check timeout in milliseconds) | 10000 milliseconds | A threshold interval for which the thread waits to get the database status during lookup as part of the database monitoring. | \$OM_HOME/roles/configurator/standalone/config/ConfigValues_OMS.xml |
| 4 | Heartbeat interval | 7000 milliseconds | Interval between successive heartbeat to check the availability of the node. | Columns of the DOMAIN table of the Order Management Server user. |
| 5 | FT Threshold interval | 100000 milliseconds | Interval after which the cluster manager assumes that the node has failed if it does not receive the heartbeat from failed node. | Columns of the DOMAIN table of the Order Management Server user. |
| 6 | Manager activation interval | 40000 milliseconds | Interval to check if the node can become a cluster manager and start performing the cluster manager activities. | Columns of the DOMAIN table of the Order Management Server user. |

| SI No. | Name | Default Value | Used | Location |
|--------|--|---------------------|--|--|
| 7 | Advisory message publish interval | 7000 milliseconds | Interval at which a heartbeat is sent using the available resources from database or JMS. This heartbeat information is used to identify whether to treat a node as a failed node or not. | Columns of the DOMAIN table of the Order Management Server user. |
| | | |  <p>Value for this property is referred from the heartbeat interval. If you change the heartbeat interval, the advisory message publish interval is also changed to the same value.</p> | |
| 8 | com.tibco.fom.orch.cluster.channel.maxIdleTime (Max Idle time reset the channel Error in Milliseconds) | 120000 milliseconds | Interval used to reset the heartbeat channel error. | |
| 9 | Handle Failed Node | 1 | <p>If the value is</p> <ul style="list-style-type: none"> • 1: Cluster manager detects and recover failed node event • -1: To disable cluster failover recovery | Columns of the DOMAIN table of the Order Management Server user. |

Advisory Messages and Impact on Fault Tolerance Processing

The advisory heartbeat message is sent when a resource failure is identified. This message is sent to the database and the JMS at a regular interval of heartbeat beat interval configured for the cluster. The messages are sent to JMS and the database, and the available resource can process the respective message.

The message contains information such as NodeId, Update Timestamp, and a Message.

Database advisory messages are stored in DOMAINMEMBERS_ADVISORY table. A record is inserted in this table till there is resource failure. The DOMAINMEMBERS_ADVISORY table is a new table.

JMS advisory messages are sent to the topic `tibco.aff.orchestrator.cluster.advisory.heartbeat`, and this message is received by the corresponding listener if the JMS resource is available along with the updates the in-memory data and the message content.

This information helps to identify the nodes that can process an advisory message and subsequently perform fault tolerance activity within the nodes. When the nodes resume the normal state, then the cluster manager performs certain fault tolerance related activities for the failed nodes. Before performing any fault tolerance activities, the cluster manager checks if there is not any advisory message available for the failed node that is older than the Fault Tolerance threshold interval. If advisory message is available for the particular failed node then the fault tolerance activities are not performed for the failed node.

The messages are deleted if they are older than twice the Fault Tolerance threshold for the sources.

Event Processing on Orchestrator when it is Not Started

If the node status has not been set to `STARTED`, the Orchestrator does not process any notification. In case of resource recovery, if all the listeners are running, they can receive messages from the JMS destination. Upon recovery, activities like cleaning up the cache, cleaning up the queue, and loading the pending batches require some time to process. Once the processing is complete the node status is marked as `STARTED` and the node is available to receive new messages from respective JMS destination. JMS listeners of the Orchestrator does not process messages until the node status is set to `STARTED`. Other JMS listeners like Transient Data Store interfaces can process the messages.

Multitenancy

The term multitenancy indicates an architecture in which a single running instance of an application simultaneously serves multiple clients or "tenants".

Isolating information, such as data and customizations, about the tenants is a particular challenge in these systems. This includes the data owned by each tenant stored in the database. A single instance of the application can now support multiple tenants. SOAP clients, the Order Management Server UI and PC can talk to the same TIBCO Order Management - Long Running instance for processing and viewing the orders based on the tenant context. The TIBCO Order Management - Long Running nodes form a cluster and support the scaling of nodes based on the load. A default tenant "TIBCO" is supported by TIBCO Order Management - Long Running without any configuration changes.

Each tenant's data is kept in a distinct database schema on a single database instance. Connections point specifically to each schema. There is a distinct JDBC connection pool per tenant where the pool to use is selected based on the "tenant context" associated with the currently logged in user.

The application primarily has two database schemas: one schema for storing admin data and one schema per tenant for storing tenant data.

Tenant and related data source information are stored in the admin database schema. Each tenant has its database schema as configured in the admin schema. By supporting multiple schemas, the application is able to route data to different databases based on the specified tenant and configuration.

Database details for the admin schema are provided in the `ConfigValues_OMS.xml` file under the category "Data Source Configuration" and the database details for the default tenant are provided under the category "Data Source Configuration Default Tenant".

Adding a New Tenant

Complete the following steps to add a new tenant to enable order processing for the added tenant.



If you are using an Oracle database, find the files in the `$OM_HOME\db\oracle\oms` directory. If you are using a PostgreSQL database, find the files in the `$OM_HOME\db\postgresql\oms` directory.

Procedure

1. Create a new database user and tablespace corresponding to the new tenant and run the `OMS_DDL.sql` script for this user.
2. To insert seed data for the new tenant, run the `OMS_SeedData.sql` script.
3. Modify the property file present at `$OM_Home\db\<oracle or PostgreSQL>\oms\TenantUtility\OMS_Manage_Tenant.properties`:
 - a) Enter the tenant name, tenant ID, tenant data source information.
 - b) Enter the admin database information.
4. Run the script `OMS_Manage_Tenant.sh` after verifying the information stored in the property file in step 2.
 through the script, you can add or update the tenant-specific configuration. This is possible by altering the tenant config section in the property file `OMS_Manage_Tenant.properties` and running the script again. Based on the poller frequency, the new tenant properties are applicable in the order flow.

Result

The new tenant is now configured and order processing is possible.

Removing a Tenant

Complete the following steps to remove an added tenant



If you are using an Oracle database, find the file in the `$OM_HOME\db\oracle\oms\TenantUtility` directory. If you are using a PostgreSQL database, find the file in the `$OM_HOME\db\postgresql\oms\TenantUtility` directory.

Procedure

- Run the following script as shown:

```
OMS_Manage_Tenant.sh -r <tenantid>
```

Manage Tenant Script Options

The following table explains options for the `OMS_Manage_Tenant.sh` script.



Ensure that the `OMS_Manage_Tenant.properties` file is configured with the correct values before invoking this script.

| Option | Description |
|---|--|
| <code>OMS_Manage_Tenant.sh</code> | Add a tenant with configuration parameters present in the default properties file. |
| <code>OMS_Manage_Tenant.sh -f <file></code> | Adds a tenant with the configuration parameters present in the file specified through the <code><file></code> command line argument. |
| <code>OMS_Manage_Tenant.sh -h</code> | Displays the options and general usage information related to the script. |
| <code>OMS_Manage_Tenant.sh -r</code> | Removes a tenant with the tenant ID specified in the default properties file. |

| Option | Description |
|---|---|
| <code>OMS_Manage_Tenant.sh -r <tenantid></code> | Removes a tenant with the tenant ID specified through the <tenantid> command line argument. |
| <code>OMS_Manage_Tenant.sh -f <file> -r</code> | Removes a tenant with the tenant ID specified in the file specified through the <file> command line argument. |

Cloning a Pluggable Database (PDB)

This utility is specifically for users who are using an Oracle Pluggable Database (PDB). Run the `$OM_HOME/db/oracle/oms/TenantUtility/OMS_Clone_Tenant.sql` script to clone the existing tenant Pluggable Database. Cloning the existing tenant Pluggable Database is a quicker way of creating a new tenant Pluggable Database from an existing Pluggable Database. By cloning a tenant Pluggable Database, you can skip executing the DDLs and seed data scripts.



You must complete the prerequisites before running the `OMS_Clone_Tenant.sql` script.

Prerequisites

- [Creating a Source Pluggable Database](#)
- [Preparing the Cloning Script](#)

Creating a Source Pluggable Database

Before executing the `OMS_Clone_Tenant.sql` script, you must complete the following steps.

Procedure

1. Create a source Pluggable Database for the tenant.
2. Execute `OMS_DDL.sql` and `OMS_SeedData.sql` for this Pluggable Database.
This newly created tenant Pluggable Database is considered as a master-copy for creating any further tenants Pluggable Databases. You can call this Pluggable Database "pdb0". Don't store any user data in pdb0 except for the seed data.
3. Navigate to wherever you can find enough available space to store the data files for the new tenant's Pluggable Database.
Usually, the tenant's directory is under `/u01/app/oracle/oradata/orcl/` (but this might vary according to the environment).
4. You can call the cloned Pluggable Database, which you are trying to create `pdb1`. Create a subdirectory called `pdb1` under your chosen directory in step 3 (for example - under `orcl/`).



You might have to log in as a root user to create the subdirectory, and then change the owner to Oracle and the group to install (considering you have installed Oracle 12c with OS user Oracle, which belongs to the install group).

5. Login as SYSDBA and execute the following command:

```
alter session set container=CDB$ROOT;
```
6. Close the source (master-copy) Pluggable Database by executing the following command:

```
alter pluggable database <<source_pdb_name>> close immediate;
```

Preparing the Cloning Script

You must prepare the script `OMS_Clone_Tenant.sql` before executing it.

Procedure

1. Replace `<<source_pdb_name>>` with the actual source Pluggable Database name. If you followed the naming convention suggested in [Creating a Source Pluggable Database](#), it would be `pdb0`.
2. Replace `<<path_to_new_pdb_directory>>` with the actual directory path to the new directory that you created to store the cloned Pluggable Database's data files. If you used the same directory and naming convention in [Creating a Source Pluggable Database](#), this would be `/u01/app/oracle/oradata/orcl/pdb1`.
3. Replace `<<new_pdb_name>>` with the new `pdb` name. If you followed the naming convention suggested in [Creating a Source Pluggable Database](#), it would be `pdb1`.
4. Execute the `OMS_Clone_Tenant.sql` script as SYSDBA.

The new tenant is created with all the users, tables, and data, which were existing in the original tenant.

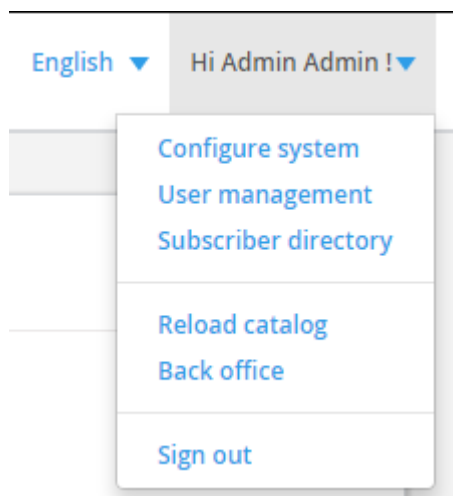
5. Update the `tnsnames.ora` file by adding a new listener entry for the newly created Pluggable Database (`pdb1`).
6. Now you are ready to use the newly created container (`pdb1`) as a tenant database. Use the service name provided for the new Pluggable Database to connect from the `ConfigValues_OMS.xml` file.

Order Capture System User Interface Tasks

The following tasks can only be completed through an administrator account: configuring the Order Capture System UI, managing users, accessing Demo Subscriber Directory, reloading the catalog, and accessing the back office.

- [Configuring the System](#) - Use this feature to run an interaction clean up, set the unsellable products and change the customer label.
- [User Management](#) - Use this feature to create, edit, or delete users.
- [Demo Subscriber Directory](#) - For demonstration purposes, use Demo Subscriber Directory to create, edit, and delete subscribers and stores.
- [Reloading the Catalog](#) - Use this feature to reload the TIBCO Product and Service Catalog to refresh the data model from the TIBCO Product and Service Catalog XML files.
- [Back Office](#) - Use this feature to look at an overview of the Order Capture System framework and to generate a statistics report.

All of these tasks can be accessed through the drop-down menu.





Both of the administrator's default user name and password are admin. This can be changed by an administrator in the User Edition page. Refer to [Editing Users](#).

Configuring the System

On the Configure system page, you can configure settings such as setting the unsellable products and running interaction cleanup.

Procedure

1. To access the Configuration page, click **Configure system** in the drop-down menu.
2. Set any of the following configurations:
 - **Non-Sellable Types:** Set the unsellable product types that do not show up in the Order Capture System product selection.
 - **Run Interaction Cleanup:** Clean up the expired interactions from the database.
 - **Interaction Life Time Seconds:** Set the number of seconds before the interaction expires.
 - **Interaction CleanUp Schedule Time Seconds:** Set the time of day when to run the cleanup.
3. Click **Save**.

User Management

Use the User Management page to enter, edit, and delete users in the system.

Entering Users

To add a user, enter the user's name, ID, and password in the User creation page.

Procedure

1. Click the drop-down menu and select **User Management**.
2. From the User management page, click **NEW USER**.
3. Enter the new user's information in the fields.



The User ID can have up to 100 characters and the password can have up to 40 characters with no restrictions as to what characters might or might not be used.



Click the **Admin** box only if you want this user to have administrative privileges.

4. Click **SAVE**.

Editing Users

Edit a user's information using the User management page.

Procedure

1. Click the drop-down menu and select **User Management**.
2. On the User management page, search for a user in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

3. Click **Edit** in the user's row in the ACTIONS column.

4. Edit the user's information where needed.
5. Click **SAVE**.

Deleting Users

Delete a user using the User management page.

Procedure

1. Click the drop-down menu and select **User Management**.
2. On the User management page, search for a user in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

3. Click **Delete** in the user's row in the ACTIONS column.
4. Click **yes** to confirm the deletion.

Demo Subscriber Directory

For demonstration purposes, you can use Demo Subscriber Directory in Order Capture System to manage a list of subscribers and stores.

In a real production environment, Order Capture System retrieves subscriber information from a Customer Relationship Management system (CRM). The configuration with a CRM is done using the WSDL defined in TIBCO Order Management - Long Running Web Services Guide. However, for demonstration purposes, Order Capture System can be configured to use the Demo Subscriber Directory in place of a subscriber inventory. Demo Subscriber Directory is enabled by default. Demo Subscriber Directory can be enabled or disabled in TIBCO Configuration Tool or TIBCO Master Data Management Configurator. For instructions on completing this task, see [Demo Subscriber Directory Toggling](#).



Demo Subscriber Directory is not meant for production and is only supported in the demonstration context.

List of Subscribers

Demo Subscriber Directory lists the subscribers that match your specific search criteria. The table is sortable in ascending or descending order. By using the links in the action column, you can edit or delete the selected subscriber.

Subscriber Creation

When creating a new subscriber, Demo Subscriber Directory prompts you to enter the information of the new subscriber, such as the first name, last name, address, and closest store. The closest store entry corresponds to an instance of a store already configured. Each subscriber is associated with a store and has a set of predefined segments. Demo Subscriber Directory prompts you to configure the initial set of segments for the subscriber. Each segment is used to retrieve the eligible list of products, services, or bundles from a product catalog.

List of Stores

Demo Subscriber Directory lists the stores that match your specific search criteria. The table is sortable by ascending or descending order. By using the links in the action column, you can edit or delete the selected store.

Store Creation

Demo Subscriber Directory prompts you to enter information, such as name and address, of the store you are creating. After you save the store creation, it is added to the list of stores.

Accessing the List of Subscribers

To add, edit, or delete a subscriber, access the List of Subscribers page in the Demo Subscriber Directory.

Procedure

1. Click the drop-down menu and select **Subscriber directory**.
2. Click **Subscribers**.
3. Click the arrow to navigate back to the Demo Subscriber Directory home page.



Click **Order Capture System**

Order Capture System

on the header to navigate back to the Order Capture System home page.

Entering Subscribers

To add a subscriber, enter the subscriber's information and configure the initial set of segments for the subscriber.

Prerequisites

Access the List of Subscribers. For instructions on completing this task, see [Accessing the List of Subscribers](#).

Procedure

1. On the List of Subscribers page, click **NEW SUBSCRIBER**.
2. On the Subscriber creation page, enter the new subscriber's information in the **Personal Information** fields.
3. Set the segments by clicking **Set Values**.



Click the cogwheel to edit each specific segment field. You can add, delete, ascend, or descend a segment.

4. Click **SAVE**.

Editing Subscribers

Edit a subscriber's information using the List of Subscribers page.

Prerequisites

Access the List of Subscribers. For instructions on completing this task, see [Accessing the List of Subscribers](#).

Procedure

1. On the List of Subscribers page, search for a subscriber in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

2. Click **Edit** in the subscriber's row in the ACTIONS column.
3. On the Subscriber edition page, edit the subscriber's information where needed.

4. Click **SAVE**.

Deleting Subscribers

Delete a subscriber using the List of Subscribers page.

Prerequisites

Access the List of Subscribers. For instructions on completing this task, see [Accessing the List of Subscribers](#).

Procedure

1. On the List of Subscribers page, search for a subscriber in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

2. Click **Delete** from the subscriber's row in the ACTIONS column.
3. Click **yes** to confirm the deletion.

Accessing the List of Stores

To add, edit, or delete a store, access the List of Stores page in the Demo Subscriber Directory.

Procedure

1. Click the drop-down menu and select **Subscriber directory**.
2. Click **Stores**.



Click the arrow to navigate back to the Demo Subscriber Directory home page.



Click **Order Capture System** on the header to navigate back to the Order Capture System home page.

Entering Stores

To add a store, enter the store's name and address into the Demo Subscriber Directory.

Prerequisites

Access the List of Stores. For instructions on completing this task, see [Accessing the List of Stores](#).

Procedure

1. On the List of Stores page, click **NEW STORE**.
2. On the Store creation page, enter the new store's information in the fields.
3. Click **SAVE**.

Editing Stores

Edit a store's information from the List of Stores page.

Prerequisites

Access the List of Stores. For instructions on completing this task, see [Accessing the List of Stores](#).

Procedure

1. From the List of Stores page, search for a store in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

2. Click **Edit** in the store's row in the ACTIONS column.
3. On the Store edition page, edit the store's information where needed.
4. Click **SAVE**.

Deleting Stores

Prerequisites

Access the List of Stores. For instructions on completing this task, see [Accessing the List of Stores](#).



You can delete a store only if it is not used by any subscribers.

Procedure

1. On the List of Stores page, search for a store in the **Search** field.



The search input supports complex searches. For more information, see [Search Syntax](#).

2. Click **Delete** from the store's row in the ACTIONS column.
3. Click **yes** to confirm the deletion.

Reloading the Catalog

Reload the TIBCO Product and Service Catalog to refresh the data model from the TIBCO Product and Service Catalog XML files.



To reload the catalog, all sessions must be cleared and you are logged out.

Procedure

1. Check all the boxes under Reload catalog:
 - Clear all sessions
 - Reload the packages
 - Logout current user session
2. Click **RELOAD**.

Back Office

With administrative privileges, you have access to the back office where an overview and a statistics report can be generated.

Overview

The Overview displays Order Capture System UI and Order Capture System server information.

The Order Capture System UI section displays information such as the version of AngularJS and JQuery.

The Order Capture System server, Runtime Model Server (RMS), section in the Overview displays information such as the start time and last update.

Statistics

Statistics in the back office is useful to monitor activity and diagnose potential performance issues arising in the Order Capture System activity.

Order Capture System is connected to and depends on many external systems and their response time to operate. Statistics can be a way to monitor this activity.

The statistics section displays information about the Order Capture System Server activity, such as low level method call durations and external system accesses.

To run the Statistics report click **Start**.

To end the report click **Stop**.



The statistics report is intended as a TIBCO engineer diagnostic tool but can be used by administrators to monitor activity.

Search Syntax

The search input supports complex searches when searching for users, subscribers, or stores.

For example, if you are searching for users or subscribers and type "th," you get results having "th" in their first name, last name or ID such as "Maria *Thompson*", "John *Smith*", or "Dorothy Robinson."

If you type "th ma", you get results having both "th" and "ma" such as "Maria *Thompson*" or "Macy *Smith*."

If you type "th OR jam," you get results having either "th" or "jam" such as "Maria *Thompson*" or "James Dickinson."

If you are searching for stores and type "sh," you get results having "sh" in the store's name or ID such as "Shop XYZ."

If you type "pr es", you get results having both "pr" and "es" such as "Product *Supplies*."

If you type "su OR bl," you get results having either "su" or "bl" such as "Supply *Shop*" and "Blue *Store*."

Managing Health Check Endpoint

TIBCO Order Management - Long Running supports the health check endpoint to check the overall health of application resources such as Enterprise Message Service (EMS), database, and disk space. You can check the health status of any service by putting the respective host and port number of that service in the following format.

Health check endpoint: `http://<host>:<port>/management/health`

Example of health check response:

```
configurator:
=====
```

```
http://localhost:9090/management/health
  {"status":"UP","diskSpace":{"status":"UP","total":367000547328,
    "free":249138311168,"threshold":10485760}}
```

Tuning Performance

This section covers the tuning performance for the Order Management Server component.

Order Management Server Performance Tuning

There have been significant performance improvements to Order Management Server. 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 [Order Management Server Performance Tuning](#)). Each queue supports multiple concurrent listeners for processing the notifications in parallel. The number of concurrent listeners for the queues are configurable and can be configured using the TIBCO Order Management - Long Running Configurator user interface.

Listener Queues

| Configuration and Setup For OMS - Listener Queues | | |
|---|--|---|
| Add New Property Clone Delete | | |
| Property | Value | Description |
| Central Log Queue | tibco.aff.centrallog.queue | Central Log Queue |
| Central Log Queue Concurrent Listener Count | 1 | Central Log Queue Concurrent Listener Count |
| Synchronous Order Submission Status Recovery Consumer Count | 1 | Synchronous Order Submission Status Recovery Consumer Count |
| Synchronous Order Submission Status Recovery Consumer Queue | tibco.aff.oms.syncorderstatusrecovery | Synchronous Order Submission Status Recovery Queue |
| Order Service Queue | tibco.aff.oms.orderservice | Queue for receiving SOAP Over JMS Order Service requests |
| Minimum number of concurrent consumers for listener | 1 | Minimum number of concurrent consumers for listener (default 1) |
| Maximum number of concurrent consumers for listener | 5 | Maximum number of concurrent consumers for listener (default 1) |
| Milestone Notify Request Queue | tibco.aff.oms.planitem.milestone.notifyrequest | Milestone Notify Request Status Change Notification Queue |

For details on the properties of the Listener Queues, see [Status Listener Queues](#).

Also, the number of concurrent listeners for JMS interfaces from AFI, AFS, and Transient Data Store components merged as the integral part of Order Management Server can also be configured according to the need.

AFI, Transient Data Store, AFS Configuration

Select Configuration

Order Management System

Cluster Outline

OMS

.....

Configuration Outline

Basic

Advanced

- Data Source Configur...
- Persistence
- Catalog Publishing J...
- Offline Catalog Confi...
- Standalone AOPD Int...
- Data Interfaces Confi...
 - JMS Destinations...

Configuration and Setup For OMS - Data Interfaces Configuration - JMS Destinations Configuration

Add New Property

Clone

Delete

| Property | Value | Description |
|---|---|---|
| GetOrderRequest receiver queue | tibco.aff.tds.order.read.request | GetOrderRequest receiver queue |
| GetOrderRequest receiver count | 3 | GetOrderRequest receiver count |
| GetOrderRequest receiver dead queue | tibco.aff.oms.tds.order.read.request.dead | GetOrderRequest receiver dead queue |
| GetOrderResponse sender queue | tibco.aff.tds.order.reply | GetOrderResponse sender queue |
| GetPlan/GetPlanItem Request receiver queue | tibco.aff.tds.plan.read.request | GetPlan/GetPlanItem Request receiver queue |
| GetPlan/GetPlanItem Request receiver count | 3 | GetPlan/GetPlanItem Request receiver count |
| GetPlan/GetPlanItem Request receiver dead queue | tibco.aff.oms.tds.plan.read.request.dead | GetPlan/GetPlanItem Request receiver dead queue |

Status Listener Queues

When Order Management Server receives orders, it sends the orders to JMS queues, which are routed to the Orchestrator engine based on the 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 let parallel processing of incoming orders. The number of such concurrent listeners is configurable.

Order Management Server receives a notification on status for orders in the system for fulfillment and supports two router 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.

Listener Queues

| | | | |
|---|---|--|---|
| <div>Select Configuration</div> <div>Order Management System</div> <div>Cluster Outline</div> <div>OMS</div> <div>Member1</div> <div>Configuration Outline</div> <div>Basic Advanced</div> <div>Listener Queues</div> <div>OMS User Interface</div> <div>Messaging Configura...</div> <div>Data Source Configur...</div> <div>Data Source Configur...</div> <div>Persistence</div> <div>Catalog Publishing J...</div> <div>Offline Catalog Confi...</div> | Configuration and Setup For OMS - Listener Queues | | |
| | Add New Property Clone Delete | | |
| | Property | Value | Description |
| | Central Log Queue | tibco.aff.centrallog.queue | Central Log Queue |
| | Central Log Queue Concurrent Listener Count | 1 | Central Log Queue Concurrent Listener Count |
| | Synchronous Order Submission Status Recovery Consumer Count | 1 | Synchronous Order Submission Status Recovery Consumer Count |
| | Synchronous Order Submission Status Recovery Consumer Queue | tibco.aff.oms.syncorderstatusrecovery | Synchronous Order Submission Status Recovery Queue |
| | Order Service Queue | tibco.aff.oms.orderservice | Queue for receiving SOAP Over JMS Order Service requests |
| | Minimum number of concurrent consumers for listener | 1 | Minimum number of concurrent consumers for listener (default 1) |
| | Maximum number of concurrent consumers for listener | 5 | Maximum number of concurrent consumers for listener (default 1) |
| | Milestone Notify Request Queue | tibco.aff.oms.planitem.milestone.notifyrequest | Milestone Notify Request Status Change Notification Queue |

The following table describes the properties of the Listener Queues.

| Name | Descriptions |
|---|---|
| Order Status Notification Queue | Destination queue for receiving status message from Orchestrator |
| Order Status Notification Queue Concurrent Listener | Order Status Notification Queue Concurrent Listener Count |
| Order Line Status Notification Queue | Order Line Status Notification Queue |
| Order Line Status Notification Queue Concurrent Listener | Order Line Status Notification Queue Concurrent Listener Count |
| Plan Status Notification Queue | Plan Status Notification Queue |
| Plan Status Notification Queue Concurrent Listener | Plan Status Notification Queue Concurrent Listener Count |
| Plan Item Status Notification Queue | Plan Item Status Notification Queue |
| Order Amendment Status Notification Queue | Order Amendment Status Notification Queue |
| Set Plan Queue | Destination queue for receiving execution plan from Orchestrator |
| Set Plan Item Queue | Destination queue for receiving execution plan item from Orchestrator |

| Name | Descriptions |
|---|--|
| Set Plan Fragment Model Queue | Destination queue for receiving plan fragment from TIBCO Product and Service Catalog |
| Status Notification Dead Queue | Destination queue for poison message for status notification. Order Management Server moves the message to the dead letter queue after retrying a pre-configured number of times to avoid messages becoming poison message |
| Set Plan Dead Queue | Dead letter queue destination for execution plan |
| Set Plan Item Dead Queue | Dead letter queue destination for execution plan item |
| Set Plan Fragment Dead Queue | Dead letter queue destination for plan fragment |
| Update Jeopardy Configuration Rule Queue | Destination to send jeopardy rule updates |
| Enrich Migrated Plan Request Queue | Destination to send request to Jeopardy Management System to enrich execution plan with the jeopardy information |

The independent AFI, AFS, and Transient Data Store components have been rewritten in release 2.0.0 and they are now an integral part of Order Management Server. So apart from the above-listed queue configurations for core Order Management Server 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 *TIBCO Order Management - Long Running User Guide*.

Changing Transient Data Store Operation Messages to Non-Persistent

TIBCO Enterprise Message Service messages including replies for all Transient Data Store operations are persistent. These messages can be made non-persistent using the provided configurations.

Procedure

- Configure the following properties under the category "Data Interfaces Configuration" and sub-category "Data Interface Flags" in TIBCO Order Management - Long Running Configurator or in the ConfigValues_OMS.xml file:

```
<ConfValue description="JMS QoS Enabled for RPC get operations" name="JMS QoS
Enabled for RPC get operations" propname="com.tibco.fom.oms.tds.get.jms.qosEnabled"
sinceVersion="3.0.2" visibility="Basic">
  <ConfBool default="false" value="false"/>
</ConfValue>

<ConfValue description="replyToDeliveryPersistent flag for RPC get operations"
name="replyToDeliveryPersistent flag for RPC get operations"
propname="com.tibco.fom.oms.tds.get.jms.replyToDeliveryPersistent"
sinceVersion="3.0.2" visibility="Basic">
  <ConfBool default="true" value="true"/>
</ConfValue>

<ConfValue description="deliveryPersistent flag for NON-RPC get operations"
name="deliveryPersistent flag for RPC get operations"
propname="com.tibco.fom.oms.tds.get.jms.deliveryPersistent" sinceVersion="3.0.2"
```

```

visibility="Basic">
    <ConfBool default="true" value="true"/>
</ConfValue>

<ConfValue description="JMS QoS Enabled for RPC set operations" name="JMS QoS
Enabled for RPC set operations" propname="com.tibco.fom.oms.tds.set.jms.qosEnabled"
sinceVersion="3.0.2" visibility="Basic">
    <ConfBool default="false" value="false"/>
</ConfValue>

<ConfValue description="replyToDeliveryPersistent flag for RPC set operations"
name="replyToDeliveryPersistent flag for RPC set operations"
propname="com.tibco.fom.oms.tds.set.jms.replyToDeliveryPersistent"
sinceVersion="3.0.2" visibility="Basic">
    <ConfBool default="true" value="true"/>
</ConfValue>

<ConfValue description="deliveryPersistent flag for NON-RPC set operations"
name="deliveryPersistent flag for RPC set operations"
propname="com.tibco.fom.oms.tds.set.jms.deliveryPersistent" sinceVersion="3.0.2"
visibility="Basic">
    <ConfBool default="true" value="true"/>
</ConfValue>

```

TIBCO Order Management - Long Running Disaster Recovery

This topic covers all details related to TIBCO Order Management - Long Running Disaster Recovery considerations.

This section covers guidelines for planning and deploying this software for disaster recovery and application continuous availability. This document does not provide guidelines for deploying process components and northbound TIBCO Order Management - Long Running client application for disaster recovery.

TIBCO Order Management - Long Running Topology for Disaster Recovery

Ensure that the Symmetric topology is used for disaster recovery. TIBCO Order Management - Long Running configured for disaster recovery must be completely identical across the tiers on the production site and standby site.

In the symmetric topology, the production site and standby site have the identical number of hosts, load balancers, TIBCO Order Management - Long Running 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.

Storage and Volumes

Create the volumes on your storage device and mount them appropriately on your TIBCO Order Management - Long Running nodes. These volumes must be created on the primary site and the standby site. 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 might have 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. Also, to using file system for storing configuration files, TIBCO Order Management - Long Running uses file systems to persist operation data in directories of the file system. These directory structure must be maintained on both production and stand by sites.

Following is the list of directory used by components of TIBCO Order Management - Long Running and its property name. These directories must have identical directory structure on both production site and standby site and must be replicated using storage-to-storage disaster recovery such as Symmetrix Remote Data Facility/Asynchronous (SRDF/A)

Order Management System

| Property Name | Description |
|--|---|
| <code>com.tibco.af.oms.router.recoveryFileFolderPath</code> | Path to folder containing Router Recovery Files. Uses by router to recover from system crush. |
| <code>com.tibco.fom.oms.afi.offline.product.directory</code> | Offline product catalog directory. |
| <code>com.tibco.fom.oms.afi.offline.product.importsuccess.directory</code> | Product catalog import success poller and web service directory |
| <code>com.tibco.fom.oms.afi.offline.product.importfailure.directory</code> | Product catalog import failure poller and WS directory |

| Property Name | Description |
|---|--|
| <code>com.tibco.fom.oms.afι.offline.planfragment.directory</code> | Offline plan fragment catalog directory. |
| <code>com.tibco.fom.oms.afι.offline.planfragment.importsuccess.directory</code> | Planfragment catalog import success poller and web service directory |
| <code>com.tibco.fom.oms.afι.offline.planfragment.importfailure.directory</code> | Planfragment catalog import failure poller and web service directory |
| <code>com.tibco.fom.oms.afι.offline.action.directory</code> | Offline action catalog directory. |
| <code>com.tibco.fom.oms.afι.offline.action.importsuccess.directory</code> | Action catalog import success poller and web service directory |
| <code>com.tibco.fom.oms.afι.offline.action.importfailure.directory</code> | Action catalog import failure poller and web service directory |

You also have to periodically replicate TIBCO Order Management - Long Running configuration files from production to disaster recovery site

| Directory | Content | Replication Requirement |
|-----------------------|---|--|
| <i>TIBCO_OM_HOME</i> | TIBCO Order Management - Long Running Software | Replicate this directory whenever there has been a change to the deployed TIBCO runtime software. For example, when: <ul style="list-style-type: none"> • software has been installed. • a hotfix, service pack or upgrade to an existing software component has been added. |
| <i>EMS_HOME</i> | TIBCO Enterprise Message Service Software | You can either install Enterprise Message Service on the disaster recovery site, or else replicate this directory. Replicate this directory when there has been a change in the Enterprise Message Service runtime software. |
| <i>OM_CONFIG_HOME</i> | TIBCO Order Management - Long Running Configuration Directory | Replicate this directory whenever there has been a change in the configuration. |

Network

Ensure that all the ports required by components of TIBCO Order Management - Long Running are configured with the same port number. This software supports distributing components across network. All

the components excluding Order Management Server UI, communicated with other component using TIBCO Enterprise Message Service. Order Management Server UI communicates with Order Management Server using HTTP. If layer 7 load balancer is used within a site, Order Management Server UI must always be configured to point to the local site load balancer, even if geo-targeted global load balancer is used to dynamically update the DNS entry and activate the standby site. You must use hostnames 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.

The following table lists the property name and host and port number used by Order Management Server UI and Order Management Server components of TIBCO Order Management - Long Running for HTTP communication:

| Property Name | Description |
|---|---|
| <code>com.tibco.af.omsServer.proxyHost</code> | Host address of the Order Management Server |
| <code>com.tibco.af.omsServer.proxyPort</code> | Port number of the Order Management Server |

If TIBCO Fulfillment Provisioning is integrated with TIBCO Order Management - Long Running, the following additional properties must point to local site load balancer:

| Property Name | Description |
|--|--|
| <code>com.tibco.af.fpServer.proxyHost</code> | Host address of the TIBCO Order Management - Long Running server |
| <code>com.tibco.af.fpServer.proxyPort</code> | Port number of the TIBCO Order Management - Long Running server |

Database

On both the primary site and the DR site, create a service name alias that TIBCO Order Management - Long Running can use to identify the database to which it connects.

TIBCO Order Management - Long Running 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 disaster recovery site have different SIDs, you must define an Oracle service name that it can use as an alias, allowing it to connect to the database whether it is currently running on the primary system or the disaster recovery system.

Refer to the Oracle Data Guard for best practices of database disaster recovery.

Messaging Server

Use the fault tolerant mechanism provided by the TIBCO Enterprise Message Service servers 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 Enterprise Message Service 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 Enterprise Message Service server logs, configuration files and file stores on a shared Storage Area Network to be accessed by multiple nodes. The shared disk subsystem must be accessible from standby servers to prevent disk failures or node outages

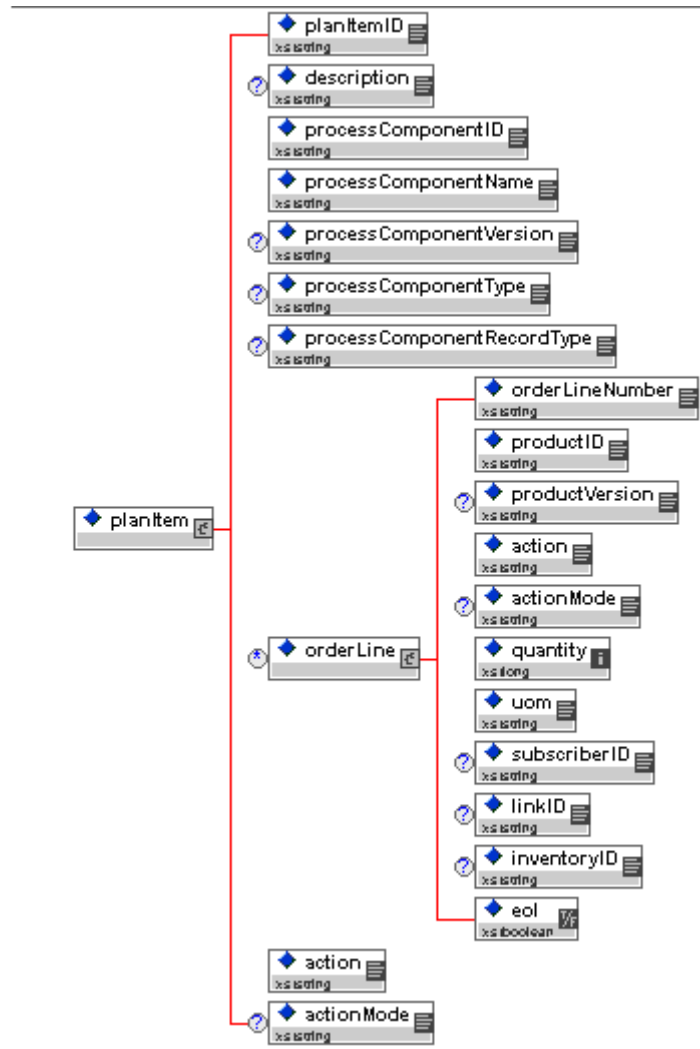
from causing a prolonged JMS Server outage. This helps you to restart the Messaging server from another node.

Schema References

This section covers all schema references for TIBCO Order Management - Long Running.

Plan Item

Plan Item

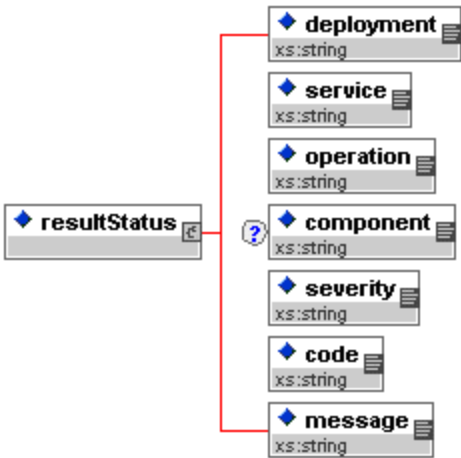


| Element | Type | Cardinality | Description |
|-----------------------------------|--------|-------------|---|
| planItem/planItemID | String | Required | Unique identifier for the plan item within the plan to be executed. |
| planItem/description | String | Optional | Description for the plan item to be executed. |
| planItem/
processComponentID | String | Required | Unique identifier for the Process Component to be executed. |
| planItem/
processComponentName | String | Required | Process component name for the Process Component to be executed. |

| | | | |
|---|---------|----------|--|
| planItem/
processComponentVersion | String | Optional | Process component version for the Process Component to be executed. |
| planItem/
processComponentType | String | Optional | Process component type for the Process Component to be executed. |
| planItem/
processComponentRecord
Type | String | Optional | Class of processComponentType. |
| planItem/orderLine | Type | 1-M | Order line type for the plan item to be executed. |
| planItem/orderLine/
orderLineNumber | String | Required | Order line number for the order line of the plan item to be executed. |
| planItem/orderLine/
productID | String | Required | Product ID for the order line of the plan item to be executed. |
| planItem/orderLine/
productVersion | String | Optional | Product version for the order line of the plan item to be executed. |
| planItem/orderLine/action | String | Required | Order line action for the order line of the plan item to be executed. |
| planItem/orderLine/
actionMode | String | Optional | Order line action mode for the order line of the plan item to be executed. |
| planItem/orderLine/
quantity | Long | Required | Quantity for the order line of the plan item to be executed. |
| planItem/orderLine/uom | String | Required | Unit of measure for the order line of the plan item to be executed. |
| planItem/orderLine/
subscriberID | String | Optional | Subscriber ID for the order line of the plan item to be executed. |
| planItem/orderLine/linkID | String | Optional | Link ID for the order line of the plan item to be executed. |
| planItem/orderLine/
inventoryID | String | Optional | Inventory ID for the order line of the plan item to be executed. |
| planItem/orderLine/eol | Boolean | Required | 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. |
| planItem/action | String | Required | Plan item action for the plan item to be executed. |
| planItem/actionMode | String | Optional | Plan item action mode for the plan item to be executed. |

Result Status

Result Status

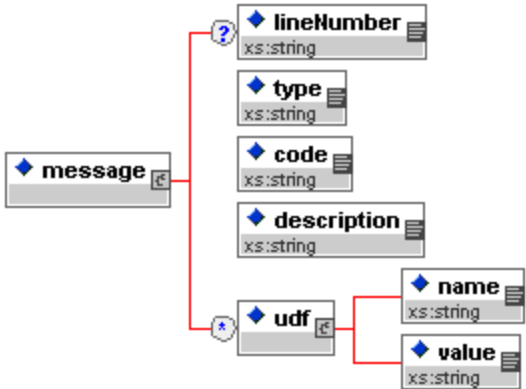


| Element | Type | Cardinality | Description |
|------------|--------|-------------|--|
| deployment | String | Required | Engine deployment that returned this result. |
| service | String | Required | Service name that returned this result |
| operation | String | Required | Operation within the service that returned this result. |
| component | String | Optional | Component within the operation and service that returned this result. |
| severity | String | Required | Severity result. Valid values are:

1. S - Success
2. W - Warning
3. E - Error |
| code | String | Required | Message code for this result. |
| message | String | Required | Message details for this result. |

Message

Message

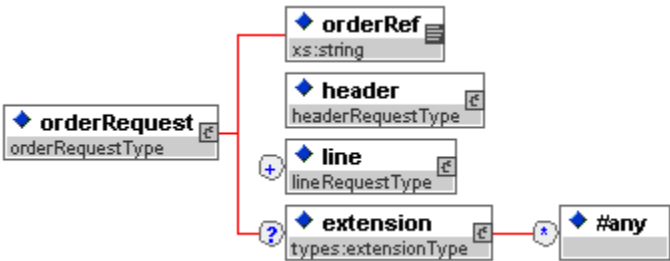


| Element | Type | Cardinality | Description |
|-------------|--------|-------------|---|
| lineNumber | String | Optional | Order line number that this message refers to. |
| type | String | Required | Message type. Valid values are:

1. Information
2. Warning
3. Error |
| Code | String | Required | Message code for this message. |
| Description | String | Required | Message text for this message. |
| udf | Type | 0-M | User defined field type. |
| udf/name | String | Required | User defined field name. |
| udf/value | String | Required | User defined field value. |

Order Request

Order Request

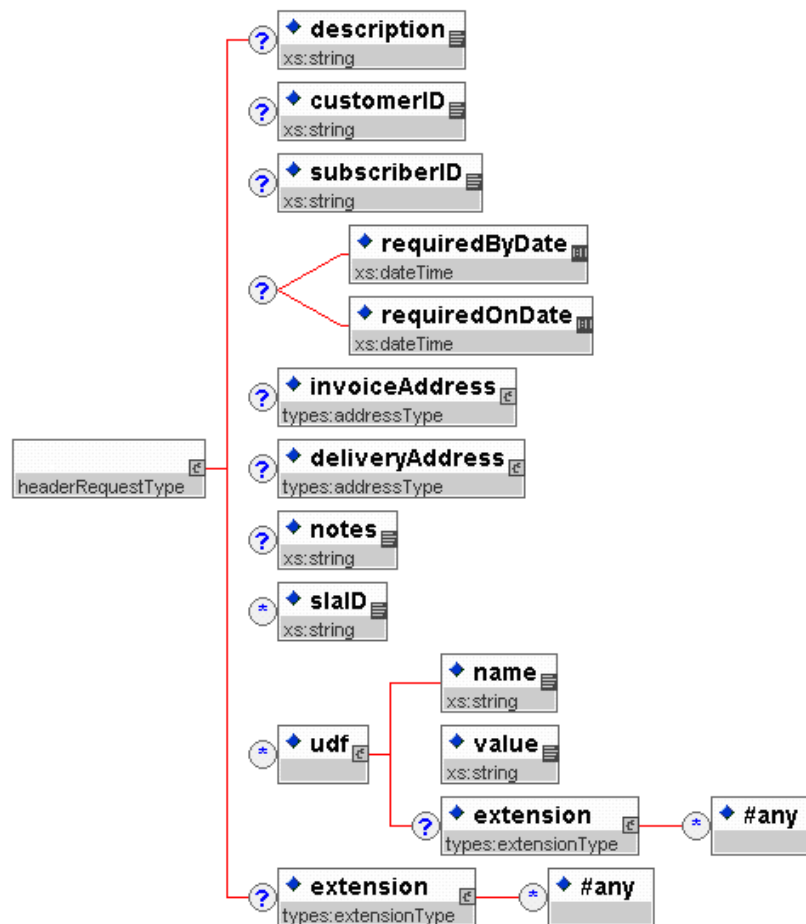


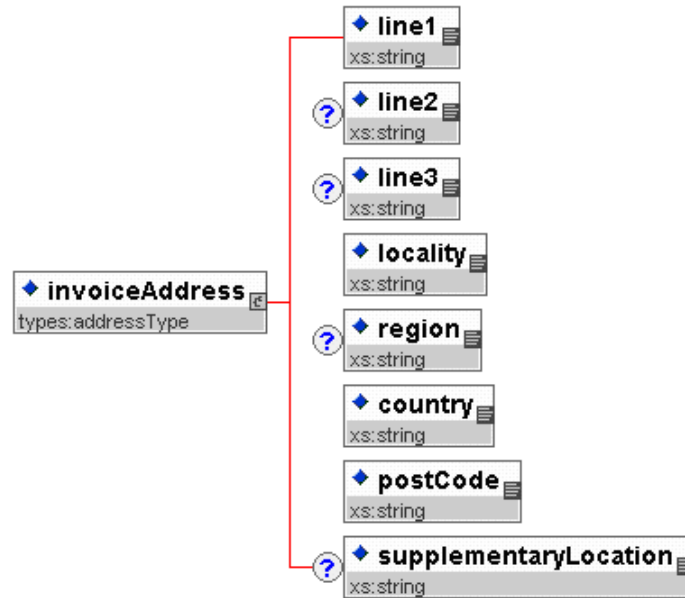
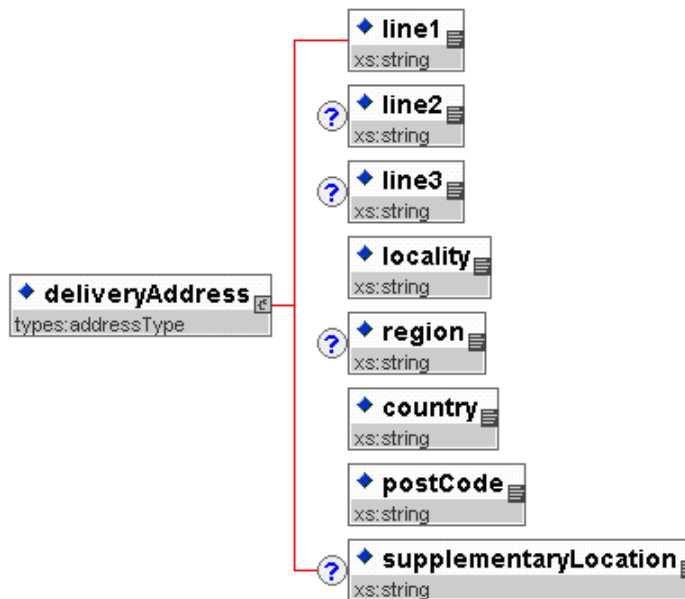
| Element | Type | Cardinality | Description |
|---------|------|-------------|-------------|
|---------|------|-------------|-------------|

| | | | |
|--------------------|--------|----------|--|
| orderRef | String | Required | External unique identifier for an order. |
| header | Type | Required | Order request header type. Refer to the Order Request Header definition for details. |
| line | Type | 1-M | Order request line type. Refer to the Order Request Line definition for details. |
| extension | Type | Optional | Extension attributes for user-defined fields. |
| extension/
#any | Any | Required | Any data |

Order Request Header

Order Request Header



Invoice Address*Delivery Address*

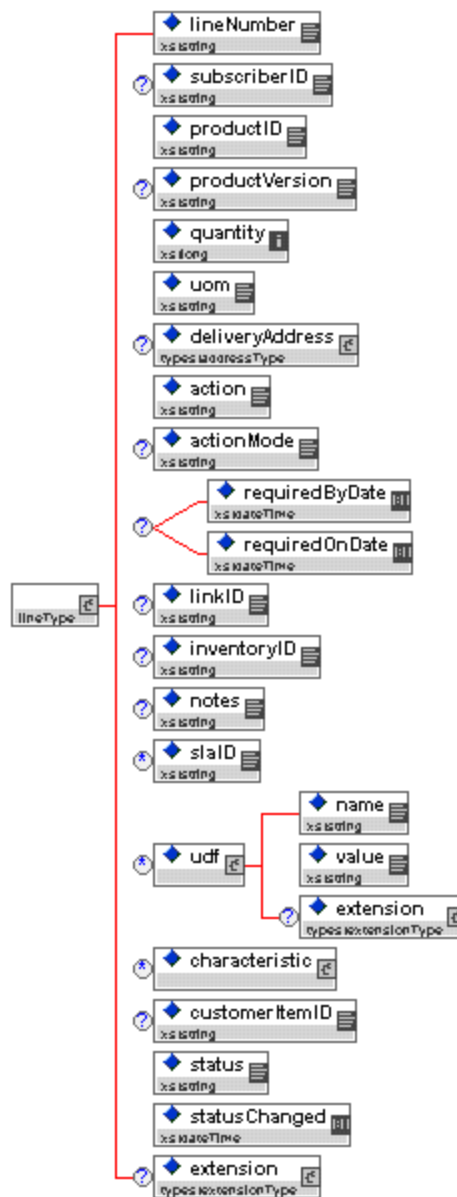
| Element | Type | Cardinalit
y | Description |
|--------------|--------|-----------------|--|
| description | String | Optional | Description for the order. |
| customerID | String | Required | Unique identifier for the customer for this order. |
| subscriberID | String | Required | Unique identifier for the subscriber for this order. |

| | | | |
|--|----------|------------------|---|
| requiredByDate | DateTime | Optional, Choice | Date and time when this order is required to start. |
| requiredOnDate | DateTime | Optional, Choice | Date and time by which this order is required to complete. |
| invoiceAddress | Type | Required | Invoice address type. |
| invoiceAddress/line1 | String | Required | Invoice address line 1. |
| invoiceAddress/line2 | String | Optional | Invoice address line 2. |
| invoiceAddress/line3 | String | Optional | Invoice address line 3. |
| invoiceAddress/locality | String | Required | Invoice address locality. |
| invoiceAddress/region | String | Optional | Invoice address region. |
| invoiceAddress/country | String | Required | Invoice address country. |
| invoiceAddress/postcode | String | Required | Invoice address post code. |
| invoiceAddress/supplementaryLocation | String | Optional | Invoice address supplementary location. |
| deliveryAddress | Type | Required | Delivery address type. |
| deliveryAddress/line1 | String | Required | Delivery address line 1. |
| deliveryAddress/line2 | String | Optional | Delivery address line 2. |
| deliveryAddress/line3 | String | Optional | Delivery address line 3. |
| deliveryAddress/locality | String | Required | Delivery address locality. |
| deliveryAddress/region | String | Optional | Delivery address region. |
| deliveryAddress/country | String | Required | Delivery address country. |
| deliveryAddress/postcode | String | Required | Delivery address post code. |
| delivery address/supplementaryLocation | String | Optional | Delivery address supplementary location. |
| notes | String | Optional | Order notes. |
| slalID | String | 0-M | Unique identifier for an SLA that is applied to this order. |
| udf | Type | 0-M | User defined field type. |
| udf/name | String | Required | User defined field name. |

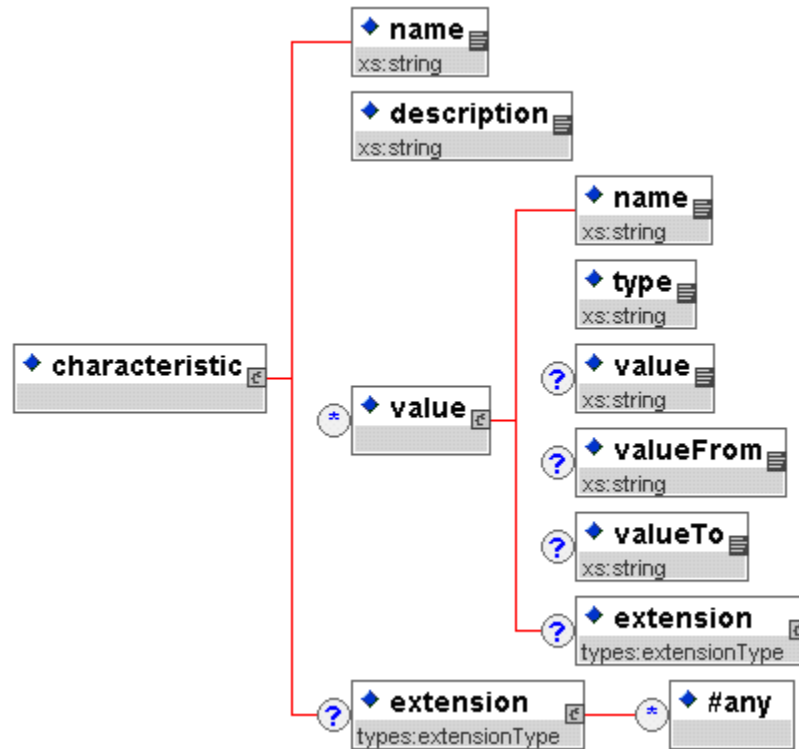
| | | | |
|--------------------|--------|----------|---|
| udf/value | String | Required | User defined field value. |
| udf/extension | Type | Optional | Extension attributes for user-defined fields. |
| udf/extension/#any | Any | Required | Any data |
| extension | Type | Optional | Extension attributes for user-defined fields. |
| extension/#any | Any | Required | Any data |

Order Request Line

Order Request Line



Order Line Characteristics



| Element | Type | Cardinality | Description |
|-----------------------|---------|-------------|---|
| lineNumber | String | Required | Unique identifier for this order line within this order. |
| subscriberID | String | Optional | Unique identifier for the subscriber for this order line. |
| productID | String | Required | Product identifier for this order line. |
| productVersion | String | Optional | Product version for the product for this order line. |
| quantity | Integer | Required | Quantity of the product being ordered. |
| uom | String | Required | Unit of measure of the product being ordered. |
| deliveryAddress | Type | Required | Delivery address type. |
| deliveryAddress/line1 | String | Required | Delivery address line 1. |
| deliveryAddress/line2 | String | Optional | Delivery address line 2. |
| deliveryAddress/line3 | String | Optional | Delivery address line 3. |

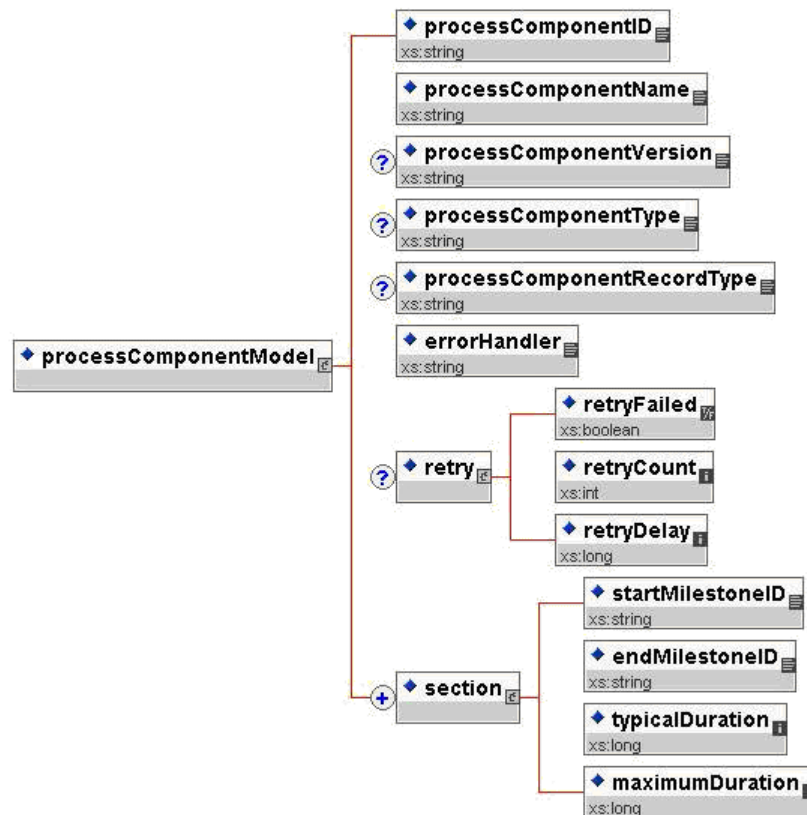
| | | | |
|---------------------------------------|----------|------------------|---|
| deliveryAddress/locality | String | Required | Delivery address locality. |
| deliveryAddress/region | String | Optional | Delivery address region. |
| deliveryAddress/country | String | Required | Delivery address country. |
| deliveryAddress/postcode | String | Required | Delivery address post code. |
| deliveryAddress/supplementaryLocation | String | Optional | Delivery address supplementary location. |
| action | String | Required | Action for this order line.
Valid values are:

1. Provide
2. Update
3. Cease |
| actionMode | String | Optional | Supplementary action mode for the action. |
| requiredByDate | DateTime | Optional, Choice | Date and time by which this order line is required to start. |
| requiredOnDate | DateTime | Optional, Choice | Date and time by which this order line is required to complete. |
| linkID | String | Optional | Unique identifier used to link across order lines on this order. |
| inventoryID | String | Optional | Unique identifier that identifies this order line product with an entry in an external inventory management system. |
| notes | String | Optional | Order line notes. |
| slaID | String | 0-M | Unique identifier for an SLA that is applied to this order line. |
| udf | Type | 0-M | User defined field type. |
| udf/name | String | Required | User defined field name. |
| udf/value | String | Required | User defined field value. |

| | | | |
|-------------------------------------|--------|----------|---|
| udf/extension | Type | Optional | Extension attributes for user-defined fields. |
| udf/extension/#any | Any | Required | Any data |
| characteristic | Type | Required | Characteristic type. |
| characteristic/name | String | Required | Characteristic name. |
| characteristic/description | String | Required | Characteristic description. |
| characteristic/value | Type | 0-M | Characteristic value type. |
| characteristic/value/name | String | Required | Characteristic value name. |
| characteristic/value/type | String | Required | Characteristic value type. |
| characteristic/value/value | String | Optional | Characteristic value. |
| characteristic/value/valueFrom | String | Optional | Characteristic value from. |
| characteristic/value/valueTo | String | Optional | Characteristic value to. |
| characteristic/value/extension | Type | Optional | Extension attributes for user-defined fields. |
| characteristic/value/extension/#any | Any | Required | Any data |
| characteristic/extension | Type | Optional | Extension attributes for user-defined fields. |
| characteristic/extension/#any | Any | Required | Any data |
| customerItemID | String | Optional | Customer item unique identifier. |
| extension | Type | Optional | Extension attributes for user-defined fields. |
| extension/#any | Any | Required | Any data |

Process Component Model

Process Component Model



| Element | Type | Cardinality | Description |
|----------------------------|--------|-------------|--|
| processComponentID | String | Required | Unique identifier for the Process Component to be executed. |
| processComponentName | String | Optional | Process component name for the Process Component to be executed. |
| processComponentVersion | String | Optional | Process component version for the Process Component to be executed. |
| processComponentType | String | Optional | Process component type for the Process Component to be executed. |
| processComponentRecordType | String | Optional | Class of processComponentType. |
| errorHandler | String | Optional | Error handler to use in the event of the Process Component returning an incomplete or unsuccessful execution response message. |

| | | | |
|--------------------------|---------|----------|--|
| retry | Type | Optional | Retry type. |
| retry/retryFailed | Boolean | Required | Flag indicating that Orchestrator might retry failed plan items. |
| retry/retryCount | Integer | Required | Number of times Orchestrator might retry the failed plan item before referring it to Plan Item Failed Handler for manual intervention. |
| retry/retryDelay | Long | Required | Delay in msec between calls to the Process Component if the plan item is retired. |
| section | Type | 1-M | Process component model section type. |
| section/startMilestoneID | String | Required | Unique identifier for the start milestone that describes this section. |
| section/endMilestoneID | String | Required | Unique identifier for the end milestone that describes this section. |
| section/typicalDuration | Long | Required | Typical duration for this section in msec. |
| section/maximumDuration | Long | Required | Maximum duration for this section in msec. |

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO® Order Management Documentation](#) page:

- *TIBCO® Order Management - Long Running Release Notes*
- *TIBCO® Order Management - Long Running Installation and Configuration Guide*
- *TIBCO® Order Management - Long Running Getting Started Guide*
- *TIBCO® Order Management - Long Running Concepts and Architecture Guide*
- *TIBCO® Order Management - Long Running Administration Guide*
- *TIBCO® Order Management - Long Running User's Guide*
- *TIBCO® Order Management - Long Running Web Services Guide*
- *TIBCO® Order Management - Long Running Best Practices Guide*

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

Legal and Third-Party Notices

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 THE LICENSE FILE) 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.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document 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, the TIBCO logo, the TIBCO O logo, ActiveMatrix BusinessWorks, TIBCO Runtime Agent, TIBCO Administrator, and Enterprise Message Service are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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. Please see the readme.txt 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. CLOUD SOFTWARE GROUP, 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.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2010-2023. Cloud Software Group, Inc. All Rights Reserved.