



TIBCO® Order Management - Long Running

User's Guide

Version 5.0.1

April 2022

Document Updated: August 2023



Contents

Figures	11
About this Product	16
Orchestrator	17
Architecture	17
Backing Store	17
Deployment	17
Resource Failure Handling	17
Batch Notification	18
Sub-batching	19
Batch Event Processing	19
Locking Strategy	20
Failover	20
Notification	22
Throttling	24
Configuring the Default Load Capacity	24
Configuring the Individual Load Capacity	24
Configuring the Activation Threshold	25
Formulas	25
Order Request Optimization	26
Sort Criteria	26
Dead Letter Queue	26
External Dependency	29
Enabling Enrichment	30
Configuration	30
Logging	31
Time Dependency	31
Non-Executing Plan Item	31
Process Component Destination	31
Order Types	32
Standard New Order Fulfillment	32
Partially Completed Order Fulfillment	32
Amend Order	33
Suspend and Activate Order	34
Order Submission	35
Synchronous Order Submission	35
Large Plan and Order Orchestration	35

Execution Plan	36
Plan Tasks with Associated Process Components	36
Actions	36
Dependencies	37
Order Header	37
Order Line	38
Orchestrator Interfaces	39
Feasibility Providers	39
Feasibility Request	40
Feasibility Response	41
Process Components	43
Plan Item Execute Request Event	43
Plan Item Milestone Release Request Event	46
Plan Item Milestone Notify Request Event	48
Plan Item Execute Response Event	50
Plan Item Suspend Request Event	52
Plan Item Suspend Response Event	54
Plan Item Activate Request Event	56
Pre-qualification Failed Handlers	59
Pre-Qualification Failed Request Event	60
Pre-qualification Failed Response Event	61
Plan Item External Error Handlers	63
Plan Item Failed Request Event	65
Plan Item Failed Response Event	67
Automated Order Plan Development	69
Overview	69
Architecture	69
Deployment	69
Model Deployment	72
Configuration	73
Main Configuration	73
Logs	74
Integration with Orchestrator	74
Features	75
Autoprovision	75
Dynamic Bundles	76
Static Bundles	77
Time Dependency	77
Product Specification Field Decomposition	77

Custom Action Based Product Decomposition	78
Sequencing	79
Delta Provisioning	82
Single Use	82
Product Affinity (Plan Item Level)	85
Inlink	87
Crosslink	88
Affinity Sequencing	90
Conditional Affinity	92
Conditional Affinity Sample	96
Configurable Handling of CrossLink + ProductComprisedOf Conflicts and Single Use + ProductComprisedOf Conflicts .	98
Sort Plan	98
Attribute-Based Decomposition	99
ProductDependsOn and ProductRequiredFor Relationships	100
Dependent and Sibling Products	104
Shared Attributes	105
Shared Attributes - Sample Test Scenarios	106
Intermediate Milestones Dependencies	107
Milestone to START Dependency	108
END to Milestone Dependency	108
Milestone to Milestone Dependency	108
Milestone without Dependency	110
Conditional Milestones Dependency	110
Order Amendment	111
Amendment Workflow	112
Modeling of the Required Characteristics in Fulfillment Catalog	112
Types of Amendment	113
OrderLine Action Change	114
RequiredbyDate Change	115
OrderLine user-defined field Change	117
OrderPriority Change	120
OrderLine Addition	120
Execution Plan Modification Rules (EPMR)	121
COMPENSATE_RESTART	122
COMPENSATE	124
RESTART	124
IGNORE	125
No Execution Plan Modification Rules Characteristic in Product	125
Amendment Configuration Flags	125

Impact on Dependencies	126
Multiple Amendments	127
Order Priority	127
Understanding Order Priority	128
Order Schema Changes	128
Lower Priority Orders	128
Custom Action	128
Manual Order Plan Development	130
Searching Orders for Manual Order Plan Development	130
Modifying the Plan in Draft Mode through Grid View	131
Create a New Plan Item	131
Create a New Milestone	133
Delete Plan Item	135
Delete Milestone	135
Modify Plan Item	136
Modify Milestones	136
Creating New Dependencies	136
Deleting Dependencies	138
Validations	138
Modifying the Plan in Draft Mode through Gantt View	138
Create a New Plan Item	139
Create a New Milestone	139
Delete Plan Item	139
Delete Milestone	139
Modify Plan Item	139
Modify Milestones	139
Creating New Dependencies	140
Validations	140
Saving the Modified Plan	140
Saving and Executing the Modified Plan	140
Discarding Changes	140
Modifying Plan in EXECUTION State	141
Jeopardy Management System	142
Jeopardy Management System	142
Jeopardy Management	142
Understanding Plan	143
Understanding Critical Path	144
Critical Path Calculation	144
Understanding Dependencies	145

Milestone Dependencies	145
End Milestones	145
Jeopardy Management for Execution Plans	145
Jeopardy Management for Plan Task	146
Must Start On Dependencies	147
Consequential Actions	147
Jeopardy Pre-release Order Processing	147
Predictive Jeopardy	148
Jeopardy Events	148
Plan Item Jeopardy	148
Plan Jeopardy	148
Order Selection for Jeopardy Management	149
Router	150
Internal Error Handler	151
Internal Error Handler Data Flow Diagram	151
Understanding Data Flow in Internal Error Handler	151
Internal Error Handler Sequence Diagram	153
Searching for Plans with Plan Item in ERROR State	153
Modifying the Plan Item State	154
Choosing Error Resolution for the Plan Item in Error State	154
Details of Each Resolution Choice	155
Submit the Error Resolution	156
State Machine Pagination	158
State Machine Pagination Sequence Diagram	158
State Machine Pagination Flow Diagram	159
Order Capture System Overview	160
Order Capture System User Interface Overview	160
Searching for Subscribers	161
Submitting an Order in Order Capture System	162
Amending an Order in Order Capture System	163
Canceling an Order in Order Capture System	164
Order Capture System Error Codes and Messages	164
Search Syntax	166
Order Management Long Running User Interface	167
Navigation	169
Changing Password	170
Order Management Server User Interface Logging Notifications	171
Alert and Confirmation Box	171
Growls for Information and Error Messages	172

Notification Logger	172
Order Management - Long Running Functionality	174
Dashboard	174
Dashboard Components	175
Order Summary	176
Auto-refreshing the Interval	176
Viewing Order Summary Data Based on the Definite Time Period	177
Backlog Orders	177
Amended Orders	177
Setting Display Preferences for Amended Orders	178
Orders in Execution	179
Inflow Orders	179
Jeopardy Dashboard	179
Orders In Jeopardy	180
Jeopardy Live Alerts	180
Jeopardy Recorded Alerts	181
Orders Page	182
Viewing Order Priority	183
Searching for an Order	184
Viewing Order Information	185
Suspending an Order	186
Resuming an Order	187
Canceling an Order	187
Amending an Order	187
Performing Bulk Action on Orders	188
Submit Order and Import XML	190
Adding an Order	190
Plans Page	191
Searching for a Plan	192
Grid View	193
GANTT Chart View	194
Accessing Dependency View of Plan Items and their Status	195
Jeopardy Rule Configuration	196
Plan Item Jeopardy	197
Plan Jeopardy	198
Adding and Configuring Rule	200
Adding Condition to Rule	201
Rule Actions	209
Configure Actions	210

Webservice Configuration	211
Notification Action Configuration	213
Hot Deployment	219
GANTT Chart	219
Jeopardy Management System	220
Viewing a GANTT Chart	220
GANTT Chart Components	221
Top Tool Bar	221
Bottom Tool Bar	221
Grid Header	222
GANTT Chart Diagram	223
GANTT Chart Details	224
Tooltips	225
Activity Log	226
About Activity Log	226
Viewing the Activity Log	227
Searching for an Order in the Activity Log	227
Interpreting the Log Messages	227
Understanding the Types of Log Messages	228
Fulfillment Provisioning Service Order Hierarchy	228
Fulfillment Provisioning Attributes and Parameters	229
Searching for Fulfillment Provisioning Components	230
Third Party Access to Order Management - Long Running Functionality User Interface	231
Implementing OMSUIClient.jar	231
Single URI to Access Order Management Server User Interface Component	231
Data Access Interfaces	233
Get Order	233
Get Order Request	233
Get Order Response	234
Get Order Messages and Message Codes	236
Get Plan	236
Get Plan Request	236
Get Plan Response	238
Get Plan Messages and Message Codes	242
Get Plan Items	243
Get Plan Items Request	243
Get Plan Items Response	246
Get Plan Items Messages and Message Codes	248
Set Plan	249

Set Plan Request	249
Set Plan Response	252
Set Plan Messages and Message Codes	253
Set Plan Item	253
Set Plan Item Request	254
Set Plan Item Response	257
Set Plan Item Messages and Message Codes	257
Get Key Mapping	258
Get Key Mapping Request	258
Get Key Mapping Response	259
Get Key Mapping Messages and Message Codes	261
Best Practices for TIBCO Order Management - Long Running	262
process component Design Guidelines	262
Process Component Technology Selection	262
BusinessWorks - Asynchronous process component	264
BusinessWorks - Synchronous Process Component	271
BusinessEvents - Process Component	271
Exception Handling Guidelines	273
General Approach	274
Example Approach	274
Plan Item Failed Handler	276
Process Component Considerations	276
Pre-Qualification Failed Handler	277
Technical Exception Handling	277
Types of Technical Exception	278
TIBCO Order Management - Long Running Components for Technical Exception Handling	278
Schema References	281
Plan Item	281
ResultStatus	283
Message	284
Order Request	284
Samples	286
Sample Order XML	286
Sample Plan Item XML	287
Sample XPATHs	287
Global Variables	289
Automated Order Plan Development Global Variables	289
Orchestrator Global Variables	293
Global Variables and Configurations	323

TIBCO Documentation and Support Services344

Legal and Third-Party Notices 345

Figures

jConsole UI for Operations to Process Dead Letter Table Entries	19
External Dependency Release request parameters in an XML form	29
Enrichment	30
Partially Completed Order Submission - Existing Fulfillment System	33
Partially Completed Order Submission – Orchestrator Fulfillment System	33
Feasibility Request	40
Feasibility Response	42
Plan Item Execute Request	45
Plan Item Milestone Release Request	47
Plan Item Milestone Notify Request Event	49
Plan Item Execute Response	51
Plan Item Suspend Request	54
Plan Item Suspend Response	55
Plan Item Activate Request	58
Pre-Qualification Failed Request	61
Pre-qualification Failed Response	62
Plan Item Failed Request	66
Plan Item Failed Response	67
Automated Order Plan Development in Collocated Mode	70
Automated Order Plan Development in Standalone Mode (Vertical Scaling)	71
Automated Order Plan Development in Standalone Mode (Horizontal Scaling)	71
Custom Automated Order Plan Development in Standalone Mode (Vertical Scaling)	72
Custom Automated Order Plan Development in Standalone Mode (Horizontal Scaling)	72
Sequencing for the products A, B, and C.	79
Order Plan Execution Sequence	80
Sample Product Model	80
Single use (Provide-Provide)	83
Single use (Provide-Cease)	84
Single use (Provide-Update)	84
Single use (Sequenced)	85
Inlink Affinity	88
Affinity Plan Fragments XSD	89
Parallel Scenario	91
Sequenced Scenario	91
Items in Sequence	92
Conditional Affinity	97
Attribute based decomposition product	99
Data Model	104

END-to-START dependency	107
Milestone to START Dependency	108
END to Milestone Dependency	108
Milestone to Milestone Dependency	108
Start Milestone Dependency	109
Milestone without Dependency	110
Conditional Milestones Dependency	110
Dependency between the compensatory plan item COMP-1_P1 and the existing plan item P1 that is canceled	122
Dependency between the compensatory plan item COMP-2_REDO-1_P1 created during second amendment and the REDO plan item from the last amendment REDO_P1, which is canceled.	123
Dependency between the REDO plan item REDO_P1 and the compensatory plan item COMP-1_P1	123
Dependency between the redo plan item REDO-2_REDO-1_P1 created during second amendment and the REDO plan item from the last amendment REDO_P1, which is canceled	124
Dependency between the REDO plan item REDO_P1 and the existing plan item P1, which is canceled	124
Dependency on plan item P1 in PENDING plan item P2 in the original plan	126
Dependency on the first level REDO plan item of P1 in PENDING plan item P2 in the amendment plan	126
Dependency on plan item P1 in plan item P2 in the original plan	126
Dependency on REDO plan item P1 in REDO plan item P2 in the amendment plan	127
Dependency on plan item P1 in plan item P2 in the original plan	127
Dependency on COMP plan item P2 in COMP plan item P1 in the amendment plan	127
Jeopardy Indications	143
Plan Dependency	144
Plan with Single Execution Path	144
Plan with Multiple Execution Paths	145
Typical and Maximum Durations	146
Jeopardy Risk Region for Plan	147
Router Diagram	150
Order Capture System Architecture	160
Order Management Long Running Login	170
Order Management Long Running Dashboard	170
Change Password	170
Change Password Dialog	171
Order Management Long Running Dashboard	175
Order Summary Section	176
Filtering Data	178
The Orders in Jeopardy Panel	180
Jeopardy Live Alerts	181
Jeopardy Recorded Alerts Panel	182
Order Management System Login	183
Order Management System Dashboard	183
Order Priority	184

.....	188
Select Orders_Bulk Order Action	188
Bulk Actions icon	189
Execute button_Bulk Order Action	189
Plan View	194
Grid View	194
Dependency View	195
Dependency View_direction switch	195
Dependency View_show status switch	196
Dependency View_legend	196
Plan Notification Event	199
Plan Element	199
Adding Rule	200
Rule Attributes	200
Adding Condition to Rule	201
Edit Attribute	201
Condition Builder	202
Utility methods as Left Operand	203
Utility Methods	204
Selecting an Operator	205
Selecting Right Operand	206
Groovy Method	207
Expression	208
Condition Editor	208
Deleting Condition	209
Groovy script generated from a defined condition	209
Delete Rule	209
Suspend Rule	210
Activate Rule	210
The web service request parameters in an XML form	213
Action Attributes	214
Template	216
Template Builder Notifications	217
GANTT Chart Background Colors	224
Section Colors	225
Time Snapshot Line	225
Plan Summary Task Bar tooltip	225
Plan Item Summary Task Bar tooltip	226
Milestone tooltip	226
Section tooltip	226

Typical End Icons tooltip	226
Maximum End Icons tooltip	226
Searching Order	227
Log Message	227
Fulfillment Provisioning Service Order View	229
Fulfillment Provisioning Process Flow	230
Filter Icon	230
Filter Icon	230
Get Order Response	235
Get Plan Response	240
Get Plan Items Request	245
Get Plan Items Response	247
Set Plan Request	251
Set Plan Item Request	256
Get Order Response	260
process component Example - Use Case Activity Diagram	265
process component Example: Receive PlanItemExecutionRequest from Orchestrator	265
process component Example: Send Backend Request	266
process component Example: Backend application call	266
process component Example: BW process receiver	267
process component Example: Set user-defined field for response	268
process component Example: BW ConsumeResponse Process	268
process component Example: Start Specific BW Technical Product Consumer Process	269
process component Example: Example of BW process component Consumer	269
process component Example: Functional Product process component	269
process component Example: Activate Event	270
process component Example: Suspend Event	271
Process Component Example: Simple Synchronous BW component	271
Process Component Life Cycle	272
planItemExecuteRequest (Destination)	272
Arguments for PlanItemExecuteRequestEvent	273
Rule Function Code	273
Edit Body: StateMachine_End_entryAction	273
Example Functional Exception Handling Overview Architecture	275
Example Functional Exception Handling TIBCO Order Management - Long Running Components	276
Technical Exception Handling Architecture Overview	278
Components involved in Technical Exception Handling	279
Process Component Technical Exception Services Overview	280
Plan Item	281
Result Status	283

Message284

Order Request284

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

Orchestrator

This section describes the functions of TIBCO Order Management - Long Running Orchestrator component.

Architecture

Orchestrator is a Java based component, which by default, collocated with Order Management Server. This simplifies the deployment and administration. The Java implementation of the Orchestrator is multi-threaded, which improves the performance.

Orchestrator needs to communicate with components such as Automated Order Plan Development, Feasibility Providers, and Error Handlers. Orchestrator invokes some of these components directly instead of using JMS. Orchestrator can invoke the Automated Order Plan Development component interface directly (if Automated Order Plan Development is in collocated mode as well) or use JMS (if Automated Order Plan Development is in standalone mode).

Backing Store

Orchestrator uses the Order Management Server database to store the state changes for each entity (order, order line, plan, plan item, order amendment). The data for supporting the other features such as clustering, member failover, and recovery is also maintained in the Order Management Server database.

Deployment

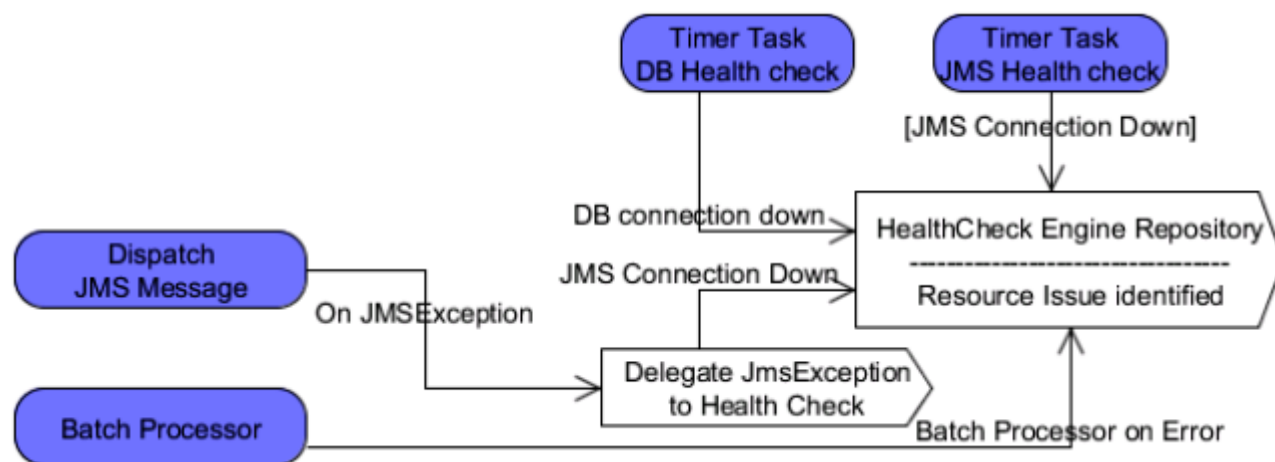
Orchestrator is automatically deployed like Order Management Server is. There are no additional operations to deploy Orchestrator.

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.

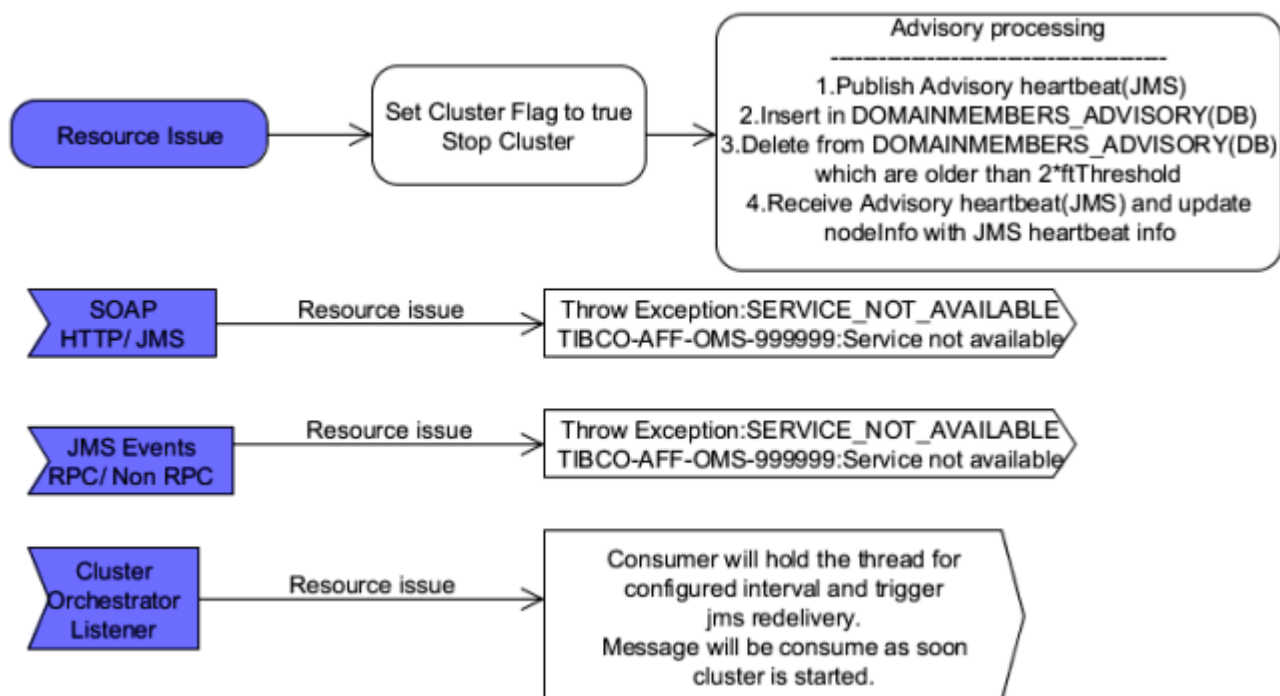
The Resource Failure Handling feature assists in the completion of the processing of previously submitted order without data loss, and in the suspending of the order processing after detecting resource failure.

Resource Failure Handling Architecture

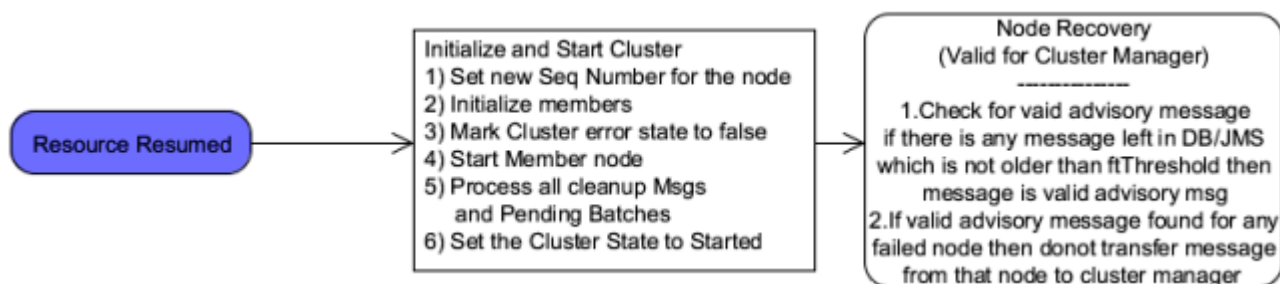


The two timer threads, one for the database and one for the JMS, keeps running at a predefined interval and checks for resource failure. If an exception is identified, then the status of the particular resource is updated to the HealthCheckEngine repository. If an exception is thrown when sending a JMS message by orchestrator then that exception is reported to the HealthCheck thread and the resources are verified for

failure. If a failure is detected then that failure is reported to the HealthCheckEngine repository. The HealthCheckEngine repository is checked by the respective application to take the respective action on resource failure or recovery.



In case of a resource issue, the cluster processing is stopped and the cluster status is changed to INIT. The advisory messages are processed by the Orchestrator for the JMS and the database. The messages are processed by the respective resource that is available. The Orchestrator does not process any requests in case of a resource failure. The SOAP requests over JMS or HTTP returns with the response code to signify resource issue. All JMS listeners of the Orchestrator are running but do not process the messages. The JMS messages are delivered again to the mentioned listeners. Messages on the Transient Data Store interfaces return an error code signifying a resource issue.



In case of resource recovery, the processing of the advisory messages are stopped and the processing of normal heartbeat resumes. The cluster is initialized with all the available members in the cluster using the heartbeat mechanism. Once respective nodes are started, they process the cleanup messages, and the pending batches, before marking the cluster state to the STARTED status. After node status is marked to STARTED, the normal processing starts and messages from JMS destinations are processed.

For more details related to Resource Failure Handling, see the *TIBCO® Order Management Long running Administration Guide*.

Batch Notification

Following are the notifications sent by Orchestrator:

1. Orchestrator sends the JMS notification to the Process Component for executing, suspending, and activating the plan items. Orchestrator also needs to notify the Milestone waiting in the Process components
2. Orchestrator also publishes notifications for the state changes of entities such as Order, Order Line, Order Amendment, Plan, and Plan Item. Third party applications can listen to these notifications.

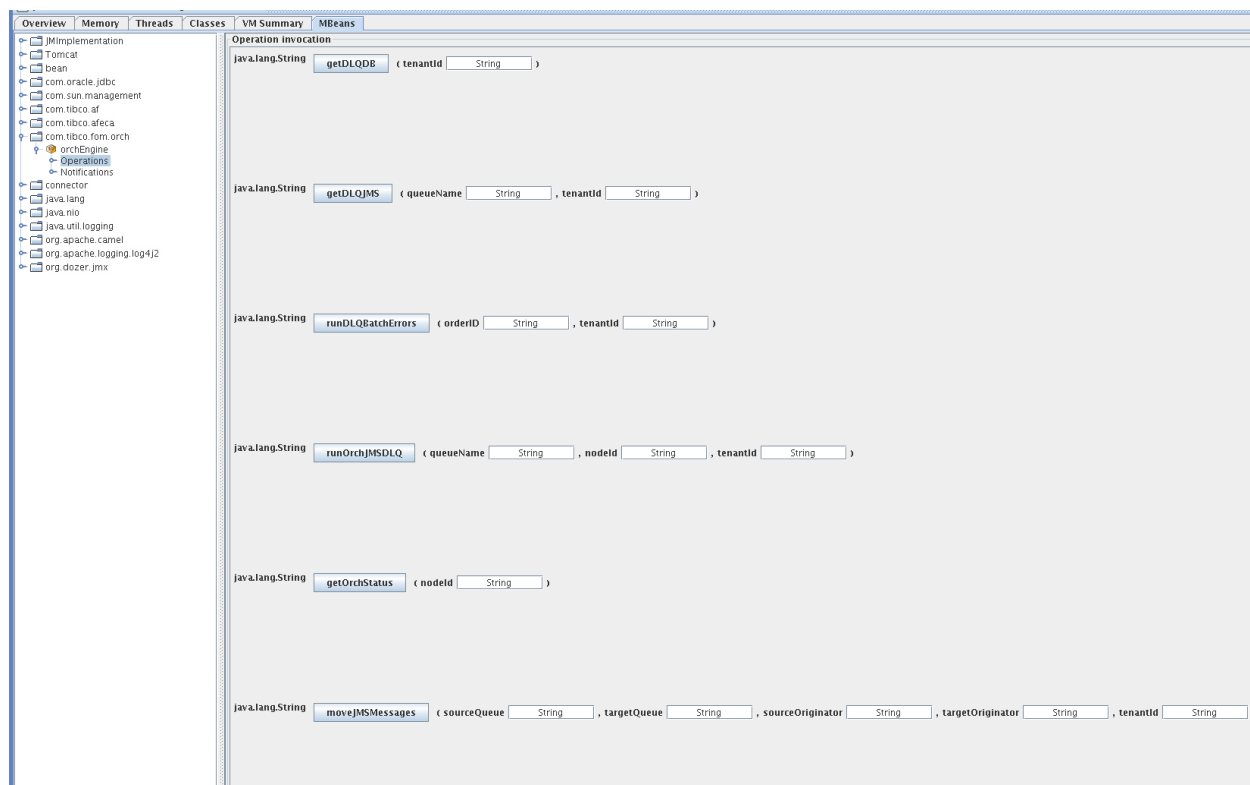
The Orchestrator needs to execute the actions triggered by the state change. The actions are configured internally to dispatch the JMS Messages, process Database notifications and logging. These actions are executed in batches using the batch processor.

Batch threads are configured to run in the Orchestrator. Batch Threads execute the actions with each thread dedicated to a group of orders and batch worker queue. The user can configure the number of batch threads (`com.tibco.fom.orch.batchProcessorsCnt`) and the maximum size (`com.tibco.fom.orch.maxBatchSize`) of the batch to process per thread, depending on the load. Actions generated by order events are posted to the Batch Queue and are processed in the sequence that was sent to Batch queue. Batch threads groups the action execute requests and executes them in batch.

Sub-batching

If the batching is configured, database notifications are batched, executed, and committed to the batch. If the transaction fails, each notification from the batch is executed separately and committed to the database. If it still fails, the order id that is related to the notification is marked as batch error, and the notification is moved to the dead letter table in database. Any other notifications for the same order are moved to dead letter table. Users can execute them manually using the JMX console. The following screenshot is of the jconsole with available operations to process dead letter table entries:

jConsole UI for Operations to Process Dead Letter Table Entries



Batch Event Processing

Events are consumed by state machine and notifications generated by state machines are posted in the Batch process. Following is the list of the events that are processed by Orchestrator:

1. JMS Events that are primarily coming from process components, Order Management Server and clean-up. Clean up activity include clean-up of checkpoints and cleaning the final state orders from local heap memory.

Below is the sequence of activities involved:

- a. State Machine receives the events from JMS.
 - b. Events are consumed by the state machine.
 - c. State machine generates the actions to be executed. The actions are configured internally to dispatch the JMS Messages, process Database notifications and logging
 - d. Action execute requests are posted to Batch Process.
2. Time Dependent Events are triggered using Timer.

Below is the sequence of activities involved:

- a. State Machine receives the events from Timer Event.
- b. State machine generates the actions to be executed. The actions are configured internally to dispatch the JMS Messages, process Database notifications and logging.
- c. Action execute requests are posted to Batch Process.

Locking Strategy

The share nothing architecture is used when processing the orders. Each order is mapped to one node and any subsequent event to the order is processed by the same node. JMS message selector is used for routing orders to the correct nodes. If the event message header does not have the routing information, messages are intercepted and routed to an appropriate node by adding the appropriate message selector.

Failover

The orchestrators can be deployed in a cluster domain. One of the members in the cluster can be cluster manager (CM) and the others act as workers (W). The cluster manager manages the workers in the clusters. Also the cluster manager monitors the members in a cluster, so that it can also process the pending events in failed nodes.

After a node goes down, the cluster manager assigns the load of failed nodes to be backed up by one of the running nodes. This node assignment considers the relative load of all the nodes currently activated in the cluster. The cluster manager fetches information related to how many plan items are being executed by different nodes in the cluster at that time. Based on this information, the cluster manager picks up the least load node to be assigned as a backup for the failed node. The cluster manager instructs this node to start handling the failed node, after which the backup node starts the backup process.

You can place a limit on the number of failed nodes from the cluster that must be backed up. This is controlled by a property with the name `com.tibco.fom.orch.cluster.backUpThreshold`. It defines the percentage of nodes in the cluster whose failure is handled by the cluster manager.

Another property by the name `com.tibco.fom.orch.cluster.backUpTimeout` is used to terminate the backup processing in rare cases, where the backup node has not completed the backup in an appropriate time span. This property is in milliseconds and defaults to one hour.

1. The cluster manager uses the heartbeat generated by the workers to add members to the cluster and also to detect failures.
2. At a given point in time, only one member acts as the cluster manager; the other members act as workers. When the node starts up, a unique sequence number is generated and assigned to each member. This number is also included in the heartbeat published by the members. Eventually all members in the cluster are aware of the sequence number assigned to the other members in the cluster, based on the heartbeat. Each member can identify the lowest sequence numbers and find the corresponding node mapped to it. The member with the lowest sequence number starts as the cluster

manager and other nodes start as workers. When the cluster manager member fails, the member next in sequence starts acting as the cluster manager.

3. The unique sequence number is generated by using Sequence (SEQ_CLUSTER_SEQ_NUMBER) from Order Management Server database.
4. The cluster domain name and members of the cluster are configured in the Order Management Server database. The tables' definitions are as follows:

table DOMAIN

Column	Description
DOMAINID	The name for the cluster domain.
DESCRIPTION	The short description of the domain.
BACKINGSTORE	The backing store (JMS). The JMS used for updating the status of the node.
HEARTBEATINTERVAL	The interval between Heart Beat in milliseconds.
MANAGERACTIVATIONINTERVAL	The interval between cycles in milliseconds to check if the node can be the cluster manager and start executing the cluster manager tasks.
FTTHRESHOLDINTERVAL	The cluster manager that assumes the node is failed if it does not receive the heartbeat from failed node after this specified interval in milliseconds.
HANDLEFAILEDNODEEVENTS	The allowed values are 1 or 0. The default is 1. If set to 1, Orchestrator transfers the events from failed nodes and start processing it. If 0, events from failed nodes are not transferred.

table DOMAINMEMBERS

Column	Description
MEMBERID	The name for the cluster member.
DESCRIPTION	A short description of the member.
DOMAINID	The domain that the member belongs to.
CLUSTERID	The cluster ID is uniquely generated by the node on runtime.
ISCLUSTERMANAGER	The flag is true if the member is cluster manager or it is false if this worker.
SEQNUMBER	The sequence number generated by the member.
LASTUPDATETIMESTAMP	The last updated Timestamp.

Column	Description
STATUS	INIT or STARTED. INIT means the node is initializing. STARTED means the node has started.
HEARTBEATTIMESTAMP	The heartbeat time stamp.
BACKUP_MEMBERID	Holds the value of the member, which is backing up the failed node.
TRANSACTIONID	This field is used by node finder when it allocates the member ID to an instance.
TRANUPDATESTAMP	This field is updated to indicate when the given member ID was allocated.
IS_STATIC	Indicates if the member ID is available for static allocation or dynamic allocation. A value of 1 indicates static and a value of 0 indicates dynamic.

5. The orchestrator restores the data during failure from the checkpoints. The orchestrator supports database check pointing.
6. On cluster failure, the orchestrator's listeners and throttling are disabled. The node joins the cluster again with a new sequence number.

Notification

External clients can listen to JMS notifications about state changes that are sent by the Orchestrator. Users can enable the JMS notifications and configure the JMS destination name. Users can also filter the state change notification.

Type of state change	Property name to enable notification	Property name to filter notification
Order Status Change	c.t.f.o.enableOrderStatusChange	c.t.f.o.order.statusChange.filter
OrderLine Status Change	c.t.f.o.enableOrderLineStatusChange	c.t.f.o.orderLine.statusChange.filter
Plan Status Change	c.t.f.o.enablePlanStatusChange	c.t.f.o.plan.statusChange.filter
PlanItem Status Change	c.t.f.o.enablePlanItemStatusChange	c.t.f.o.planItem.statusChange.filter
Order Amendment Status Change	c.t.f.o.enableOrderAmendmentStatusChange	c.t.f.o.orderAmendment.filter



For readability, the property names in this table use *c.t.f.o* as an abbreviation for *com.tibco.fom.orch*.

To enable these notifications, set the value of respective property to true and define a filter if required. Filter property can be configured as a comma-separated value of more than one status.

Example for the setting filter property for Plan Status change:

```
<ConfValue description="Flag to enable or disable the publishing of Order Status
Change notification message" name="Enable Order Status Change Notification"
propname="com.tibco.fom.orch.enableOrderStatusChange" sinceVersion="2.1"
visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>
<ConfValue description="Filter for Order status change notification; * will publish
all" name="Order status change filter"
propname="com.tibco.fom.orch.order.statusChange.filter" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
<ConfValue description="Flag to enable or disable the publishing of OrderLine Status
Change notification message" name="Enable OrderLine Status Change Notification"
propname="com.tibco.fom.orch.enableOrderLineStatusChange" sinceVersion="2.1"
visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>
<ConfValue description="Filter for OrderLine status change notification; * will
publish all" name="OrderLine status change filter"
propname="com.tibco.fom.orch.orderLine.statusChange.filter" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
<ConfValue description="Flag to enable or disable the publishing of Plan Status Change
notification message" name="Enable Plan Status Change Notification"
propname="com.tibco.fom.orch.enablePlanStatusChange" sinceVersion="2.1"
visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>
<ConfValue description="Filter for Plan status change notification; * will publish
all" name="Plan status change filter"
propname="com.tibco.fom.orch.plan.statusChange.filter" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
<ConfValue description="Flag to enable or disable the publishing of PlanItem Status
Change notification message" name="Enable PlanItem Status Change Notification"
propname="com.tibco.fom.orch.enablePlanItemStatusChange" sinceVersion="2.1"
visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>
<ConfValue description="Filter for PlanItem status change notification; * will publish
all" name="PlanItem status change filter"
propname="com.tibco.fom.orch.planItem.statusChange.filter" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
<ConfValue description="Flag to enable or disable the publishing of OrderAmendment
Status Change notification message" name="Enable Order Amendment Status Change
Notification" propname="com.tibco.fom.orch.enableOrderAmendmentStatusChange"
sinceVersion="2.1" visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>
<ConfValue description="Filter for OrderAmendment status change notification; * will
publish all" name="Order Amendment status change filter"
propname="com.tibco.fom.orch.orderAmendment.filter" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
<ConfValue description="Flag to enable or disable the publishing of Plan development
notification message" name="Enable Plan development notification"
propname="com.tibco.fom.orch.enablePlanDevelopmentNotification" sinceVersion="2.1"
visibility="Basic">
```

```
<ConfBool default="false" value="false" />
</ConfValue>
```

Throttling

Throttling controls the number of messages that Orchestrator can consume and that can be dispatched to the process component. If Orchestrator dispatches more messages than the process components load capacity, then the process component becomes a bottleneck. The load capacity property helps to define a limit of Total Count of PlanItems in an execution status; when this value is exceeded, the process of outgoing messages is suspended.

You can control the number of incoming messages Orchestrator consumes at any given point of time by configuring the number of order receivers and process component receivers. For the outgoing messages sent to the process components, you can configure the default load capacity of the process component. The default load capacity indicates how many requests can be accepted by the process component. Orchestrator stops consuming new orders when the number of process component requests that are not responded to exceeds the default load capacity.

You can also configure the load capacity for individual process components. This property overrides the default load capacity. You can configure the activation threshold in Percentage of load capacity so that the node can enable the listeners when the number of pending process component requests is less than this configured value.

Configuring the Default Load Capacity

This configuration applies to all the nodes (Members) deployed. The value defined for this property applies to all the process components that are not included in the individual load capacity configuration (for example, Default Load Capacity =3, then Process Component1=3, Process Component2=3, and so on).

Procedure

1. Navigate to **Order Management System > FOM Order Management System > Orchestrator Configuration > Generic Configuration**.
2. Enable **Throttling** and define **Default Load Capacity** and **Activation Threshold Percentage**.
3. Restart the service.

Configuring the Individual Load Capacity

This configuration applies to all the nodes (Members) deployed, but you can individually define the load capacity per each process component.



If you do not include all the process components, then those process components use the value from the Default Load Capacity configuration.

Procedure

- Open the file `$OM_HOME/roles/configurator/standalone/config/ProcessComponent.props` and add the following two properties per each process component:

```
<Process Component ID>.destination.name=<Queue Name>
```

```
<Process Component ID>.loadcapacity=<Number>
```

- Process Component ID = PlanFragment ID
- Queue Name = Queue name for the PlanItemExecuteRequest
- Event Number = Load capacity for this process component

Example:

```
NameSimplePCProvide.destination.name=tibco.aff.orchestrator.planItem.NameSimplePCProvide.execute.request NameSimplePCProvide.loadCapacity=7
```

Configuring the Activation Threshold

This property allows you to define a percentage of when the process of outgoing messages to the process components can be resumed.

Procedure

1. Navigate to **Order Management System > FOM Order Management System > Orchestrator Configuration > Generic Configuration**.
2. Define the **Activation Threshold Percentage**.

Formulas

Be aware of the following formulas to identify when a process of outgoing messages are suspended or resumed.

Using Default Load Capacity

- Node (Member) Load Capacity: $MLC = DLC * PCN / MRUN$
 - DLC = default load capacity
 - PCN = Number of different PC in execution status (Example: PC1 + PC2 = 2).
 - MRUN = sum of running Members.
- Node (Member) Activation Threshold: $ACT = MLC * PER$
 - PER = Activation Threshold as a % of Load Capacity for throttling
- Suspend Condition: $TC > MLC$
 - TC = Total Count of PlanItems in execution status.
- Resume Condition: $TC < ACT$
 - TC = Total Count of PlanItems in execution status.

Using Individual Load Capacity

- Node (Member) Load Capacity: $MLC = SLC / MRUN$
 - SLC = sum of PC load capacity (Using the property file)
 - MRUN = sum of Members that are only running
- Node (Member) Activation Threshold: $ACT = MLC * PER$
 - PER = Activation Threshold as a % of Load Capacity for throttling
- Suspend Condition per Member: $TCM > MLC$
 - TCM = Total Count of PlanItems in execution status for a Member
- Resume Condition: $TCM < ACT$
 - TCM = Total Count of PlanItems in execution status for a Member

Mixing Default and Individual Load Capacity

- Node (Member) Load Capacity: $MLC = SLC / MRUN$
 - SLC = sum of PC load capacity (Using the Property file) + (number of unregistered PC*Default Load Capacity)
 - Unregistered PC = PC's that are not included in the Property File, those PC's use the Default Load Capacity
 - $MRUN$ = sum of Members that are only running
- Node (Member) Activation Threshold: $ACT = MLC * PER$
 - PER = Activation Threshold as a % of Load Capacity for throttling
- Suspend Condition per Member: $TCM > MLC$
 - TCM = Total Count of PlanItems in execution status for a Member
- Resume Condition: $TCM < ACT$
 - TCM = Total Count of PlanItems in execution status for a Member

Order Request Optimization

The Router sends the order request to the Orchestrator with the payload. If the Orchestrator is throttled to not pick more messages from the router, the message size grows with the load. Enterprise Message Service Backing store grows when you have huge payload in the order request. User can enable a flag so the payload is truncated from the message. The message contains only the order id in JMS Properties. The node that picks up this message sends query to the database to get the corresponding payload, which needs to be processed.

```
<ConfValue description="Flag to control whether Orchestrator must receive only the
reference or complete xml payload in the submit order request message from omsServer"
isHotDeployable="true" name="Receive OrderRequest by Reference"
propname="com.tibco.fom.orch.receiveOrderRequestByReference" readonly="false"
sinceVersion="2.1" visibility="Basic">
  <ConfBool default="true" value="true" />
</ConfValue>
```

Sort Criteria

Use the sort criteria parameter to get the orders sorted by the submitted date in ascending or descending order.

The following example shows how to configure the get orders request to respond with 15 orders ascending by the submitted date.

```
<ord:GetOrdersRequest ExternalBusinessTransactionID="123">
  <!--Optional:-->
  <ord:sortCriteria>
    <!--Zero or more repetitions:-->
    <ord:sortField sortBy="ASC" sortSequence="1"> <!--SortBy can be "ASC" or
"DESC" -->
      <ord:field>submittedDate</ord:field>
    </ord:sortField>
  </ord:sortCriteria>
  <ord:count>15</ord:count> <!--to restrict the number of orders in result -->
</ord:GetOrdersRequest>
```

Dead Letter Queue

When a JMS message reaches the maximum retry defined; it is sent to a dead queue configured with the respective route. In case of time dependencies, if the time dependency is not executed within the predefined

retry count, it is saved into database and is not picked by Orchestrator. This can be still triggered from JConsole using JMX interface.

Following are the cases when the dead letter queue is used:

1. JMS Event fails after max retries: JMS messages are routed to Dead letter Queue in JMS. This can be viewed using JMS Clients. The user can replay it by moving the dead letter queue messages back to source queue events or by using JMX operations provided to move them to source queue.
2. Timer Events for Time dependent plan items fails after max retries: Time Events that fail after the max number of retries are moved to dead letter table in database.
3. Batching Errors: If the batching is configured, database Notifications are batched and executed and committed in batch. If the transaction fails, each notification from the batch is executed separately and committed in the database. If it still fails, order id that is related to the notification is marked as batch error and notification is moved to dead letter table in database. Any other notifications for the same order are moved to the dead letter table. User can execute them manually using JMX console.
4. Database table structure for dead letter queue:

DEAD_LETTER table

Column	Description
SID	ID
DEADLETTERID	Dead Letter ID (Any reference based on the dead letter type) For Time dependency and Batch, its order id.
DEADLETTERTYPE	Allowed Values are TIME_DEPENDENCY and BATCH.
VALUE	Data blob.
LASTSTATUSCHANGE	Timestamp when record was last updated.

5. Any JMS events on those orders before manually executing these requests are moved to JMS dead letter queue.
6. Any Timer events on those orders before manually executing these requests are moved to DB dead letter queue.
7. Following JMX operations are exposed to the user for viewing and executing the dead letter queue.

Operation	Arguments	Expected Results
getDLQDB	A string value indicates dead letter type. Allowed value is Tenantid. It can be an empty string.	Displays list of all dead letters according to the input parameter. If input is empty string; it lists out all the dead letters including batch errors and time events.
getDLQJMS	A string value indicates dead queue name or it can be an empty string. Allowed values: queueName and tenantid.	Displays list of messages pending on respective dead queue provided as input. If input is empty string; it lists out all the pending messages on all Orchestrator related dead queues.

Operation	Arguments	Expected Results
runDLQBatchErrors	A string value indicates orderID or it can be an empty string. Allowed values: orderid and tenantid.	Executes further processing of respective orderID. If input is an empty string, it processes all orderIDs available in dead letter.
runOrchJMSDLQ	A string value indicates dead queue name or it can be an empty string. Allowed values: queuename, nodeid and tenantid.	Processes all messages pending on respective dead queue. If input is an empty string, it processes messages pending on all dead queues.
getOrchStatus	The node ID that you want the status for.	Returns the status for a given node ID indicating if the node is in INIT or STARTED state.
moveJMSMessages	<p>sourceQueue - The queue from which messages have to be moved.</p> <p>targetQueue - The target queue where they are copied.</p> <p>sourceOriginator - Filters messages from sourceQueue based on originator value.</p> <p>targetOriginator - Indicates the new value of originator header for the messages that are moved.</p> <p>tenantId - Indicates the tenant whose messages have to be moved.</p>	Moves messages from a source to a target queue and also changes the value of the originator header if indicated by the caller.

The following dead queues are supported by JMX operations to move back the messages to its source destination:

Dead Queue Name	Description
tibco.aff.orchestrator.order.submit.dead	Submit Order Request dead queue
tibco.aff.orchestrator.planItem.execute.reply.dead	PlanItem execution response dead queue
tibco.aff.orchestrator.planItem.milestone.notify.request.dead	MilestoneNotifyRequest from process components to Orchestrator dead queue
tibco.aff.orchestrator.order.withdraw.dead	Withdraw Order dead queue
tibco.aff.orchestrator.order.activate.dead	Activate order request dead queue
tibco.aff.orchestrator.planItem.suspend.reply.dead	PlanItem suspend response dead queue
tibco.aff.orchestrator.order.suspend.dead	Suspend order request dead queue

Dead Queue Name	Description
tibco.aff.orchestrator.cache.cleanup.dead	Cache cleanup dead queue
tibco.aff.orchestrator.provider.order.feasibility.reply.dead	External Feasibility reply dead queue
tibco.aff.orchestrator.provider.order.prequal.failed.reply.dead	External PreQualificationFailed reply dead queue
tibco.aff.orchestrator.provider.planItem.failed.reply.dead	PlanItem error handler response dead queue
tibco.aff.orchestrator.planItem.externalDependency	PlanItem External Dependency Release
release.request.dead	Request dead queue
tibco.aff.orchestrator.cache.addEvent.dead	Time dependency add event dead queue
com.tibco.fom.oms.afi.orch.opdresponse.sender.dead.queue	OPDResponse to Orchestrator dead queue
com.tibco.af.oms.orderService.amendment.task.ack.dead.queue	Order amendment acknowledgment task dead queue

External Dependency

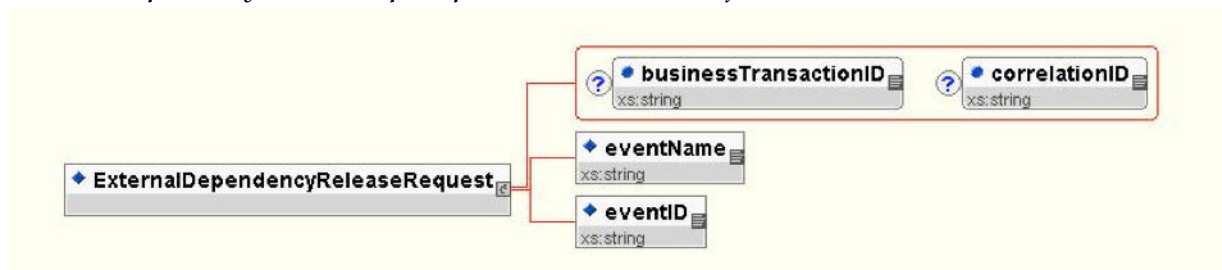
External dependency is a dependency type that waits for notification from an external system by an event before it is satisfied. This dependency is specified by two attributes namely:

1. **event name:** Name of the event that satisfies this dependency.
2. **event id:** Unique identifier of the event name that satisfies this dependency.

The plan development/enrichment module, which adds an external dependency to the milestone of a plan item has to ensure that the combination of event name and event id is unique across all the orders. A unique dependency id is generated according to the two attributes mentioned above and saved in Order Management Server database.

To satisfy a particular dependency, an external dependency request has to be sent by the process component to Orchestrator using the following schema on the queue `tibco.aff.orchestrator.planItem.externalDependency.release.request`. The process component might choose to add 'originator' as a JMS message property with value as 'NodeID' if known to indicate to Order Management Server the node from which this order is being processed.

External Dependency Release request parameters in an XML form

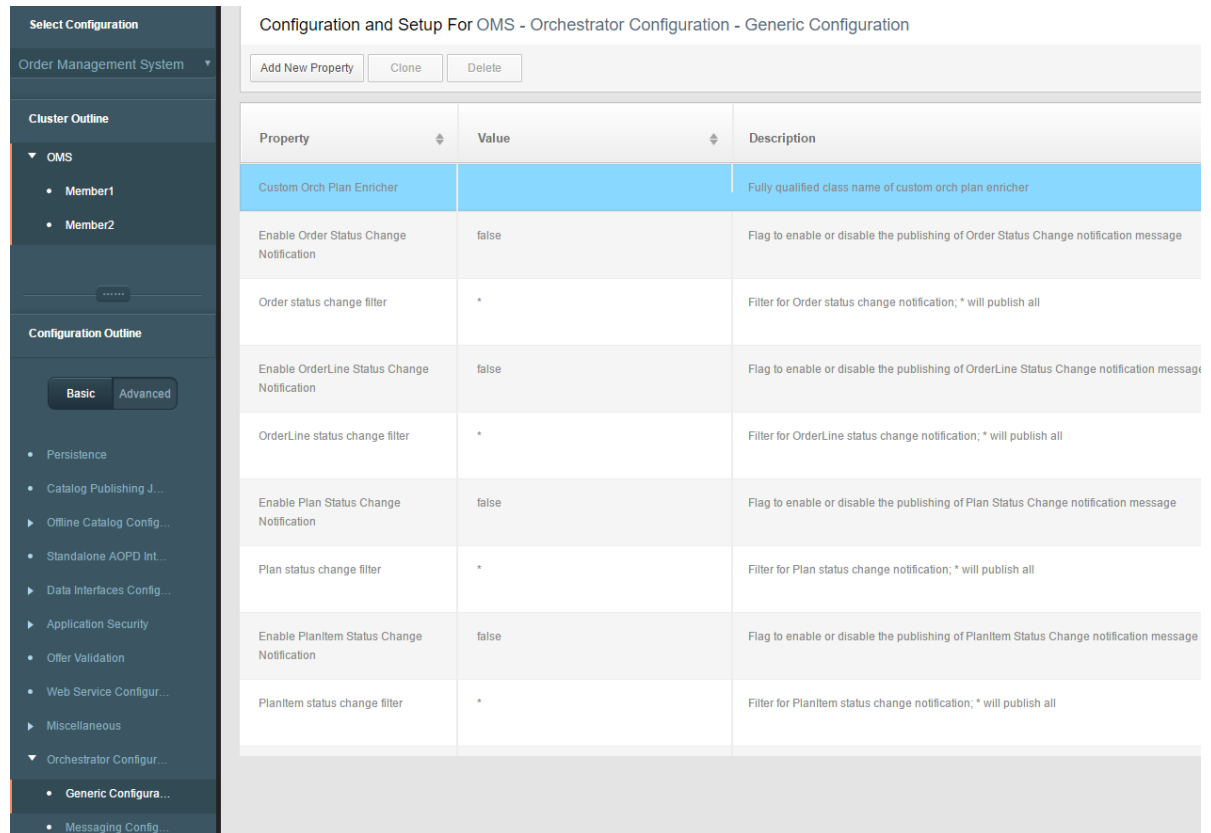


A hook has been provided to enrich the plan returned by OPD by adding external dependencies if needed. This can be achieved by implementing an interface `com.tibco.aff.oms.server.jms.orch.custom.OrchestratorPlanEnricher` present in `omsCommon-1.0.jar` located in `$OM_HOME/lib`. The interface defines a single method `enrichPlan()`, which needs to be implemented. The implementation must be compiled with `omsCommon-1.0.jar` in the classpath. The compiled dependency must be added in the lib directory of the `omsServer` service, and the fully qualified name of the implementer class must be added to the `com.tibco.fom.orch.enrich.CustomOrchPlanEnricher` property in Configurator. To complete the process, restart the `omsServer` service.

Enabling Enrichment

1. Create a new java project in IDE with `omsCommon-1.0.jar` in build path.
2. Implement an interface with name `com.tibco.aff.oms.server.jms.orch.custom.OrchestratorPlanEnricher`.
3. Copy the .jar file to `$OM_HOME/roles/omsServer/standalone/lib`.
4. Configure the implemented class name through Configurator.

Enrichment



Configuration and Setup For OMS - Orchestrator Configuration - Generic Configuration

Property	Value	Description
Custom Orch Plan Enricher		Fully qualified class name of custom orch plan enricher
Enable Order Status Change Notification	false	Flag to enable or disable the publishing of Order Status Change notification message
Order status change filter	*	Filter for Order status change notification; * will publish all
Enable OrderLine Status Change Notification	false	Flag to enable or disable the publishing of OrderLine Status Change notification message
OrderLine status change filter	*	Filter for OrderLine status change notification; * will publish all
Enable Plan Status Change Notification	false	Flag to enable or disable the publishing of Plan Status Change notification message
Plan status change filter	*	Filter for Plan status change notification; * will publish all
Enable PlanItem Status Change Notification	false	Flag to enable or disable the publishing of PlanItem Status Change notification message
PlanItem status change filter	*	Filter for PlanItem status change notification; * will publish all

By default, a blank value is present in the Configurator, which indicates no enrichment.

5. Redeploy Order Management Server.



A sample do-nothing implementation is included with distribution in `omsCommon-1.0.jar` present at `$OM_HOME/lib` with the name `com.tibco.aff.oms.server.jms.orch.custom.CustomOrchestratorPlanEnricher`.

Configuration

There are configurations that can be modified to change the defaults for external dependencies.

1. The queue on which external dependency release request must be sent.
2. The number of receivers on the external dependency release request queue.
3. PlanItem External Dependency Release Request dead queue.

Logging

1. Successful integration with enrichment class

```
INFO : Sending plan of ORDER with orderID {} for enrichment to custom class {}.
```

2. Class configured through Configurator found but invalid implementation

```
INFO : Custom class {} is not a valid implementation of OrchestratorPlanEnricher.
Skipping plan enrichment for orderID {}.
```

3. Class configured through Configurator not found.

```
INFO : Class with name [{}] supplied for custom plan enrichment not found in
CLASSPATH. Skipping plan enrichment for orderID {}.
```

Time Dependency

Time dependency in plan items is satisfied when a certain time period has elapsed, or a certain absolute date and time has been reached. Time dependencies take the form of an absolute date time and once the time has reached or passed, then the dependency is considered satisfied.

Time dependencies of a planItem are scheduled to be EXECUTED at the specified absolute time and only executed once the time is reached. If execution fails then the Orchestrator tries to execute it until maximum retries (*com.tibco.fom.orch.timeDependency.maxRetryCount*) is reached. If it fails during max retries then the Orchestrator puts the order into DEAD LETTER for future reference and the time dependencies are not scheduled and not executed by the Orchestrator.

Non-Executing Plan Item

A non-executing plan item does not have to be submitted to a process component service.

A comma-separated string of planFragment IDs is defined with property name *com.tibco.fom.orch.nonexecuting.planfragmentID*, which indicates a planItem having these PlanFragments is treated as non-executing planItem.

```
<ConfValue description="ID of the plan fragment to be assigned for Non Executing plan
items" name="Non Executing Planfragment ID"
propname="com.tibco.fom.orch.nonexecuting.planfragmentID" sinceVersion="2.1"
visibility="Advanced">
  <ConfString default="NON_EXECUTING" value="NON_EXECUTING" />
</ConfValue>
```

The Orchestrator does not send any notification to Process Component service and it completes the planItem along with its milestone dependencies.

Process Component Destination

Currently, process component notifications are sent to a default destination if there is no owner defined in the process component or to a destination with owner name if owner is defined in process component. This destination can be overridden and a new destination can be used as destination for process component notifications. This can be configured by using property *com.tibco.fom.orch.overridePlanfragmentDestination*. If this property is set to true then the destination is picked from *ProcessComponent.props* file for respective process component ID. Content of this property file is loaded when file is modified and the updated value

is used for sending notifications. This is applicable only when the error handler type is internal. This properties file has destination defined for process component as follows:

```
<Process Component ID>.destinationName=<Destination value>
```



The Destination value is a String.

If there is no destination defined for a process component in file *ProcessComponent.props* or property *com.tibco.fom.orch.overridePlanfragmentDestination* is configured as *false* then default behavior is used and all the process component related notifications are sent to a predefined queue. If the property is *true* and the value is not configured, the default destination is used. The default value is *false*.

For an alternate process component, you have to configure *<Process Component ID>* and *<Alternate Process Component ID>* in *ProcessComponent.prop* file.

<Process Component ID>.alternateProcessComponent=<Alternate Process Component ID> and *com.tibco.fom.orch.overridePlanfragmentDestination* property must be *true*.

You can track the change details in **Activity Log** on Order Management Server UI.

Order Types

TIBCO Order Management - Long Running supports the following order fulfillment modes:

- Standard New Order Management
- Partially Completed Order Fulfillment
- Amend Order
- Cancel Order
- Suspend Order
- Activate Order

Standard New Order Fulfillment

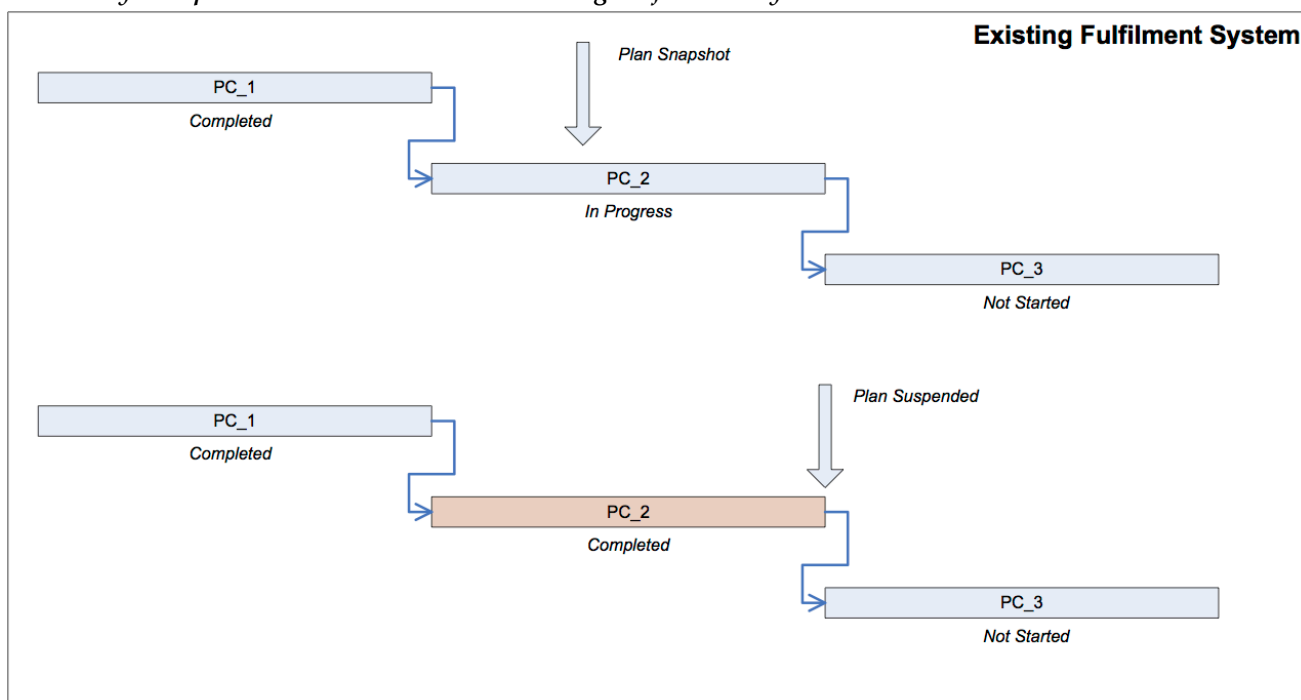
Standard new order fulfillment is the process of submitting an order for orchestration, which creates a new execution plan and orchestrate it to completion. This is the normal mode of operation for submitting new orders to the system.

Partially Completed Order Fulfillment

Partially completed order fulfillment works in a similar manner, except as part of the plan development step certain plan items are indicated as having previously been completed. The new orchestration is created based on this information. This allows migration of partially completed orders from other fulfillment systems into Orchestrator.

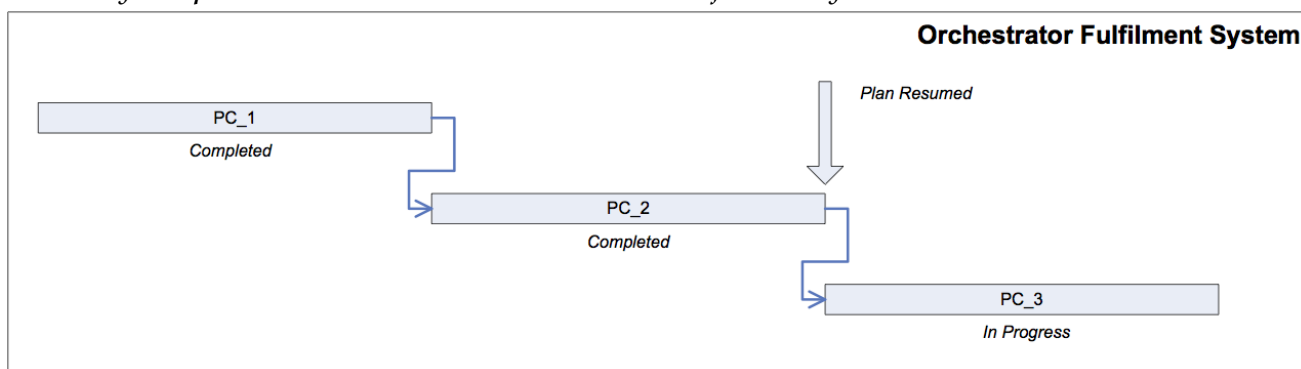
In order for a partial order fulfillment to work, the previous fulfillment system must have suspended execution of the plan at a location that allows for migration. This means pushing orders through to completion of the currently in progress process component, and then suspending the order once all current in progress tasks have been completed. A sample scenario is outlined below:

Partially Completed Order Submission - Existing Fulfillment System



In the existing fulfillment system a plan is in execution. The plan snapshot occurs when process component PC_2 is in progress. This plan is in an inconsistent state and cannot be migrated. Therefore the in progress process component must be pushed through to completion before migrating the order. At this point PC_2 is completed and at this point the order is suspended before PC_3 begins.

Partially Completed Order Submission – Orchestrator Fulfillment System



The partially completed order is submitted to Orchestrator just as a new order. During the callout for plan development a migration component analyzes the order and determines if it is partially completed or not. If it's a new order, it is processed normally. If it's partially completed then a partial execution plan is created and returned to Orchestrator. Orchestrator sets the status of previously completed plan items to complete so that they do not execute again. Orchestration continues at the next steps in the execution plan when the plan is resumed.

Amend Order

An order amendment is the process of making changes to a previously submitted order. An order can be amended by sending through the new order with the same orderID and orderRef as the previously submitted order.

Orders might only be amended in certain lifecycle states.

Amendable	Not Amendable
<ul style="list-style-type: none"> Submitted Feasibility Plan Development Pre-Qualification Failed Execution Suspended START Error-Handler 	<ul style="list-style-type: none"> Complete Canceled Withdrawn

Amendments prior to creation of a plan take the form of updating the order in the database and then restarting the order lifecycle back from Submitted. At this point a plan has not been generated and does not have to be modified. When the fulfillment process reaches the Plan Development step, the updated order as it exists from the amendment is used to generate the plan. This applies to order status of Submitted, Feasibility, Plan Development, and Pre-Qualification Failed.

For amendments that occur after a plan has been created, but when the plan is still Pending, then the order is updated in the database, the existing plan is discarded and the order starts back from Submitted. This applies to order status of Execution, but with a plan status of Pending. However, this is a very rare scenario as the plan immediately goes into EXECUTION state.

For amendments that occur after a plan has started executing only certain aspects of an order might be amended. These are outlined below. Any other aspects of an order not explicitly detailed here are not amendable. This applies to order and plan status of Execution.

There is no limitation on the number of amendments that are possible for any given order, but only one amendment might be active at any one time. Once the amendment has completed, and the order resumes execution then it is possible to amend the order again.

If the order goes to Pre-Qualification Failed state from Order Management Server, it cannot be amended.

If an order is amended in Pre-Qualification Failed state, the action of order line in the amendment order request cannot be CANCEL.

For more information, see [Order Amendment](#).

Suspend and Activate Order

The order can be suspended at any point during the fulfillment process.

1. If the order is in any of the pre-EXECUTION states, it is suspended immediately.
2. If the order is in EXECUTION state, Orchestrator sends the suspend requests to all the process components associated to the plan items that are in execution state. These process components might either respond with an execution suspend response, if they can suspend the processing or execution complete response, if they can't. Based on the response, the executing plan items go into SUSPENDED or COMPLETE state. Finally, order and plan state is changed to SUSPENDED.
3. The orders that are in final states such as COMPLETE or CANCELLED or already in SUSPENDED state cannot be suspended again.

The suspended orders can be activated back into the EXECUTION state so as to proceed ahead with the fulfillment.

1. If the order was in any of the pre-EXECUTION states before suspension, it is activated immediately and processing carry on further.

2. If the order was in EXECUTION state before suspension, Orchestrator activates it by sending the activation requests for all the process components associated with the plan items that were SUSPENDED. Finally, order and plan state is changed to EXECUTION.

Order Submission

A customer order is received from an external order capture or request injection system, for example, a CRM system or a Business-to-Business (B2B) gateway, and Web services. The order must be in the XML format, must conform to the order schema, and is received through a Java Message Service (JMS) message.

Synchronous Order Submission

This feature allows you to synchronously submit the order and receive response on completion of order fulfillment. This feature is useful when order fulfillment is a short-running process.

The basic function of synchronous submit order web service is to accept a new order request, and return the response to the calling application after the order has been completed through Orchestrator. The order is stored in Order Management Server and then sent to Orchestrator, which on completion (or failure) of order responds back. This response is then shown to the calling application. This is different from the Submit Order Service, which accepts the new order request, stores the order in Order Management Server, sends the order to orchestrator, and returns the response back to the calling application with the order status as *submitted* without waiting for Orchestrator to respond back.

Large Plan and Order Orchestration

The overall orchestration process and state machine design has been modified to enable the orchestrator to receive large orders and orchestrate the corresponding large plans to completion. An example of a large order is an order with 1300 order lines and four plan items getting generated from each order line.

Previously, the state machine was generated in a single thread, which received the message. This has been modified to do this task in parallel according to the configuration. This enables the application to leverage the modern CPU multicore capabilities to speed up the state machine XML generation by processing the plan items in parallel.

The conversion of state machine templates to complete state machine representation has been optimized by modifying the data structures related to string manipulation so that lesser objects must be created and basic string manipulation operations like replace and search are faster.

The SCXML implementation being used has been enhanced to make the state machine operations faster. It would save a lot of time if instead of scanning the entire state machine for completed plan items, a predefined count is used to compare against the plan items in one iteration.

For regular orders, a single state machine is formed for the entire plan. This process has been changed for orders with a large plan. A child state machine is created per plan item along with the state machine for the order or plan. This allows for fast evaluation of conditions when events are fired on the state machine because the conditions are not evaluated across the states, but by using the values stored in the database tables. The database condition evaluation is enforced when the state machine is split.

For regular orders, the state machine transitions have been event-enabled so that a redundant condition evaluation does not occur on each event getting fired on the state machine. By extra processing in memory, the state machine has been tweaked to make the transitions specific and make the operations faster.

The following flags have been introduced to enable, disable, and control the large plan orchestration:

```
<ConfValue description="Generator Thread Count" isHotDeployable="true" name="Generator
Thread Count" propname="com.tibco.fom.orch.generator.threadCount" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfNum default="20" value="20" />
</ConfValue>
```

```
<ConfValue description="Split StateMachine Threshold" isHotDeployable="true"
name="Split StateMachine Threshold"
propname="com.tibco.fom.orch.generator.splitStateMachineThreshold" readonly="false"
```

```

sinceVersion="3.0" visibility="Basic">
  <ConfNum default="50" value="50" />
</ConfValue>

<ConfValue description="Enable DB Condition Evaluator for Statemachines for Simple
StateMachine" isHotDeployable="true" name="Enable DB Condition Evaluator for
Statemachines for SimpleStateMachine"
propname="com.tibco.fom.orch.generator.enableDBConditionEvaluatorStateMachine"
readonly="false" sinceVersion="3.0" visibility="Basic">
  <ConfBool default="false" value="false" />
</ConfValue>

```

The following configurations have been introduced in ConfigValues_OMS.xml under the category "OMS UI Grid View and Gantt View Pagination" to support large plans on the Order Management Server user interface:

```

<Category description="OMS UI Grid View and Gantt View Pagination" name="Grid View and
Gantt View Pagination" visibility="Basic">
  <ConfValue description="Enable Grid View and Gantt View Pagination" name="Enable
Grid View and Gantt View Pagination" propname="com.tibco.af.omsui.pagination.enable"
sinceVersion="3.0" visibility="Basic">
    <ConfBool default="false" value="false"/>
  </ConfValue>
  <ConfValue description="OMS UI Grid View Page Size" name="Grid View Page Size"
propname="com.tibco.af.omsui.gridView.pageSize" sinceVersion="3.0" visibility="Basic">
    <ConfNum default="20" value="20"/>
  </ConfValue>
  <ConfValue description="OMS UI Gantt View Page Size" name="Gantt View Page Size"
propname="com.tibco.af.omsui.ganttView.pageSize" sinceVersion="3.0" visibility="Basic">
    <ConfNum default="10" value="10"/>
  </ConfValue>
</Category>

```

Under this category, use the property `com.tibco.af.omsui.pagination.enable` to enable or disable the Order Management Server user interface pagination for plan view. Use the property `com.tibco.af.omsui.gridView.pageSize` to set the page size for Grid View, and use the property `com.tibco.af.omsui.ganttView.pageSize` to set the page size for GANTT view.

Execution Plan

Execution Plan is a process model, which is developed for a concrete order and can also be termed as a collection of the activities that have to be completed to fulfill a customer order. Execution plans usually specify how the process components must be arranged to fulfill the order.

An execution plan consists of the following:

- Plan Tasks or Plan Items with an associated process component and action
- Actions
- Dependencies on the plan items

Plan Tasks with Associated Process Components

One or more plan tasks or items comprise an execution plan. Each plan item is created to fulfill a particular product against a particular action. The process component specified in plan item is invoked for the fulfillment.

Actions

Each plan task has an action associated with it. These are the possible actions you can select for each plan task:

- Provide
- Cease
- Update

- Custom Actions

A plan task manages a particular item. Each action defines what needs to be done for a particular item. An action serves as an annotation to make the execution plan more understandable.

Dependencies

Plans are automatically generated by the system based on the product model for a given order.

A *dependency* can be defined as a relationship between milestones in the plan items. For example, Milestone B cannot start until Milestone A completes.

For details, see [Intermediate Milestones Dependencies](#).

Order Header

The table below lists the information contained in an order header:

Type	Description
Order Ref ID	A unique identifier supplied by the system that submits the order. The Order Reference is used to determine whether the order is a new request or an amendment. Order Management Server does this by checking if an order with the same Order Reference is already stored in the database. If the order already exists, the order is an amendment. If there is not, then it is a new order request. If the orderRefId is not supplied by external system, then Order Management system generates the reference ID and sends in response.
Order ID	Internal ID of the order generated and assigned by Order Management Server.
Status	Current status of an order. For instance, COMPLETE.
Execution Plan	Execution Plan ID.
Required By Date	Indicates the date and time when the order must start fulfillment.
Notes	Notes about the order. Basically, this is any additional text that might be supplied by the summitter or submitting system.
Subscriber ID	Reference ID of the subscriber.
Customer ID	Used to retrieve the current customer profile and to identify the customer to other systems interested in the order.
Changed Date	Date when the order is changed.
Execution Status	Execution status of an order. For instance, COMPLETE.
Required On Date	Currently not supported.
Invoice Address	The address to invoice for the order, if different from the customer address.
Delivery Address	The address to deliver the order, if different from the customer address.

Type	Description
Service Level Agreements (SLA)	This is a list of the identifiers of any service level agreement that applies to a particular order.

Order Line

An Order contains order lines. Each order line has the following information:

Type	Description
Line No	Line no. of an order.
Product ID	The identifier of the specification of the product to be provided.
Inventory ID	Inventory ID.
Action	<p>The action required for the specific product referred to in the order line. You can enter one of the following actions:</p> <ul style="list-style-type: none"> • Provide: The customer has requested a new service. • Cease: The customer has requested that an existing service must cease. • Update: The customer has requested that an existing service be updated in some way. • Cancel: the customer has canceled the requested product. • <Custom Action>: Any defined custom action.
Required By Date	Indicates the date and time when the order line must start fulfillment.
Quantity	The number of the product required.
Required On Date	Currently not supported.
Subscriber ID	Reference ID of the subscriber.
Product Version	Version of a particular product.
Link ID	Link reference ID.
Status	The current status of the order. This is automatically filled in and you cannot amend it. The status changes with the order item's lifecycle.
Status Changed	The date and time that the order line status last changed. This is automatically filled in and you cannot amend it. It initially shows the date and time the order line was created and is updated to reflect later status changes.
UOM (Unit of Measurement)	The unit of measure of the product required.
Delivery Address	The address to deliver the order, if different from the customer address.

Type	Description
Characteristics	A list of product characteristics that are supplied as input parameters to the order to provide additional information to the product specification. For instance, this might be the color of a mobile telephone.

Orchestrator Interfaces

TIBCO Order Management - Long Running Orchestrator is the primary fulfillment engine and implemented in a Java component. Orchestrator requests TIBCO Order Management - Long Running Automated Order Plan Development to generate the execution plan for the order submitted by Order Management Server. On receiving the execution plan from Automated Order Plan Development, it executes the plan tasks for order fulfillment.

This section provides the details of interfaces exposed by Orchestrator for external integration.

For process component development, the following two project library files are shipped with the product under the \$OM_HOME/be/projectLibs directory:

- `OM_Orchestrator.projlib` - This library is used in the TIBCO BusinessEvents studio projects for BE 5.x based process component development. It contains the Orchestrator resources. For example, events, channels/destinations, XML payload schema, JMS connections, JMS application properties, and global variables.
- `OM_Orchestrator_ForDesigner.projlib` - This library is used in the TIBCO Designer projects for the TIBCO BusinessWorks-based process component development and contains simple resources. For example, XML payload schema, JMS connections, JMS application properties, and global variables.



This project library is used and imported in the `OM_TestHarness` project.

The XML schema for all the interfaces exposed by Orchestrator are available in the `$OM_HOME/schemas/schema/orchestrator` directory.

Feasibility Providers

Overview

Feasibility Provider is a customer-specific implementation that checks whether an order is feasible for fulfillment. Feasibility might involve network inventory capacity analysis, stock checks, order line validation, or any other number of checks.

Order Management validation engine (OPE) might be used as part of feasibility checking to determine if the order has the required order lines to constitute a valid order.

Feasibility checking is an optional step in the fulfillment process within Orchestrator. By disabling feasibility checking, no Feasibility Provider is required. However, if feasibility is enabled, then a Feasibility Provider must be available for orders to proceed. Feasibility Providers must conform to the following requirements to be a valid implementation.

Specification

- Receive event messages on a JMS queue.
- Interpret the Feasibility Request event and determine if the order is feasible for fulfillment.
- Create and send a response event on a JMS queue.
- In the response event, specify whether the feasibility check has completed successfully by setting the completed flag to `TRUE` if it was able to determine feasibility or `FALSE` if it was not able to conclude feasibility.

- In the response event specify whether the order has passed feasibility by setting the passed flag to TRUE if the order is feasible, or FALSE if it is not feasible.
- In the response event, optionally specify a list of warning or error messages whether the order has passed feasibility or not.

Since the feasibility checking process is a customer-implemented component, the functional specification for this process is out of scope of this document.

Feasibility Request

Feasibility Request is an event sent by Orchestrator to the Feasibility Provider to request order feasibility checking. It is an asynchronous request/reply event to a JMS queue that returns a reply to the default reply queue for Feasibility Response.

If an exception occurs during feasibility checking, then it must be logged to the Feasibility Provider log. The details of the exception are returned in the response.

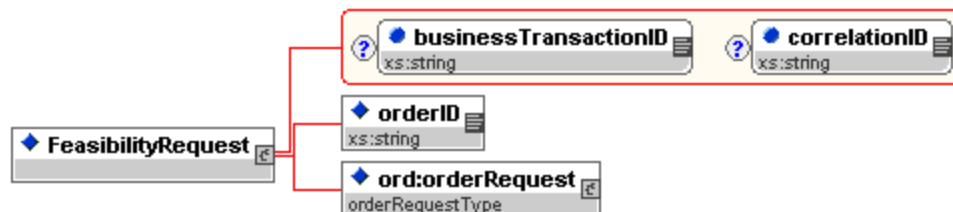
Event Type	Asynchronous request event
Queue or Topic	Queue
Destination	tibco.aff.orchestrator.provider.order.feasibility.request

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the NODE_ID that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The event has the following payload:

Feasibility Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.

Element	Type	Cardinality	Description
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderId	String	Required	Order ID for the order to feasibility check.
orderRequest	Type	Required	Order Request type. See Appendix A for the specification of this type.

Feasibility Response

Feasibility Response is an event sent by Feasibility Provider back to Orchestrator in response to a Feasibility Request event. It is an asynchronous reply event to a JMS queue.

The response for feasibility has **completed** and **passed** flags. Orchestrator routes the order lifecycle based on the returned value of these flags. The two flags can be used to distinguish between technical and business exceptions. For example, a failure to complete would generally indicate a technical exception so complete would be `false`. A validation failure would indicate a business exception where complete would be `true`, but passed would be `false`.

Completed	This flag indicates whether the feasibility call completed. If this is set to <code>true</code> , the passed flag becomes relevant.
Passed	This flag indicates whether the order has passed feasibility.

Based on different scenarios, these flags must be set as follows:

	Completed	Passed	Description
Technical Error	<code>false</code>	<code>false</code> <code>true</code>	Orchestrator refers the order to the Pre-Qualification Failed Handler if error handling is enabled for feasibility, or the error is withdrawn if error handling is not enabled.
Business Error	<code>true</code>	<code>false</code>	Orchestrator refers the order to the Pre-Qualification Failed Handler if error handling is enabled for feasibility, or the error is withdrawn if error handling is not enabled.
Success	<code>true</code>	<code>true</code>	The processing continues as normal.

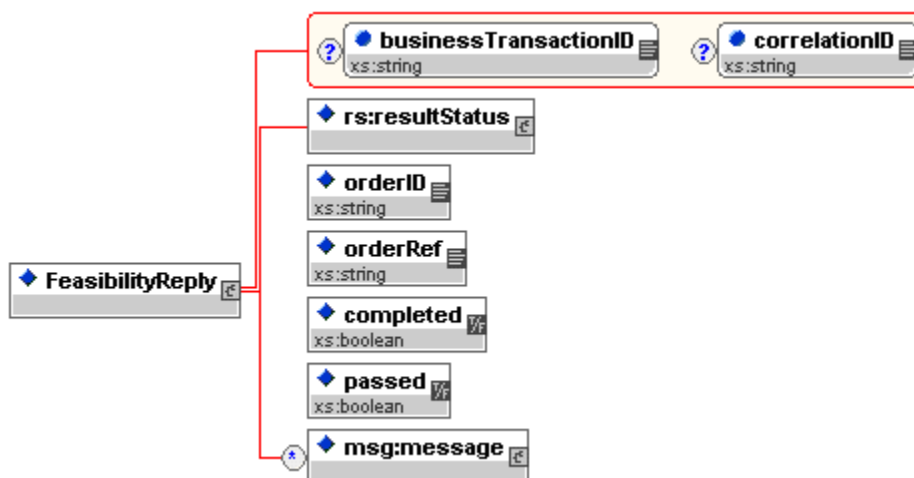
Event Type	Asynchronous reply event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.provider.order.feasibility.reply</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the FeasibilityRequest message, received from the Orchestrator, which must be mapped and sent back in the response message.

The event has the following payload:

Feasibility Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Required	Unique identifier to correlate the request message with a response message. Even though this field is marked as optional in the response schema, it is required for Orchestrator to be able to correlate the response with the correct version of the submitted order. Populate this field the same as correlationID in the request message.
resultStatus	Type	Required	Result status type. See Appendix A for the specification of this type.
orderID	String	Required	Order ID for the order that was feasibility checked.
orderRef	String	Required	Order ref for the order that was feasibility checked.
completed	Boolean	Required	Flag indicating whether the feasibility call completed.
passed	Boolean	Required	Flag indicating whether the order has passed feasibility.

message	Type	0-M	Message type. See Appendix A for the specification of this type. This list of messages is passed to the Pre-Qualification Failed Handler if invoked.
---------	------	-----	--

Process Components

Overview

A Process Component is the implementation of a series of steps that are required to fulfill a plan item. Process components must be implemented by using any JMS-enabled technology provided they meet the requirements outlined here. Typically short-lived, automated processes are implemented in TIBCO BusinessEvents or TIBCO BusinessWorks, when long-lived and manual processes are implemented by using TIBCO BPM.

These are required components in the architecture.

Specification

Process Components must conform to the following requirements to be a valid implementation.

1. Receive event messages on a JMS queue.
2. Receive the following event types:
 - a. Plan Item Execute Request
 - b. Plan Item Suspend Request
 - c. Plan Item Activate Request
3. For plan item execute requests, perform a series of tasks that are required to fulfill the product and action specified in the plan item. Once it has completed, send a plan item execute response.
4. When instructed to do so, halt execution at certain milestones until notified by Orchestrator it might continue.
5. For plan item suspend requests, halt execution of an in-progress process component. This might or might not be possible so it is valid to send back a plan item execute response if execution completed, or plan item suspend response if execution was suspended.
6. For plan item activate requests, resume execution of a previously suspended process component. This resume takes the form of one of the following cases:
 - a. Resume execution from the point where it was previously suspended.
 - b. Cancel execution and roll back previously completed tasks.
 - c. Cancel execution and do not roll back previously completed tasks.
7. Create and send response events on a JMS queue.
8. Respond with the following event types:
 - a. Plan Item Execute Response
 - b. Plan Item Suspend Response

Plan Item Execute Request Event

Plan Item Execute Request Event is sent by Orchestrator to a Process Component to request the fulfillment of a particular plan item. It is received by the Process Component and a series of tasks then executes. It is an asynchronous event to a JMS queue. The response is another asynchronous event on a different JMS queue.

Event Type	Asynchronous event
Queue or Topic	Queue
Destination	tibco.aff.orchestrator.planItem.execute.request



The destination name `tibco.aff.orchestrator.planItem.execute.request` is valid only if owner value is `""`. Otherwise, the destination would be as follows: *(If owner value is defined), the destination would be* `tibco.aff.orchestrator.planItem.<ownertype>.execute.request`.

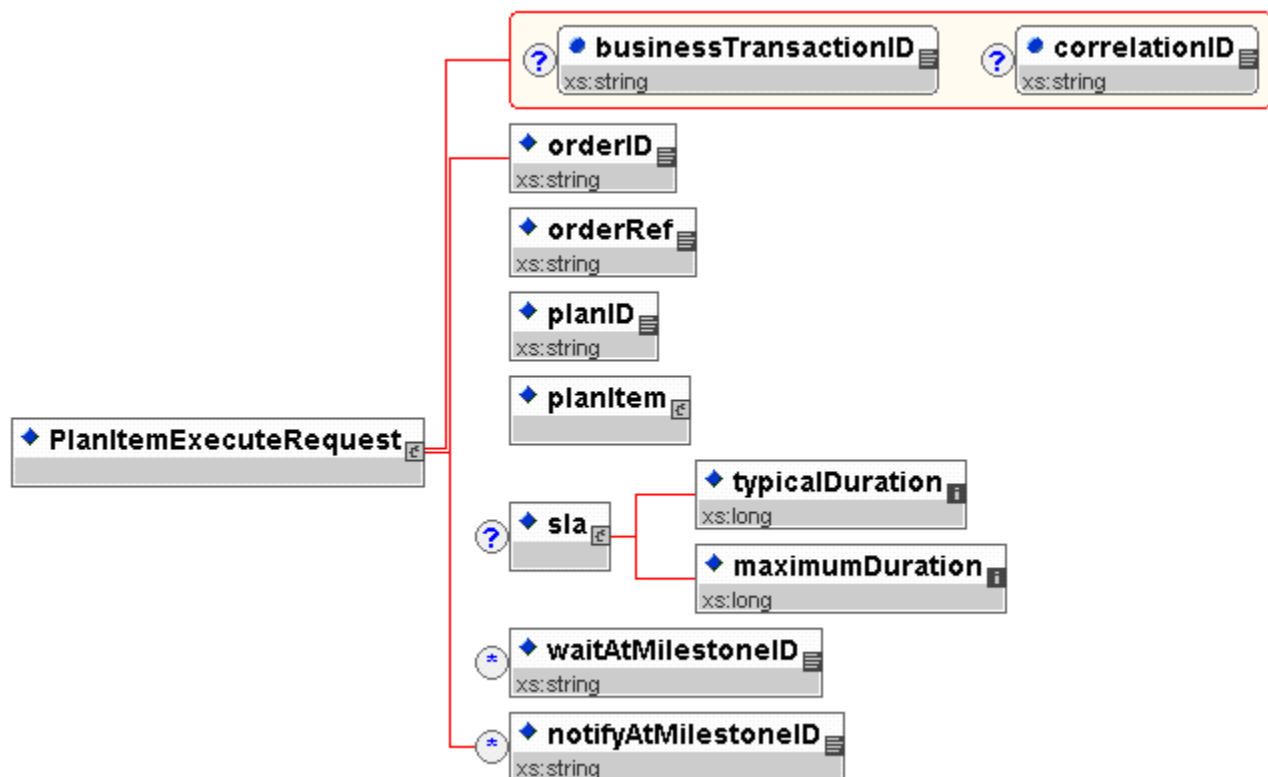
For example, if the owner value in the plan fragment model is BPM, the destination would be `tibco.aff.orchestrator.planItem.BPM.execute.request`.

The event has the following properties:

Property	Type	Cardinality	Description
processComponentID	String	Required	Unique identifier for the Process Component to be executed.
processComponentName	String	Required	Name of the Process Component to be executed. This is the name as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentVersion	String	Required	Version of the Process Component to be executed. This is the version as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentType	String	Required	Type of the Process Component to be executed. This is the type as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentRecordType	String	Required	It is a class of processComponentType. This is the processComponentRecordType as configured in the Process Component Model. If there is no model specified then this field is null.
JMSPriority	Integer	Required	It is the standard JMS message priority to be sent in the outbound message to support order priority.
originator	String	Optional	The value of the NODE_ID that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The payload specification is as follows:

Plan Item Execute Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the plan item to execute.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item to execute.
planID	String	Required	Internal unique identifier for the plan that contains the plan item to execute.
planItem	Type	Required	Plan item type for the plan item to execute. See Appendix A for the specification of this type.
sla	Type	Optional	Service level agreement type.

sla/typicalDuration	Long	Required	Typical duration in msec for this execution when SLAs are implemented in the Process Component.
sla/maximumDuration	Long	Required	Maximum duration in msec for this execution when SLAs are implemented in the Process Component.
waitAtMilestoneID	String	0-M	Milestone ID for a milestone within the Process Component where execution must wait until notified by Orchestrator that it must proceed.
notifyAtMilestoneID	String	0-M	MilestoneID for a milestone within the Process Component where the Process Component must notify Orchestrator that the milestone has been passed during execution.

Plan Item Milestone Release Request Event

Plan Item Milestone Release Event is sent by Orchestrator to a Process Component to instruct it to continue execution when stopped at a particular milestone. It might be possible that this notification occurs before the Process Component has reached the milestone during execution. Therefore it is necessary for the Process Component to maintain a state of the milestone at any time during execution. There is no response to this interface.

Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.milestone.release.request</code>

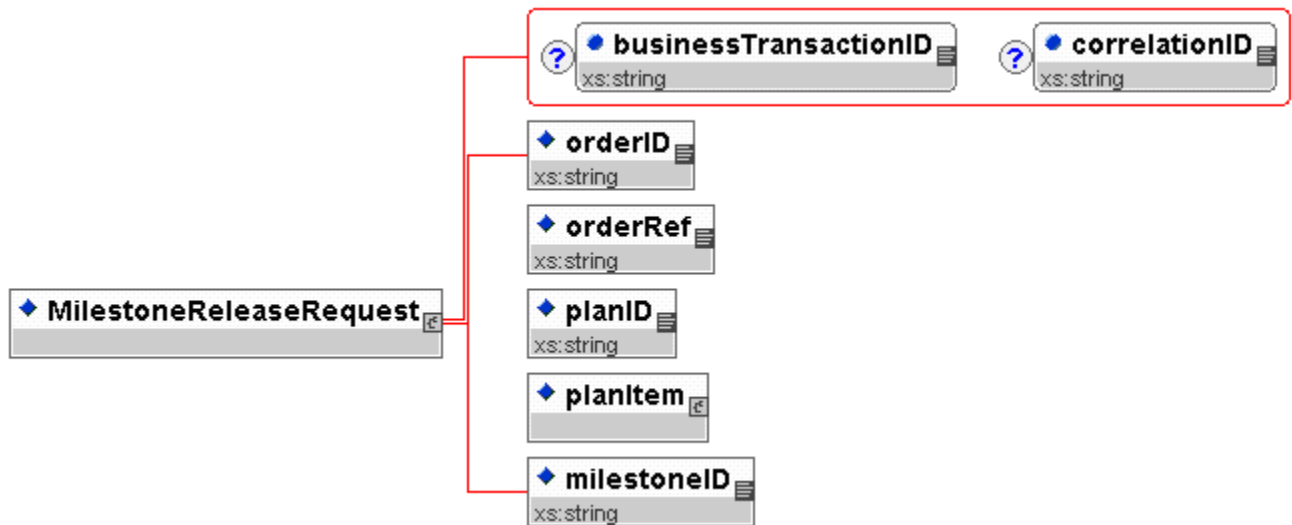
The event has the following properties:

Property	Type	Cardinality	Description
processComponentID	String	Required	Unique identifier for the Process Component to be executed.
processComponentName	String	Required	Name of the Process Component to be executed. This is the name as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentVersion	String	Required	Version of the Process Component to be executed. This is the version as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.

processComponentType	String	Required	Type of the Process Component to be executed. This is the type as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
planID	String	Required	Internal unique identifier for the plan that contains the plan item with the milestone to be released.
planItemID	String	Required	Unique identifier for the plan item that contains the milestone to be released.
milestoneID	String	Required	Unique identifier for the milestone within the plan item and plan to be released.
originator	String	Optional	The value of the NODE_ID that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The payload specification is as follows:

Plan Item Milestone Release Request



Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.

correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the plan item with the milestone to be released.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item with the milestone to be released.
planID	String	Required	Internal unique identifier for the plan that contains the plan item with the milestone to be released.
planItem	Type	Required	Plan item type for the plan item with the milestone to be released. See Appendix A for the specification of this type.
milestoneID	String	Required	Unique identifier for the milestone within the plan item and plan to be released.

Plan Item Milestone Notify Request Event

Plan Item Milestone Notify Event is sent by a Process Component to Orchestrator to notify the orchestration engine that a particular milestone has been passed during execution. This event enables the Orchestrator to release the milestone, which was waiting on the current milestone that was notified.

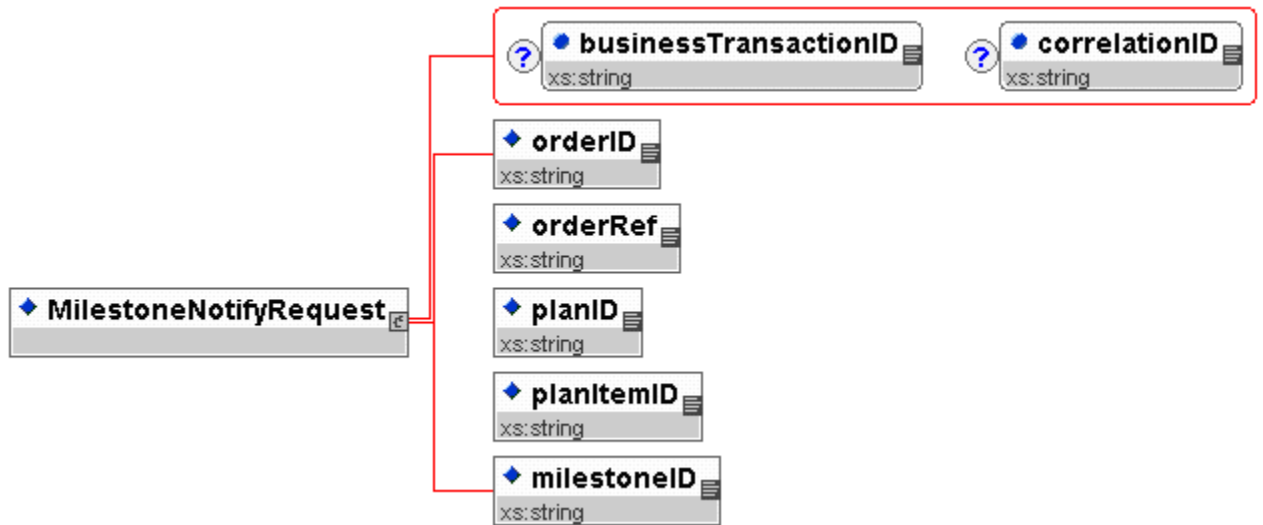
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.milestone.notify.request</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the PlanItemExecuteRequest message, received from the Orchestrator, which must be mapped and sent back in this response message.

The payload specification is as follows:

Plan Item Milestone Notify Request Event



Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderId	String	Required	Internal unique identifier for the order associated with the plan containing the plan item with the milestone to notify.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item with the milestone to notify.
planID	String	Required	Internal unique identifier for the plan that contains the plan item with the milestone to notify.

planItemID	String	Required	Unique identifier for the plan item within the plan with the milestone to notify.
milestoneID	String	Required	Unique identifier for the milestone within the plan item in the plan to notify.

Plan Item Execute Response Event

Plan Item Execute Response Event is sent by a Process Component as a response to a Plan Item Execute Request Event or a Plan Item Suspend Event. Orchestrator receives the result and interprets the result accordingly. It is an asynchronous event to a JMS queue.

The response for Plan Item Execute has success, completed, and canceled flags. Orchestrator does not take any action in response to the canceled flag. However it does route plan items to either Plan Item Internal Error Handler or External Error Handler Component if either completed or success is set to false. Functionally, Orchestrator handles both of these the same. Plan Item Failed Handlers might choose to handle the exception differently depending on a completed or failure status.

The two flags can be used to distinguish between technical and business exceptions. For example, a failure to complete would generally indicate a technical exception so completed would be false. A validation failure would indicate a business exception where complete would be true, but success would be false.

Completed	This flag indicates that the Process Component completed. If this is set to true, then the Success flag becomes relevant. If this is false then the Process Component did not complete and the Success flag is automatically considered to be false as well.
Success	This flag indicates whether the Process Component was successful. This is only relevant if the complete flag is set to true.

The possible response scenarios are:

	Complete	Passed	Description
Technical Error	False	False True	Orchestrator retries the Process Component call for the defined number of retries with the defined retry interval. If the Process Component call continues to fail, then it refers the plan item to the Plan Item Failed Handler.
Business Error	True	False	Orchestrator refers the plan item to the Plan Item Failed Handler.
Success	True	True	Processing continues as normal.

In addition to the completed and success values, the Plan Item Execute Response Event also allows returning a canceled flag. This is only valid if responding to a Plan Item Activate Request Event and it indicates whether the cancellation was completed successfully whether or not a roll back was requested. The completed and success values retain the same definitions in the event of an activation request as in an execution request.

The possible response scenarios are:

	Canceled	Description
Execute Request	False	No cancellation occurred.
Activate Request with Rollback	True	Cancellation requested with roll back.
Activate Request without Rollback	True	Cancellation requested with no roll back.

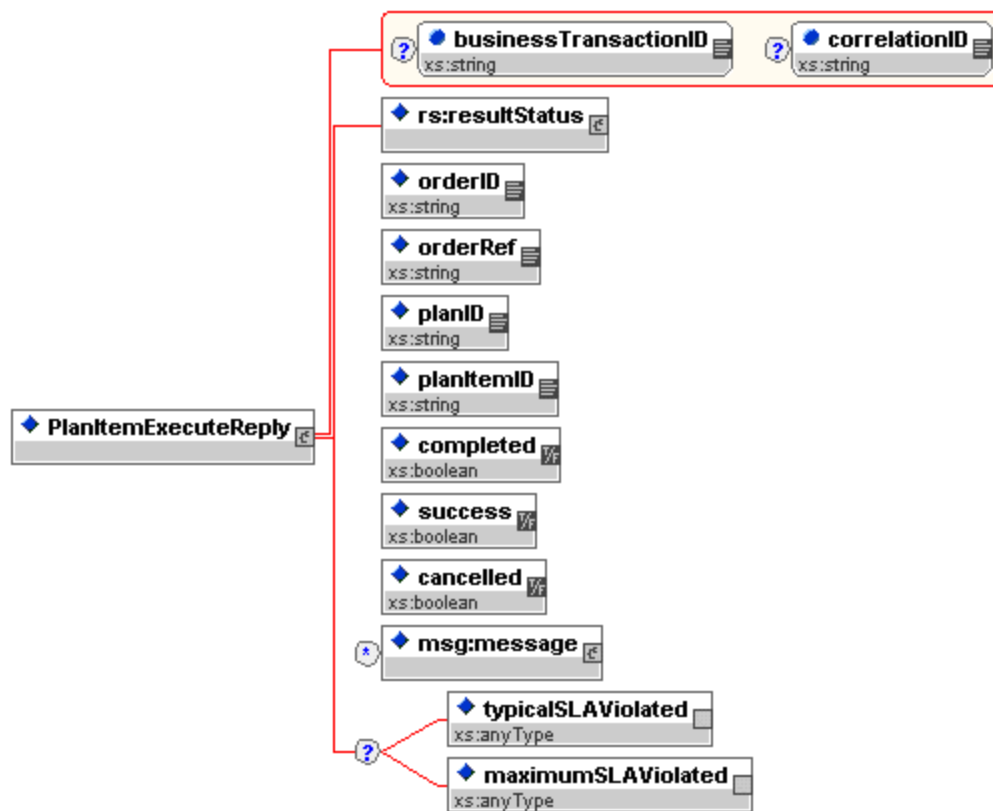
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.execute.reply</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the PlanItemExecuteRequest message, received from the Orchestrator, which must be mapped and sent back in the response message.

The payload specification is as follows:

Plan Item Execute Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
resultStatus	Type	Required	Result status type. See Appendix A for the specification of this type.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the plan item to execute.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item to execute.
planID	String	Required	Internal unique identifier for the plan that contains the plan item to execute.
planItemID	String	Required	Unique identifier for the plan item within the plan to be executed.
completed	Boolean	Required	Flag indicating if the Process Component completed processing.
success	Boolean	Required	Flag indicating if the Process Component completed successfully.
canceled	Boolean	Required	Flag indicating that the Process Component successfully canceled previously completed tasks.
message	Type	0-M	Message type. See Appendix A for the specification for this type.
typicalSLAViolated	Type	Optional	Flag indicating that the execution time of the Process Component violated the typical SLA duration.
maximumSLAViolated	Type	Optional	Flag indicating that the execution time of the Process Component violated the maximum SLA duration.

Plan Item Suspend Request Event

Plan Item Suspend Request Event is sent by Orchestrator to a Process Component to request suspension of execution of a particular plan item. It is received by the Process Component, which then either suspends execution or completes execution. It is an asynchronous event to a JMS queue. The response is another asynchronous event on a different JMS queue.

Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.suspend.request</code>



The destination name `tibco.aff.orchestrator.planItem.suspend.request` is valid only if owner value is `""`. Otherwise, the destination would be as follows: *(If owner value is defined), the destination would be `tibco.aff.orchestrator.planItem.<ownertype>.suspend.request`.*

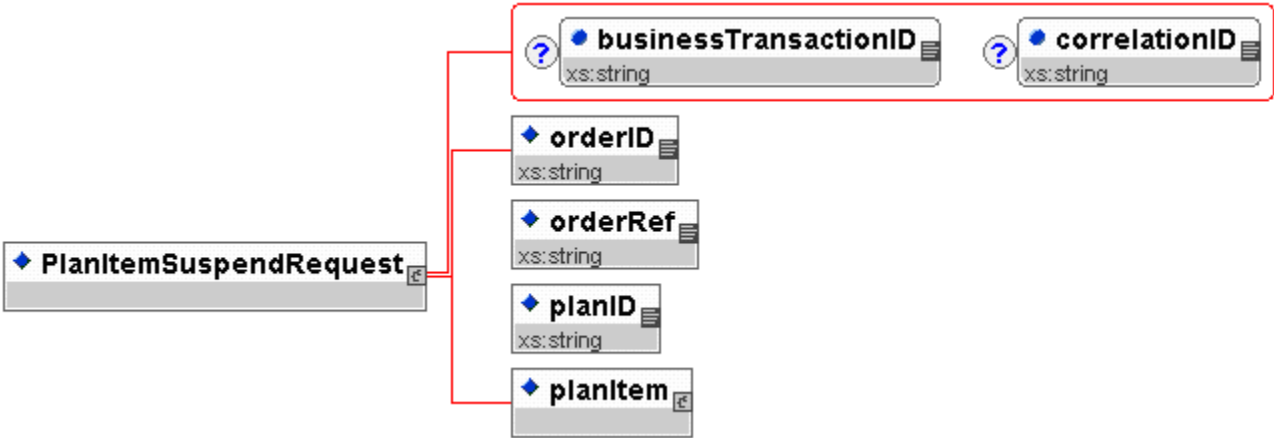
For example, if the owner value in the plan fragment model is BPM, the destination would be `tibco.aff.orchestrator.planItem.BPM.suspend.request`.

The event has the following properties:

Property	Type	Cardinality	Description
processComponent ID	String	Required	Unique identifier for the Process Component to be executed.
processComponent Name	String	Required	Name of the Process Component to be executed. This is the name as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponent Version	String	Required	Version of the Process Component to be executed. This is the version as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponent Type	String	Required	Type of the Process Component to be executed. This is the type as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponent RecordType	String	Required	It is a class of processComponentType. This is the processComponentRecordType as configured in the Process Component Model. If there is no model specified then this field is null.
JMSPriority	Integer	Required	It is the standard JMS message priority to be sent in the outbound message to support order priority.
originator	String	Optional	The value of the <code>NODE_ID</code> that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The payload specification is as follows:

Plan Item Suspend Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderId	String	Required	Internal unique identifier for the order associated with the plan containing the plan item to be suspended.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item to be suspended.
planID	String	Required	Internal unique identifier for the plan that contains the plan item to be suspended.
planItem	Type	Required	Plan item type for the plan item to be suspended. See Appendix A for the specification of this type.

Plan Item Suspend Response Event

Plan Item Suspend Response Event is sent by a Process Component as a response to a Plan Item Suspend Request Event. Orchestrator receives the result and interprets the result accordingly. It is an asynchronous event to a JMS queue.

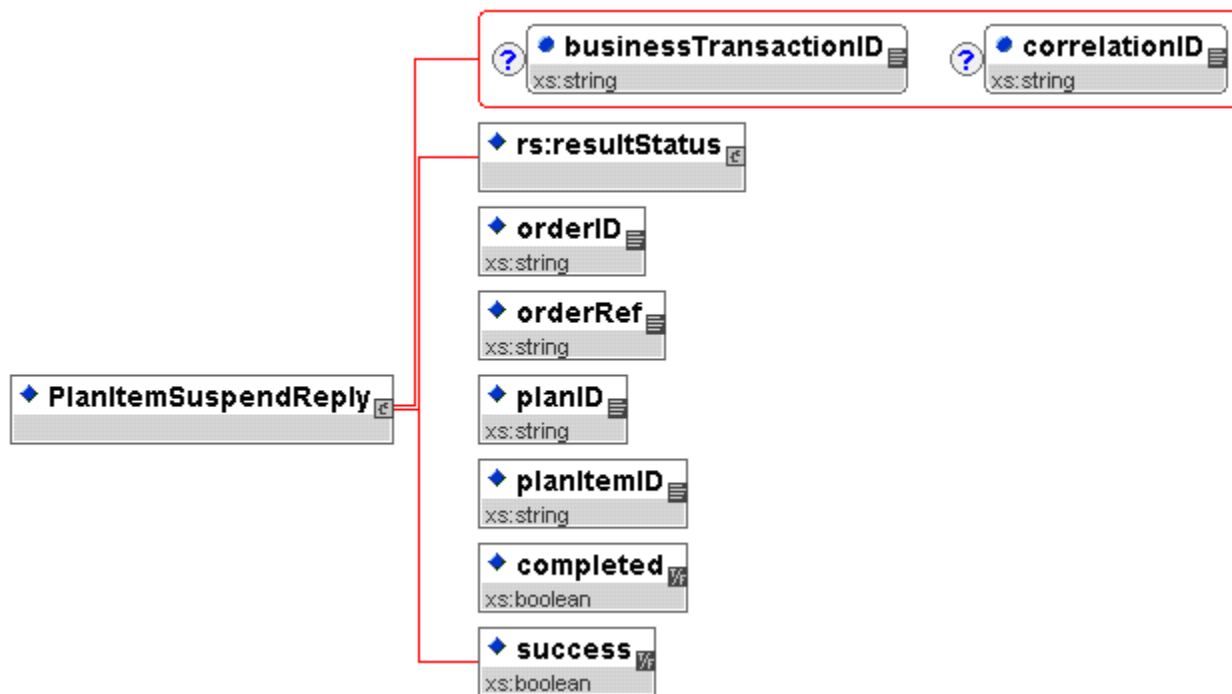
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.suspend.reply</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the PlanItemSuspendRequest message, received from the Orchestrator, which must be mapped and sent back in the response message.

The payload specification is as follows:

Plan Item Suspend Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransacti onID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
resultStatus	Type	Required	Result status type. See Appendix A for the specification of this type.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the plan item to suspend.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item to suspend.

planID	String	Required	Internal unique identifier for the plan that contains the plan item to suspend.
planItemID	String	Required	Unique identifier for the plan item within the plan to be suspended.
completed	Boolean	Required	Flag indicating if the Process Component suspend completed processing.
success	Boolean	Required	Flag indicating if the Process Component suspend completed successfully.

If any of the flag `completed` or `success` is false in `PlanItemSuspendResponse`, then the process component suspension gets failed and you have to resubmit the `PlanItemSuspendResponse`.

Plan Item Activate Request Event

Plan Item Activate Request Event is sent by Orchestrator to a Process Component to request activation of a previously suspended plan item. It is received by the Process Component, which then resumes, cancels with roll back, or cancels without roll back. It is an asynchronous event to a JMS queue. There is no specific response to a Plan Item Activate Request Event, however the Process Component is expected to complete processing and return a Plan Item Execute Response Event as usual.

Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.planItem.activate.request</code>

The destination name `tibco.aff.orchestrator.planItem.activate.request` is valid only if owner value is `""`. Otherwise, the destination would be as follows: *(If owner value is defined), the destination would be `tibco.aff.orchestrator.planItem.<ownertype>.activate.request`.*

For example, if the owner value in the plan fragment model is `BPM`, the destination would be `tibco.aff.orchestrator.planItem.BPM.activate.request`.

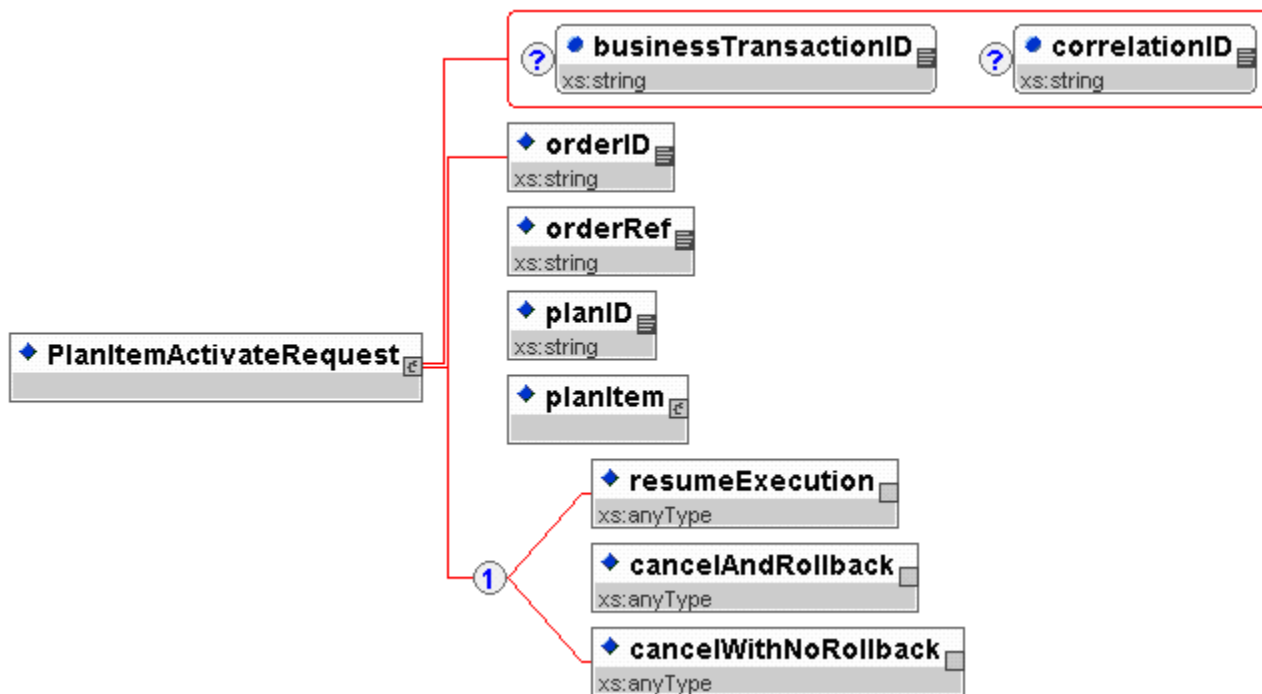
The event has the following properties:

Property	Type	Cardinality	Description
processComponentID	String	Required	Unique identifier for the Process Component to be executed.
processComponentName	String	Required	Name of the Process Component to be executed. This is the name as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.

processComponentVersion	String	Required	Version of the Process Component to be executed. This is the version as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentType	String	Required	Type of the Process Component to be executed. This is the type as configured in the Process Component Model for the specified processComponentID. If there is no model specified then this field is null.
processComponentRecordType	String	Required	It is a class of processComponentType. This is the processComponentRecordType as configured in the Process Component Model. If there is no model specified then this field is null.
JMSPriority	Integer	Required	It is the standard JMS message priority to be sent in the outbound message to support order priority.
originator	String	Optional	The value of the NODE_ID that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The payload specification is as follows:

Plan Item Activate Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the plan item to activate.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the plan item to activate.
planID	String	Required	Internal unique identifier for the plan that contains the plan item to activate.
planItem	Type	Required	Plan item type for the plan item to activate. See Appendix A for the specification of this type.
resumeExecution	Type	Required, Choice	Flag indicating that the Process Component must resume execution from the point where it was previously suspended.

cancelAndRollback	Type	Required, Choice	Flag indicating that the Process Component must cancel execution and roll back previously completed tasks.
cancelWithNoRollback	Type	Required, Choice	Flag indicating that the Process Component must cancel execution and not roll back previously completed tasks.

Pre-qualification Failed Handlers

Overview

A Pre-Qualification Failed Handler is a customer-implemented component used to manage failed feasibility and plan development steps in Orchestrator. During the fulfillment process, Orchestrator would optionally call out to a Feasibility Provider and always call out to a Plan Development Provider.

The Feasibility Provider analyzes the order to determine if it can be fulfilled. If it indicates that the order cannot be fulfilled, then the fulfillment process cannot proceed. The order might be referred to the Pre-Qualification Failed Handler for manual intervention if feasibility error handling is enabled.

The Plan Development Provider designs a plan from the order. If a plan cannot be designed then the fulfillment process cannot proceed and the order might be referred to the Pre-Qualification Failed Handler for manual intervention if OPD error handling is enabled.

Pre-Qualification Failed Handler is an optional component in the architecture. Customers might choose to implement full error handling within the Feasibility Provider and the Plan Development Provider. In that case it is not valid to return anything back to Orchestrator other than **passed** in the case of feasibility and **success** in the case of plan development. If anything else is returned to Orchestrator there is no handler to process the result and the plan remains either in feasibility or OPD state.

Specification

Pre-Qualification Failed Handlers must conform to the following requirements to be a valid implementation.

1. Receive event messages on a JMS queue.
2. Interpret the Pre-Qualification Failed Request event and determine how best to route the failed order for further processing.
3. If necessary, distinguish between feasibility and plan development failures and direct the order to the correct handling component.
4. Create and send a response event on a JMS queue.
5. In the response event specify one of three possible actions that Orchestrator is to take in response to the error: resubmit, retry, or withdraw.

The three actions that can be specified are as follows:

Resubmit	The Pre-Qualification Failed Handler sends back a new orderRequest that Orchestrator resubmits automatically. This is handled as a pre-execution order amendment and execution begins from Submitted state. An audit history of the original order is maintained in the Transient Data Store. To resubmit, the orderID and orderRef in the new order must match that of the original order.
Retry	Orchestrator resubmits the order to either the Feasibility Provider or the Plan Development Provider as it was originally submitted. If this new call fails, the order is sent back to the Pre-Qualification Failed Handler again.

Withdraw	The order is withdrawn from Orchestrator and not be fulfilled. It is deleted from the engine and Transient Data Store.
----------	--

The details of the Pre-Qualification Failed Handler is left as a customer-specific implementation. An example of a handler could be the following:

1. Receive a Pre-Qualification Failed Handler Request event messages on a JMS queue by a BusinessWorks process.
2. BusinessWorks starts a process instance in iProcess.
3. The iProcess process model creates a manual task and displays a form in a work queue for operations support. The form displays the order and the details of the failure.
4. Operations support reviews the task and chooses one of the following options:
 - a. Make changes to the order that allows the order to pass feasibility and plan development. The order must be resubmitted.
 - b. Update configurations in back-end systems or product catalog that allows the order to pass feasibility and plan development. The order might then be retried.
 - c. Determine the order is invalid and withdraw the order.
5. The task is completed.
6. iProcess then invokes a BusinessWorks service that creates the Pre-Qualification Failed Handler Response event, and populates the event with the appropriate information and flags the response to resubmit, retry, or withdraw as appropriate. This event is then sent on a JMS queue to Orchestrator. The iProcess procedure then terminates.
7. Orchestrator receives the Pre-Qualification Failed Handler Response and processes it accordingly.

Pre-Qualification Failed Request Event

Pre-Qualification Failed Request Event is sent by Orchestrator in response to a failed feasibility or plan development call. It is received by the Pre-Qualification Failed Handler for manual processing. It is an asynchronous event to a JMS queue. The response is another asynchronous event on a different JMS queue.

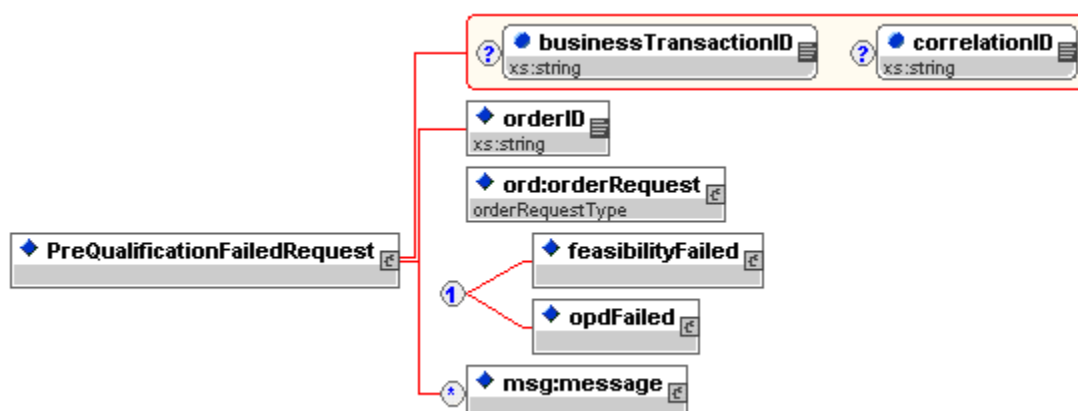
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.provider.order.prequal.failed.request</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the <code>NODE_ID</code> that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The event has the following payload:

Pre-Qualification Failed Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderId	String	Required	Order ID for the order that failed feasibility or plan development.
orderRequest	Type	Required	Order Request type for the order that failed feasibility or plan development. See Appendix A for the specification of this type.
feasibilityFailed	Type	Required, Choice	Flag indicating that this failure was due to a feasibility failure.
opdFailed	Type	Required, Choice	Flag indicating that this failure was due to a plan development failure.
message	Type	0-M	Message type. See Appendix A for the specification of this type. This is any list of messages passed back from the Feasibility Provider or the Plan Development Provider.

Pre-qualification Failed Response Event

Pre-Qualification Failed Response Event is sent by Pre-Qualification Failed Handler as a response to the failed feasibility or plan development step. Orchestrator receives the result and interprets the result accordingly. It is an asynchronous event to a JMS queue.

Event Type	Asynchronous event
------------	--------------------

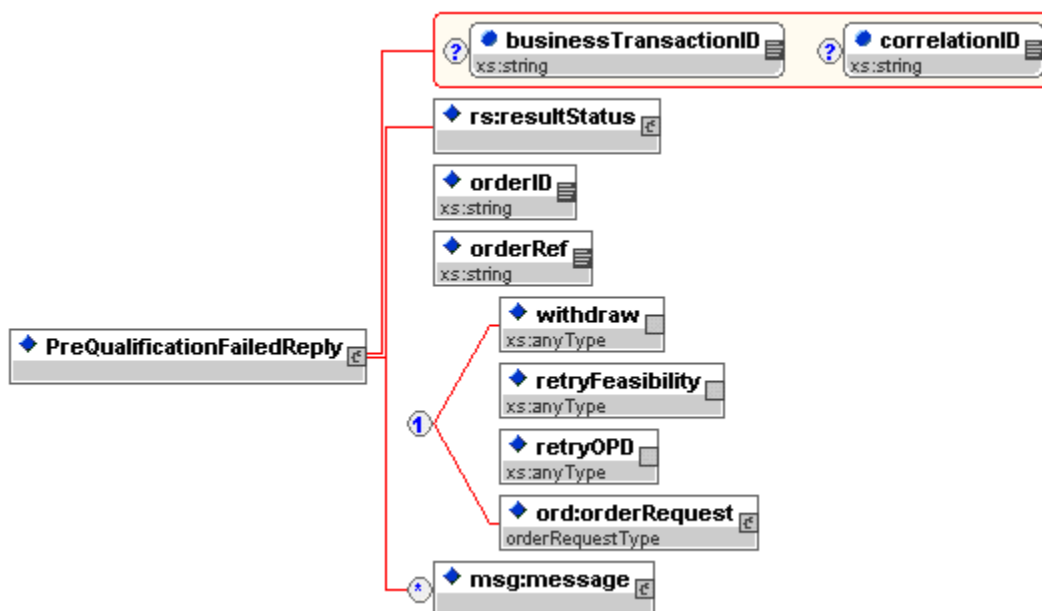
Queue or Topic	Queue
Destination	tibco.aff.orchestrator.provider.order.prequal.failed.reply

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the Pre-QualificationFailedRequest message, received from the Orchestrator, which must be mapped and sent back in the response message.

The event has the following payload:

Pre-qualification Failed Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransacti onID	String	Optional	Unique identifier for tracing purposes across function calls.

correlationID	String	Required	Unique identifier to correlate the request message with a response message. Even though this field is marked as optional in the response schema, it is required for Orchestrator to be able to correlate the response with the correct version of the submitted order. Populate this field the same as correlationID in the request message.
orderID	String	Required	Internal unique identifier for the order that failed feasibility or plan development.
orderRef	String	Required	External unique identifier for the order that failed feasibility or plan development.
withdraw	Type	Required, Choice	Flag indicating that the order must be withdrawn.
retryFeasibility	Type	Required, Choice	Flag indicating that the order must retry feasibility without any changes. This response might be made in cases where the request was for either feasibility or plan development failure.
retryOPD	Type	Required, Choice	Flag indicating that the order must retry plan development without any changes. This response might be made in cases where the request was for plan development failure. Technically there is no restriction on doing so in the case of a feasibility failure, but functionally it does not make sense to do so.
orderRequest	Type	Required, Choice	Order request type. See Appendix A for the specification of this type. This is provided in the case of an order resubmit.
message	Type	0-M	Message type. See Appendix A for the specification of this type. If populated, this list of messages are returned in any Submit Order Response message occurring after the call to the PreQualification Failed Handler occurred.

Plan Item External Error Handlers

Overview

Plan Item External Error Handler is a customer-implemented component used to manage failed plan items. During fulfillment, the Orchestrator requests plan item execution from a Process Component. When execution completes, the Process Component might specify one of three result conditions:

- Execution completed successfully
- Execution completed, but with error
- Execution did not complete

Successful execution results in the plan item being flagged as Complete and execution continuing with the next items in the plan.

Error or incomplete execution means that the plan item has not been successfully completed and therefore the next items in the plan must not begin execution until the conditions that caused the failure are rectified.

Orchestrator does not distinguish between an incomplete execution and an error response when determining how to handle the failed plan item. Both are handled the same by invoking the Plan Item Error Handler and indicating, which failure mode returned. The semantics of how to determine whether a Process Component is incomplete or in error is left to a customer-specific interpretation and implementation. The implementation between Process Components and the Plan Item Error Handler must be consistent.

Plan Item Error Handler is an optional component in the architecture. Customers might choose to implement full error handling within the Process Component. In that case it is not valid to return anything back to Orchestrator other than **success**. If anything else is returned to Orchestrator, there would be no handler to process the result and the plan remains in an execution state without progressing beyond the failed plan item.

Specification

Plan Item Error Handlers must conform to the following requirements to be a valid implementation.

1. Receive event messages on a JMS queue.
2. Interpret the Plan Item Failed Request event and determine how best to route the failed plan item for further processing.
3. If necessary, distinguish between incomplete execution and error response execution and interpret the Error Handler field in the Plan Item Failed Request and direct the plan item to the correct handling component.
4. Create and send a response event on a JMS queue.
5. In the response event, specify one of three possible actions that Orchestrator is to take in response to the reply: retry, resume, or complete.

The three actions that can be specified are as follows:

Retry	The plan item is resubmitted to the Process Component to start over from the beginning. This occurs immediately upon receipt of the Plan Item Failed Response event as all dependencies have been previously satisfied. The Process Component is notified to re-execute the plan item through a Plan Item Execute Request event. If Orchestrator has been automatically set up to retry failed process components, and this retry fails as well, the retry count is not reset to zero. In other words, if the retry from a Plan Item Failed Handler response fails, then that failure is immediately redirected back to the Plan Item Failed Handler for processing again as a new failed plan item, and not automatically retried further.
Resume	The plan item is resumed in the Process Component from the point of failure. The implementation details of this are left for Process Component design. However it is conceivable that the Process Component might choose to handle a resume the same as a full retry if it is not functionally possible to resume execution from the point of failure. The Process Component is notified to resume the plan item through a Plan Item Activate event.
Complete	The plan item is considered to be completed and not resubmitted to the Process Component. Orchestrator marks the plan item as Complete and processing continues. Any dependencies on the newly Completed plan item is evaluated and further plan items triggered as in the normal execution.

The details of the Plan Item Failed Handler is left as a customer-specific implementation. An example of a handler could be the following:

1. Receive a Plan Item Failed Handler Request event message on a JMS queue by a BusinessWorks process.

2. BusinessWorks starts a process instance in iProcess.
3. The iProcess process model makes a service call out to Transient Data Store to retrieve the order.
4. The iProcess process model creates a manual task and displays a form in a work queue for operations support. The form displays the order and the details of the failure.
5. Operations support reviews the task and chooses one of the following options:
 - a. Make changes to the order, which lets the order to process successfully in the Process Component. The changed order is then saved in Transient Data Store and the plan item might be retried or resumed.
 - b. Make changes to back-end systems that lets the order to process successfully in the Process Component. The plan item might be retried or resumed.
 - c. Implement the required functionality for the plan item manually in the back-end systems. The plan item might be completed.
6. iProcess then invokes a BusinessWorks service that creates the Plan Item Failed Handler Response event, populates the event with the appropriate information, and flags the plan item to complete, retry, or resume. This event is then sent on a JMS queue to Orchestrator.
7. Orchestrator receives the Plan Item Failed Handler Response and processes it accordingly.

Plan Item Failed Request Event

Plan Item Failed Request Event is sent by Orchestrator in response to a failed plan item. It is received by the Plan Item Failed Handler for manual processing. It is an asynchronous event to a JMS queue. The response is another asynchronous event on a different JMS queue.

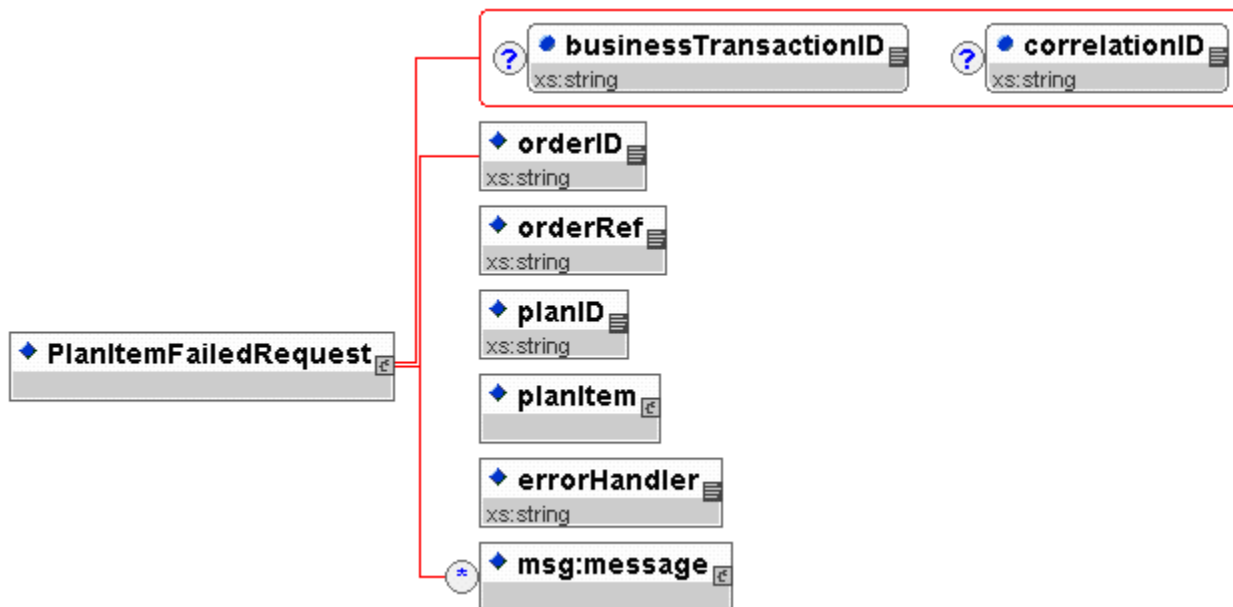
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	<code>tibco.aff.orchestrator.provider.planItem.failed.request</code>

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the <code>NODE_ID</code> that is assigned to the instance. This property is sent by the Orchestrator in all the outbound JMS messages and is expected to be mapped back by the external systems (process components, feasibility providers, pre-qualification failure handlers, and error handlers) in the corresponding response messages.

The event has the following payload:

Plan Item Failed Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransaction ID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Optional	Unique identifier to correlate the request message with a response message.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the failed plan item.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the failed plan item.
planID	String	Required	Internal unique identifier for the plan that contains the failed plan item.
planItem	Type	Required	Plan item type for the plan item that failed. See Appendix A for the specification of this type.
errorHandler	String	Required	Name of the error handler to invoke for this failed plan item. This value is either populated from the Process Component Model for the Process Component if it exists or with the default error handler from the Orchestrator configuration.
message	Type	0-M	Message type. See Appendix A for the specification of this type. This is the list of messages as returned in the Plan Item Execute Response Event.

Plan Item Failed Response Event

Plan Item Failed Response Event is sent by Plan Item Failed Handler as a response to the failed plan item. Orchestrator receives the result and interprets the result accordingly. It is an asynchronous event to a JMS queue.

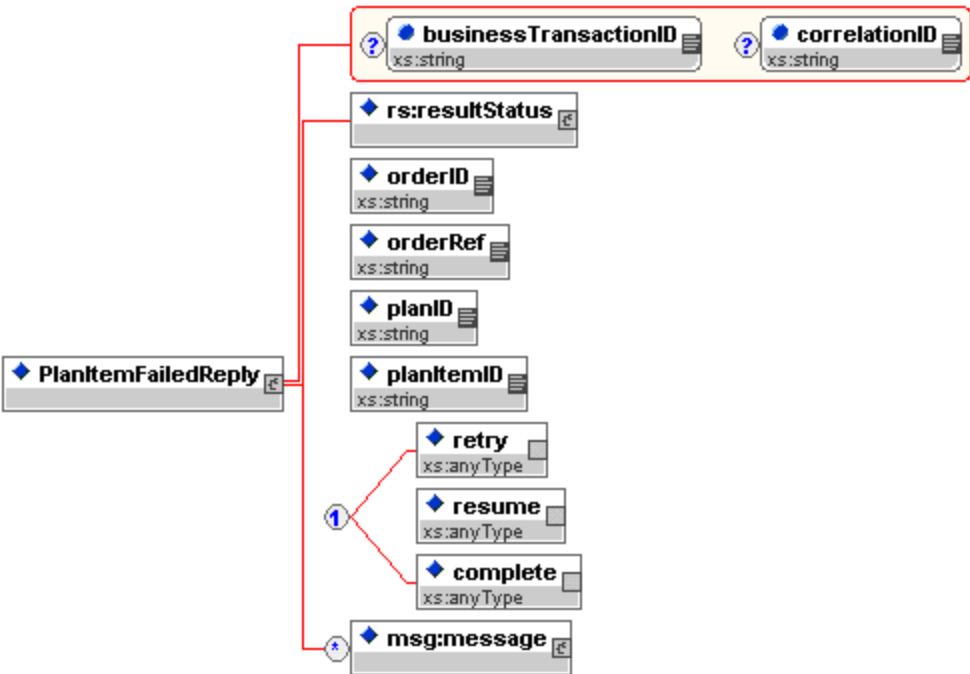
Event Type	Asynchronous event
Queue or Topic	Queue
Destination	tibco.aff.orchestrator.provider.planItem.failed.reply

The event has the following property:

Property	Type	Cardinality	Description
originator	String	Optional	The value of the originator property in the PlanItemFailedRequest message, received from the Orchestrator, which must be mapped and sent back in the response message.

The event has the following payload:

Plan Item Failed Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
---------	------	-------------	-------------

businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
correlationID	String	Required	Unique identifier to correlate the request message with a response message. Even though this field is marked as optional in the response schema, it is required for Orchestrator to be able to correlate the response with the correct version of the submitted order. Populate this field the same as correlationID in the request message.
orderID	String	Required	Internal unique identifier for the order associated with the plan containing the failed plan item.
orderRef	String	Required	External unique identifier for the order associated with the plan containing the failed plan item.
planID	String	Required	Internal unique identifier for the plan that contains the failed plan item.
planItemID	String	Required	Unique identifier for the plan item within the plan that failed.
retry	Type	Required, Choice	Flag indicating that the plan item must be retried.
resume	Type	Required, Choice	Flag indicating that the plan item must be resumed from the point of failure.
complete	Type	Required, Choice	Flag indicating that the plan item must be marked as Complete and execution continue.
message	Type	0-M	Message type. See Appendix A for the specification of this type. If populated, this list of messages are returned in any Submit Order Response message occurring after the call to the Plan Item Failed Handler occurred.

Automated Order Plan Development

This section describes the functions of the Automated Order Plan Development feature in TIBCO Order Management - Long Running.

Overview

Basic *Automated Order Plan Development* is the capability to create custom order plans that fulfill an order, taking into account the specifications of the required products and the products currently provided to a customer.

A *Product Model* contains bundles and products services. A Product model also contains concepts such as sequencing and dependencies.

When an order is received, order lines are decomposed by using a product model. The product specification for each order line is extracted from a product catalog by the decomposition component.

The product specification is required to create execution plan fragments. These execution plan fragments define services, products, and resources required. For example, an order line might contain a bundle, which might be comprised of several products and services. Taking into consideration factors such as sequencing and dependencies, these execution plan fragments are then combined to create a single execution plan.

Architecture

Starting with TIBCO Order Management - Long Running version 2.1.0, Automated Order Plan Development is a Java based component. The Automated Order Plan Development component is, by default, collocated with Order Management Server. This simplifies the deployment and administration. It also improves the performance.

Deployment

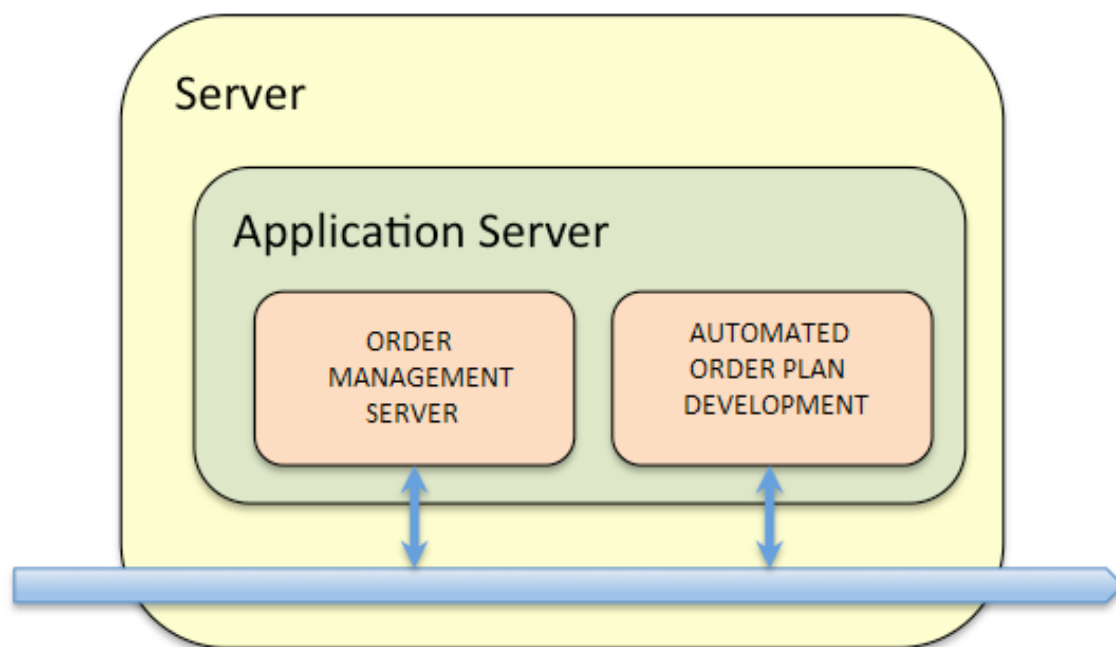
Automated Order Plan Development can be deployed in different ways:

- Collocated
- Standalone
- Custom

Collocated Mode

In *collocated* mode, the Automated Order Plan Development component is deployed on the same application server than Order Management Server. This is the default mode, and it does not require any separate deployment. The omsServer service under roles contains both the Order Management Server component and Automated Order Plan Development component. Even though Automated Order Plan Development is collocated on the same application server, most of the communication is made through Enterprise Message Service.

Automated Order Plan Development in Collocated Mode



To configure this application to use Automated Order Plan Development in collocated mode, set:

```
com.tibco.fom.aopd.deployMode=AOPD_colocated
```

in the file `$OM_HOME/roles/omsServer/standalone/config/profiles.properties`. It tells the application to use the Automated Order Plan Development service stat in the omsServer service.

Standalone Mode

In *standalone* mode, Automated Order Plan Development is not running on the same application server than Order Management Server. Automated Order Plan Development can run on another application server, either on the same machine or a separate machine.

To configure this application to use Automated Order Plan Development in standalone mode, set:

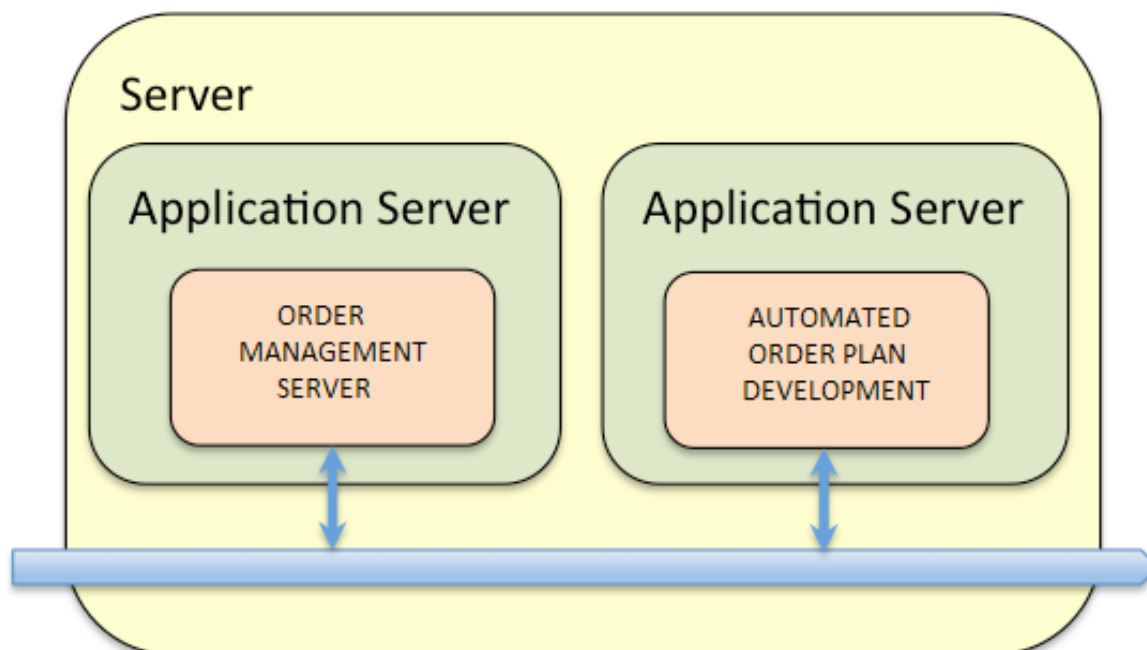
```
com.tibco.fom.aopd.deployMode=AOPD_standalone
```

in the file `$OM_HOME/roles/aopd/standalone/config/profiles.properties`. It tells this application to not use the Automated Order Plan Development implementation that is in omsServer service.

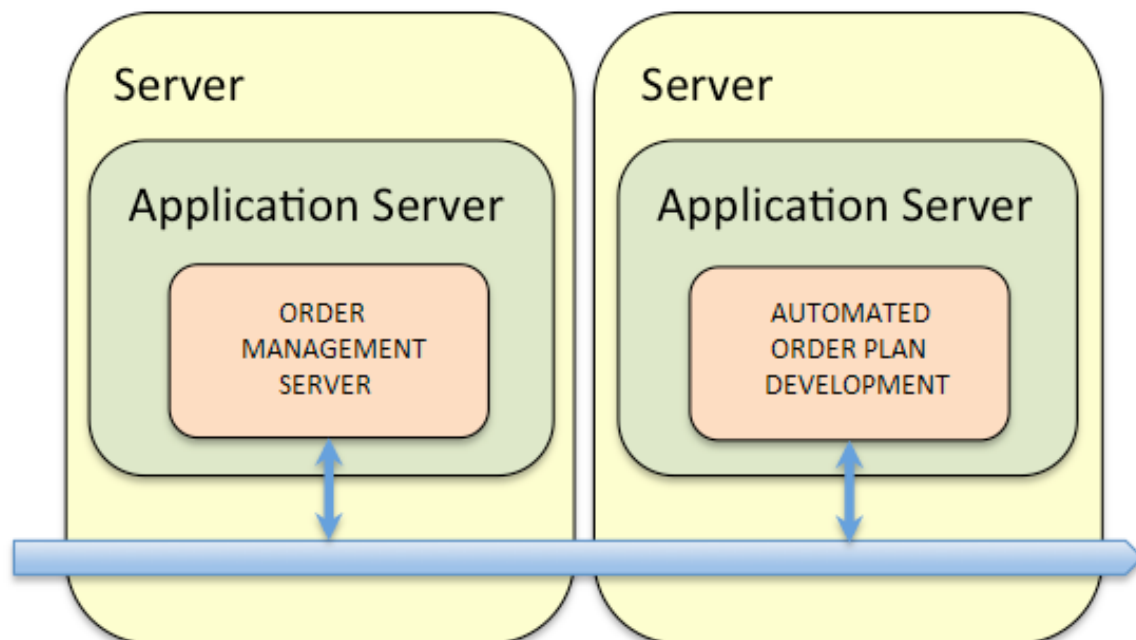
Using shipped Automated Order Plan Development implementation

The installation provides the Automated Order Plan Development implementation as a separate service, which can be run independently. The service can be started separately, either on the same machine (vertical scaling), or a different machine (horizontal scaling).

Automated Order Plan Development in Standalone Mode (Vertical Scaling)



Automated Order Plan Development in Standalone Mode (Horizontal Scaling)



The service is located in `$OM_HOME/roles/standalone/aopd`.



The omsServer service already contains an implementation of Automated Order Plan Development, so there is no need to start the Automated Order Plan Development service. The Automated Order Plan Development service is provided for convenience to scale horizontally or vertically.

Using custom Automated Order Plan Development implementation

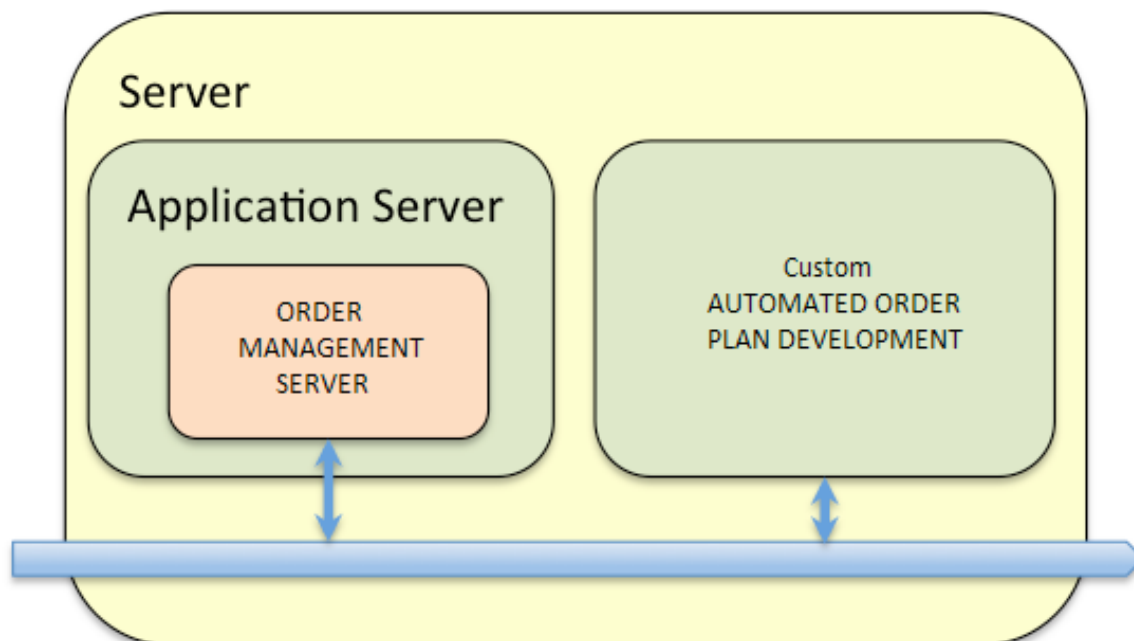
It is also possible to create a custom Automated Order Plan Development component, as long as it fulfills the Enterprise Message Service interface with the other components. In a same way, the custom Automated Order Plan Development can be deployed, either on the same machine, or a different machine.

To configure this application to use Automated Order Plan Development in custom mode, set:

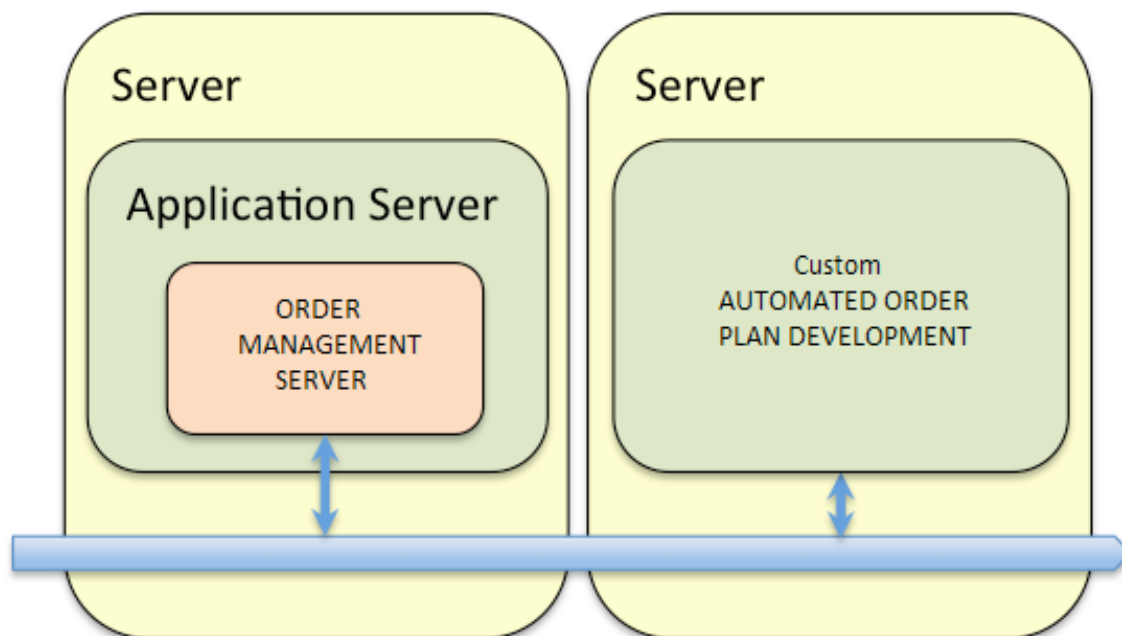
```
com.tibco.fom.aopd.deployMode=AOPD_custom
```

in the file \$OM_HOME/roles/aopd/standalone/config/profiles.properties. It tells this application to not use the Automated Order Plan Development implementation that is in the omsServer service.

Custom Automated Order Plan Development in Standalone Mode (Vertical Scaling)



Custom Automated Order Plan Development in Standalone Mode (Horizontal Scaling)



Model Deployment

Two types of models, *Product* and *Action*, have to be loaded in Automated Order Plan Development. They are loaded in Automated Order Plan Development by using Enterprise Message Service. Automated Order Plan Development has two listeners, one for each model. The Action and Product model listeners are active whether Automated Order Plan Development is deployed in standalone and collocated mode.



Additional listeners, *Execution Plan* and *Amendment* are active when Automated Order Plan Development is in standalone mode. In collocated mode, there is direct API call from AFI to Automated Order Plan Development bypassing the queues and listeners configuration for Execution plan and Amendment plan requests.

Automated Order Plan Development stores Action and Product models into the Order Management Server database (data model table) for both online and offline loading. Automated Order Plan Development, on startup, loads Action and Product models directly from the Order Management Server database.

If something is published online, Automated Order Plan Development gets the latest updates on Action and Product model listeners. If an existing product is loaded again then it is first updated in Order Management Server database, deleted from local repository and then the incoming product model is loaded in Automated Order Plan Development.

Configuration

Automated Order Plan Development configuration is stored in files in the directory `$OM_HOME/roles/configurator/standalone/config`. These files can be either manually edited or accessed from Configurator.

Main Configuration

The main Automated Order Plan Development configuration is stored at `$OM_HOME/roles/configurator/standalone/config/ConfigValues_AOPD.xml`.

Parameter Name	Description
DisableParentItemsDependencyImpact	DisableParentItemsDependencyImpact flag is used to ignore modification rule application on child plan item. In case of parent plan item modification, if flag is set to true. (default: false)
AffinityUDFNameMerge	Merges affinity user-defined field name (default: false)
CharacteristicsWithoutAffinityPostfix	To not merge certain User Defined Fields during Affinity Sequencing, those User Defined Fields must be added as CSV in the variable (default: "")
SkipItemSequence	Within Automated Order Plan Development, if the sequence is -1, it skips the product and all its mandatory children in the Execution Plan (default: -1)
MergeAffinityItemDescription	Merges affinity item description (default: false)
HierarchySingleUse	Uses unconditional removal of child product (default: false)
EnableAffinityUDFParent	Enables affinity user-defined field parent (default: false)
UDFList	List of internal User Defined Fields to be skipped for affinity merging (default: "EPMR_ACTION_PROVIDE, EPMR_ACTION_UPDATE, EPMR_ACTION_Cease, EPMR_ACTION_WITHDRAW, COMPENSATE_PROVIDE, COMPENSATE_UPDATE, COMPENSATE_Cease")
EnableBiDirectionalLinkID	Enables extended behavior for PDO/MDO and LinkID mapping (default: false)

Parameter Name	Description
AllowMultipleRequiredProducts	Multiple Required Products for the same link ID are available (default: false)
IgnorePDOFirstChildDependency	Ignores First child dependency for source product in ProductDependsOn relationship (default: false)
HandleConflict	Configurable handling for ProductComprisedOf conflict (default: "Apply")
ABDProductOrderline	Include only the order line for attribute-based decomposition (default:false)
ABDIncludeCharacteristics	Include plan item User Defined Fields for evaluating attribute-based decomposition (default:false)
CompensateRestartForNoEPMRChar	Enables COMPENSATE and RESTART behavior in case of the required Execution Plan Modification Rules characteristic that is not present in the product model.
EnableDateShiftCompRedo	Enables the backward compatibility for Date Shift amendment to generate comp redo tasks (default: false)
EnableModificationIdentifyingAttribute	Enables the backward compatibility for user defined field change amendment by using MODIFICATION_IDENTIFYING_ATTR (default: false)
NoDependencyInCOMPPlanItems	Enables the backward compatibility of NO dependency in the COMPENSATE plan item on the existing plan item being canceled.
EnableParentIDUDFCheck	The Parent_ID user-defined field check, which was not present in plans generated in earlier versions of TIBCO Order Management - Long Running, caused issues in amendment for later versions (ones with a user-defined field). A flag <code>com.tibco.af.aopd.flags.enableparentidudfcheck</code> , is introduced, which toggles checking this user-defined field during amendments. By default, the value is <code>true</code> , which enables checking of this user-defined field. Set the flag to <code>false</code> to disable the check.

Logs

The Automated Order Plan Development logs are at `$OM_HOME/roles/configurator/standalone/config/AOPDLog4j.xml`.

Integration with Orchestrator

The integration between Automated Order Plan Development and Orchestrator is configured in the file `$OM_HOME/roles/configurator/standalone/config/ConfigValues_OMS.xml`, in the category called *Orchestrator-AOPD Integration Configuration*. This configuration can be accessed and modified using the Web Configurator.

Parameter Name	Description
OPDRequest from Orchestrator receiver queue	Name of the OPDRequest from Orchestrator receiver queue (default: tibco.aff.orchestrator.provider.order.opd.request)
OPDRequest from Orchestrator receiver count	Number of the OPDRequest from Orchestrator receiver (default: 3)
ExecutionPlanNewRequest to Automated Order Plan Development sender queue	Name of the ExecutionPlanNewRequest to Automated Order Plan Development sender queue (default: tibco.aff.ocv.events.plan.new.request)
ExecutionPlanAmendRequest to Automated Order Plan Development sender queue	Name of the ExecutionPlanAmendRequest to Automated Order Plan Development sender queue (default: tibco.aff.ocv.events.plan.amend.request)
ExecutionPlanNewResponse from Automated Order Plan Development receiver queue	Name of the ExecutionPlanNewResponse from Automated Order Plan Development receiver queue (default: tibco.aff.ocv.events.newplan.reply)
ExecutionPlanNewResponse from Automated Order Plan Development receiver count	Number of the ExecutionPlanNewResponse from Automated Order Plan Development receiver (default: 3)
ExecutionPlanAmendResponse from Automated Order Plan Development receiver queue	Name of the ExecutionPlanAmendResponse from Automated Order Plan Development receiver queue (default: tibco.aff.ocv.events.amendplan.reply)
ExecutionPlanAmendResponse from Automated Order Plan Development receiver count	Number of the ExecutionPlanAmendResponse from Automated Order Plan Development receiver (default: 3)
OPDResponse to Orchestrator sender queue	Name of the OPDResponse to Orchestrator sender queue (default: tibco.aff.orchestrator.provider.order.opd.reply)

Features

Autoprovision

Autoprovision is a condition that determines the relationship between a parent product and a child product. The relationship can be either be *Static* or *Dynamic*. Autoprovision flag determines whether the product is mandatory or not. For instance, consider a product A having a child product B, which has AutoProvision set as TRUE.

Static: If Autoprovision is set to TRUE, then the product is considered to be a static bundle and the child is automatically be assumed to be a part of the order. This indicates that the child product is implicitly assumed to be on the order no matter there is an order line with that product.

Example: Consider bundle B comprised of products P1 and P2.

- AUTOPROVISION flag is TRUE between B and P1.
- AUTOPROVISION flag is FALSE between B and P2.

When bundle B is ordered, P1 is provisioned automatically though it is not in the order line.



This process of order fulfillment is known as *Implicit Fulfillment*.

Dynamic: Auto Provision is set to FALSE. This indicates that the child product must be explicitly included in the order.

The [product diagram](#) shows the mandatory products with solid lines (Autoprovision TRUE, Static bundle) and are included in the order automatically. Otherwise, the products have to be ordered separately, which is a Dynamic bundle feature.



The instanceOptional field is not used during plan generation in Automated Order Plan Development.

Dynamic Bundles

Dynamic Bundles lets for a bundle to be modeled by using a product hierarchy in a product catalog and items are selected by the user and then submitted for order plan development. An example would be where a bundle is modeled to have mandatory items and optional items and the customer needs to select the options.

The *optional* products are specified as specific order lines within the order. The bundle is also specified as an order line but the decomposition component recognizes the options belonging to the parent bundle.

The *mandatory* products are automatically added for order planning.

Any *required* products is validated as part of validation to ensure the basket or the customer image has the required product before any decomposition occurs.

It is possible to reuse the common products having Autoprovision=false using LinkParentID and LinkedParentID User Defined Fields in the order line. The LinkParentID-LinkedParentID and LinkID User Defined Fields are used to define PCO-tree, although the LinkParentID-LinkedParentID user-defined field has the higher priority.

- Link child to parent based on LinkParentID-LinkedParentID if present. Else,
- Link child to parent based on LinkID if present. Else,
- Link child to parent randomly.

For example, consider the following product model:

```
T-COM Wireline --> (PCO) Additional Voice Service(autoprovision=false)
T-COM Wireline --> (PCO) Tarrif1(autoprovision=false)
Additional Voice Service --> (PCO) Tarrif1(autoprovision=false)
```

Therefore, the order can be:

Orderline Number	Product	LinkParentID	LinkedParentID
1	T-COM Wireline	T-COM Wireline	
2	Additional Voice Service	Additional Voice Service	T-Com Wireline
3	Tarrif1	Tarrif1	Additional Voice Service
4	Tarrif1	Tarrif1	T-Com Wireline

The LinkParentID-LinkedParentID User Defined Fields are added for order line number 3 in the following format to add dependency between OrderLine 2 and OrderLine 3:

```
<ord1:udf>
```



```

<ord1:name>LinkParentID</ord1:name>
<ord1:value>Tarri1</ord1:value>
</ord1:udf>
<ord1:udf>
  <ord1:name>LinkedParentID</ord1:name>
  <ord1:value>Additional Voice Service</ord1:value>
</ord1:udf>

```



This change is backward-compatible. To use the LinkID functionality, do not add the LinkParentID-LinkedParentID User Defined Fields in the order line.

Static Bundles

Through Static Bundles, a bundle to be modeled by using a product hierarchy in the catalog, with the bundle only containing mandatory options. An example of this would be a bundle with mandatory products and when a customer orders this bundle all the dependent products are provisioned without the customer needing to selecting any more products.

Time Dependency

The decomposition component has the ability to provision an execution plan for a given order based upon the time constraints, if any, placed on the products within that order. Example: it determines when a particular product execution must be started. This time constraint could apply to an individual plan item within an execution plan or to the entire execution plan. Time dependency are added in each plan item if the requiredByDate specified in order is in future.

Time dependency defines the absolute time when the particular plan fragment starts execution. It is calculated on the basis of requiredByDate present in either the Orderheader or OrderLine. The expected behavior for the required by date is as follows

1. If requiredByDate is set on the order level, the start time dependency applies to all plan items with no leading dependencies
2. If requiredByDate is set on the order line level only, the start time dependency applies to plan items for that order line, which have no leading dependency
3. If requiredByDate is set on the order header level and on the order line level, the following behavior applies:
 - a. If requiredByDate in Order Header is later than requiredByDate in line item, then the start time used is the one at order level
 - b. If requiredByDate in Order Header is earlier than requiredByDate in line item, then the start time used is the one at order line level.



RequiredOnDate is no longer used or supported.

Product Specification Field Decomposition

Each product has a modeled set of characteristics within a product catalog. When a product is decomposed to a plan item, the default and the instance characteristics are copied over into the User Defined Fields (UDFs) of every plan item. Through this, the information is reused later when the plan item is executed.

For example, consider a product "Line Access 5MB" has characteristics modeled such as Speed=5, QOS=4, IPAccess=false. These are all modeled as instance variables. When an order is submitted for Line Access or is part of a bundle, the plan item uses the same instance characteristics copied as User Defined Fields into the plan item. When the plan item is executed, the User Defined Fields can be passed to the service call.

When an order is made the characteristics are visible as User Defined Fields for each order line. When you submit the order, the User Defined Fields are converted into User Defined Fields for the new plan items and if the order line is a bundle then those items can have User Defined Fields as well, which are copied to the execution plan. All these User Defined Fields can be used later through the service call.

Custom Action Based Product Decomposition

The custom action provides flexible way to define products and product fulfillment by allowing product decomposition and characteristic list inclusion. The ProductComprisedOf (henceforth, referred to as PCO) relationship enables you to model complex product hierarchies. This allows a product modeler to model specific product decomposition according to the specified action.



Irrespective of an action, all the PCO or Characteristic relationships are valid.

The following table describes the custom action for the PCO and Characteristic relationships:

ProductComprisedOf (PCO)		Characteristic (C)	
If PCO.ActionID=null	The child product is always a part of the decomposition during decomposition	If C.ActionID=null	The characteristic is always included as planItem User Defined Fields
If PCO.ActionID=not null	The child product is only added if the following order action is specified during decomposition: order Action = the ActionID	If C.ActionID=not null	The characteristic is included if the following order action is specified during decomposition: order Action = ActionID

Scenario for the Custom Action Based Product Decomposition

The following table describes how a custom action impacts the product decomposition:



This scenario is applicable for characteristic list inclusion based on custom action.

Data Model Configuration	Order	Plan
Action repository has record with ID as HomeMove and recordtype as PROVIDE. Product B has PCO relationship with P1, P2, P3 with autoprovision=true P1.PCO.ActionID=null P1.PCO.ActionID=PROVIDE P1.PCO.ActionID=HomeMove	OL=B(PROVIDE)	There are three planItems: <ul style="list-style-type: none"> • B • P1 • P2 B depends on P1 & P2
	OL=B(UPDATE)	There are two planItems: <ul style="list-style-type: none"> • B • P1 B depends on P1
	OL=B(HomeMove)	There are two planItems: <ul style="list-style-type: none"> • B • P3 B depends on P3

Sequencing

The product catalog defines the sequencing requirements between the fulfillment steps for products in a product offering.

When the order plan is being developed, the information in the product catalog is used such that the instance sequence defined for each sub-product and products, which contain these sub-products translates to a dependencies between *Plan Fragments* associated with each product/sub-product and the fulfillment happens in the correct sequence.

Sequencing is governed by certain rules on the Plan Fragments.

Order Management supports the action-based sequencing. This use case based sequence of back-end systems are utilized in the decomposition to ensure back-end (process component) are called in the correct order. Based on the order line action, the following types of sequencing are used:

- PROVIDE
- CEASE
- UPDATE

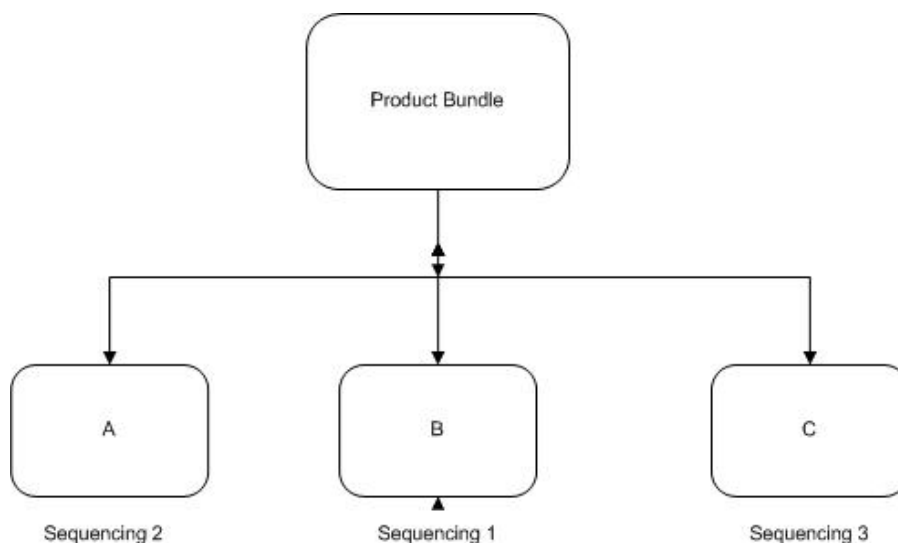
PROVIDE Sequencing: This scenario occurs when the order line action is PROVIDE and all the sub-products use the provide instance sequence number.

CEASE Sequencing: This scenario occurs when the order line action is CEASE and all the sub-products use the cease instance sequence number.

UPDATE Sequencing: This scenario occurs when the order line action is UPDATE and all the sub-products use the update instance sequence number.

The following figure shows the sequencing for the products A, B, and C.

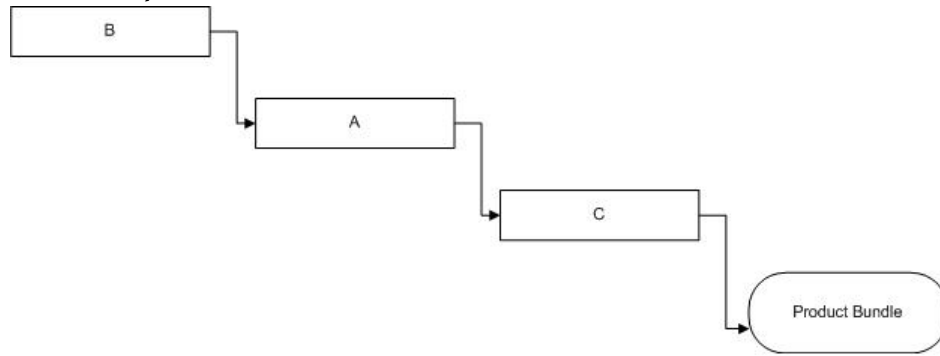
Sequencing for the products A, B, and C.



Sequence number is the relationship attribute value based on the actions PROVIDE, CEASE, and UPDATE.

For example, a bundle is comprised of Product A, Product B, and Product C, with PROVIDE sequencing set to 2, 1 and 3 respectively. When an order plan is developed, Product B is executed first, followed by Product A and then Product C.

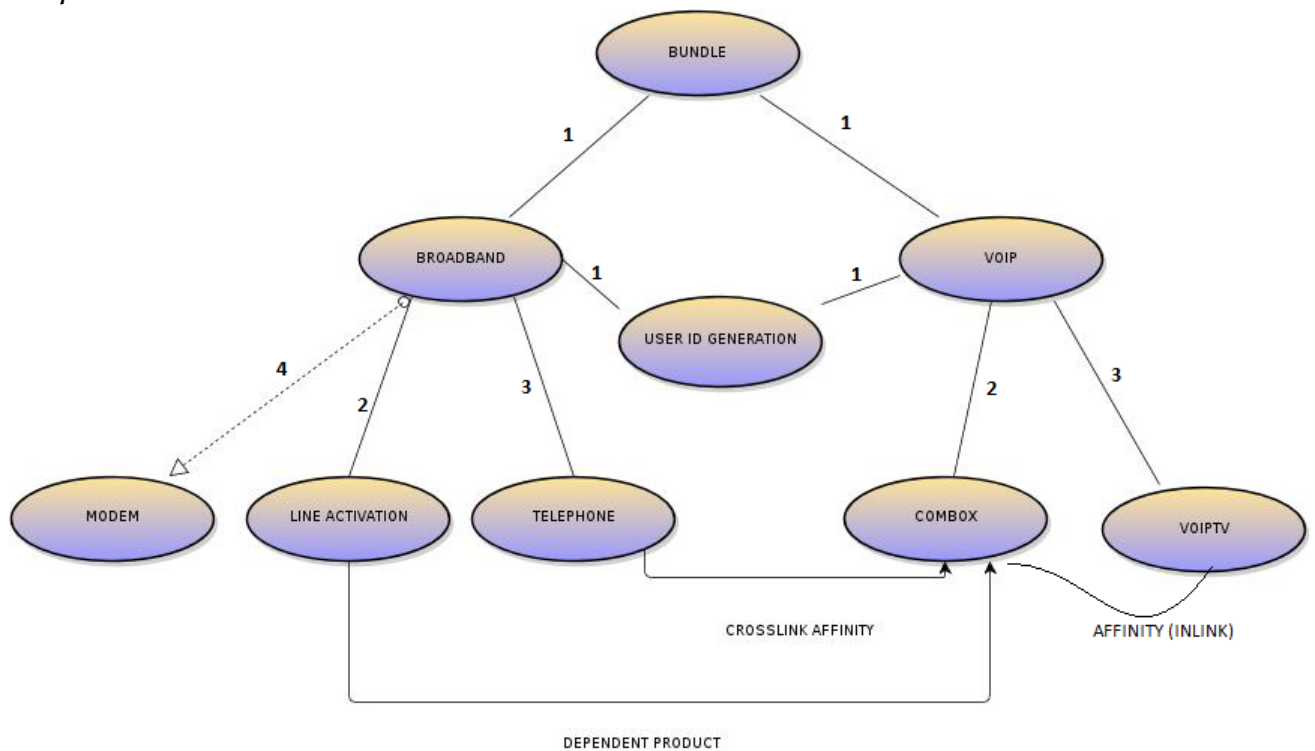
Order Plan Execution Sequence



Similarly, a CEASE sequencing order can also be defined for the same Product Bundle with a sequencing of 3, 1, and 2 for products A, B, and C respectively. In this manner, the order might be fulfilled in the correct sequence taking into account what action needs to be performed.

The Order lines are converted into plan items during the plan development by using the information in the product catalog. The diagram explains the sample product model and its components (product offerings). This diagram is used to briefly explain the different Plan Development Concepts (for details, see [Automated Order Plan Development](#)).

Sample Product Model



The table describes the diagram elements of the Product Model Hierarchy.


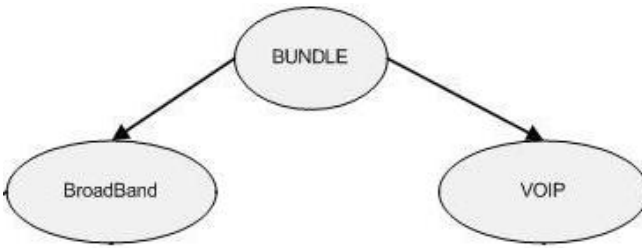
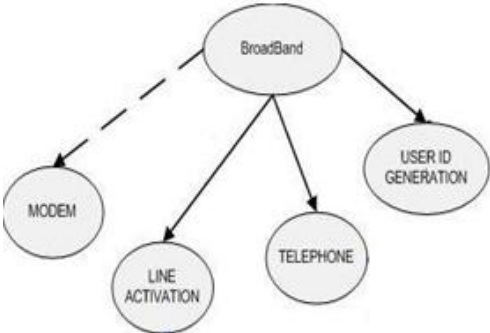
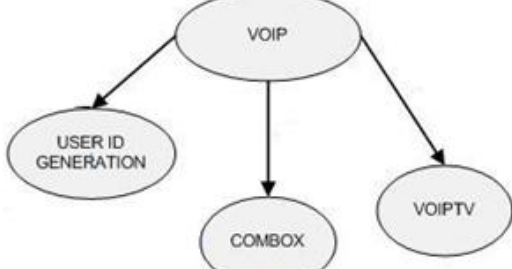
Diagram Element	Description
	Product entity.

Diagram Element	Description
 <pre> graph TD BUNDLE --> BroadBand BUNDLE --> VOIP </pre>	<p>The arrows represent the <i>ProductComprisedOf</i> relationship in the product catalog between BUNDLE and a group of products. Thus, the diagrams state that:</p> <ul style="list-style-type: none"> • BUNDLE is comprised of the product BroadBand. • BUNDLE is comprised of the product VOIP.
 <pre> graph TD BroadBand -.-> MODEM BroadBand --> LINE_ACTIVATION BroadBand --> TELEPHONE BroadBand --> USER_ID_GENERATION </pre>	<p>The Broadband product offering contains the following mandatory products:</p> <ul style="list-style-type: none"> • Telephone • UserID • Line Activation <p>The dotted line indicates that the Modem is an optional product.</p>
 <pre> graph TD VOIP --> USER_ID_GENERATION VOIP --> COMBOX VOIP --> VOIPTV </pre>	<p>The VOIP product offering contains the following mandatory products:</p> <ul style="list-style-type: none"> • VOIPTV • COMBOX • UserID <p>The UserID products has two parents.</p>

Product Model Description:

The product BUNDLE is comprised of the two product offerings:

- Broadband.
- VOIP.

The Broadband product offering contains the products as the Telephone, UserID, and Line Activation as the mandatory product. Modem is an optional product.

VOIP has the COMBOX, UserID, and VOIPTV as part of its technical products.

The product UserID, here has two parents - Broadband and VOIP. The product UserID has single use record attribute set to true with both its parents.

Using (relationship) sequencing, all the child products of the BUNDLE would be fulfilled or processed in parallel, and all must complete before the entire BUNDLE can be fulfilled. Often, additional sequencing is required within elements at the same hierarchy level in the model. This can be accomplished by providing sequence numbers on the *ProductComprisedOf* relationship.

Product Model Description with the Sequencing Feature:

The correct fulfillment sequencing of the product plan execution as per the diagram is:

1. The UserID is created.
2. The order on the Telephone and COMBOX is processed. The Telephone and COMBOX are installed.
3. The Line Activation is completed.
4. Modem is installed.
5. VOIPTV is installed.
6. Broadband Product Order and VOIP Order is completed.
7. The entire BUNDLE order (Broadband and VOIP) is completed.

Taking the product as an example, the table shows the sequencing of the products:

Parent Product	Product Offering
BUNDLE	UserID
BUNDLE	Telephone/COMBOX
BUNDLE	Line Activation
BUNDLE	Modem Installation (optional)
BUNDLE	VOIP
BUNDLE	BUNDLE Order complete

Delta Provisioning

Delta Provisioning ensures that products, which have been defined for 'single use' are not provisioned more than once for a given order. The combination of the order line action of the products is used to determine how the products are provisioned.

Single Use

Single Use ensures that if the products have the same product ID and have been defined for single use with the order line actions as PROVIDE then those products are not provisioned more than once. It deletes one of the instances and ensures that the dependencies point to the single instance, which remains in the plan. This is done for products with the same parent only.

E.g. for a batch of phones typically we would only want to only send one shipment.

Product Model description in relation to Single Use:

The [Product Model diagram](#) shows the Single Use feature. If the Order is a 'BUNDLE in a single Order line', the UserID is generated only once, although it has been ordered twice by the products Broadband and VOIP respectively.

If the product exists more than once on the order, then it is only included once in the final plan. If the product exists on the order and in the inventory, it is not included in the plan.

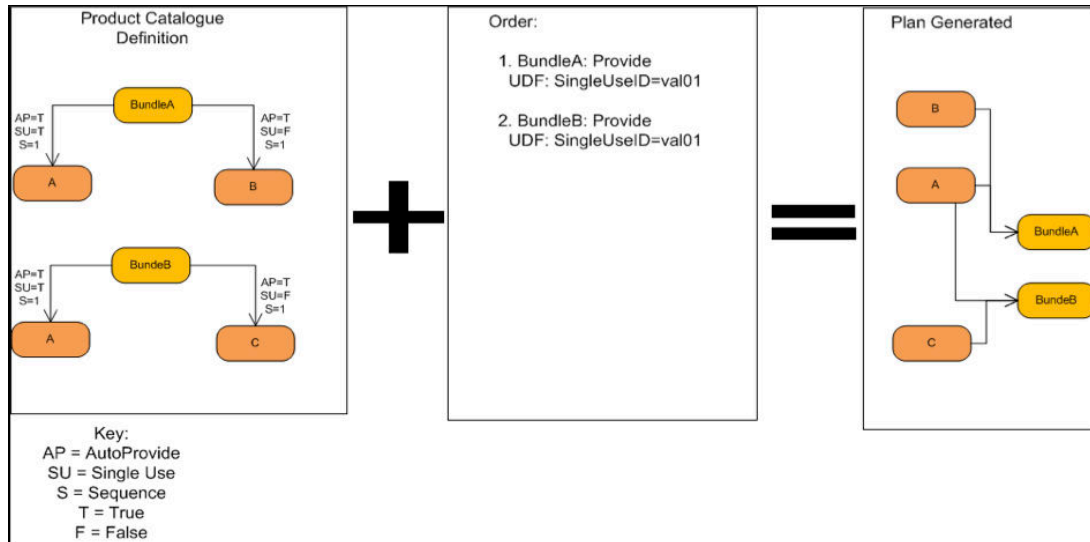
Provide Single Use

The product catalog contains 2 bundles BundleA and BundleB. BundleA contains 2 sub-products A and B. Both the sub-products have sequence set to "1" and auto provision set to "True". A has the attribute single use set to "True" when B has the attribute set to "False". BundleB contains 2 sub-products A and C. Both the sub-products have sequence set to "1" and auto provision set to "True". A has the attribute single use set to "True" when C has the attribute set to "False".

The order sent into AFF contains 2 order lines. Order line 1 contains BundleA with order line action Provide. Order line 2 contains BundleB with order line action Provide. Both the order lines contain a user-defined field with name SingleUseID and the value is the same for both BundleA and BundleB.

The generated plan contains only one instance of sub product A. BundleA, which contains a dependency to sub product A and B. BundleB contains a dependency to sub product A and C. The User Defined Fields is not merged into the retained product.

Single use (Provide-Provide)



Cease Single Use

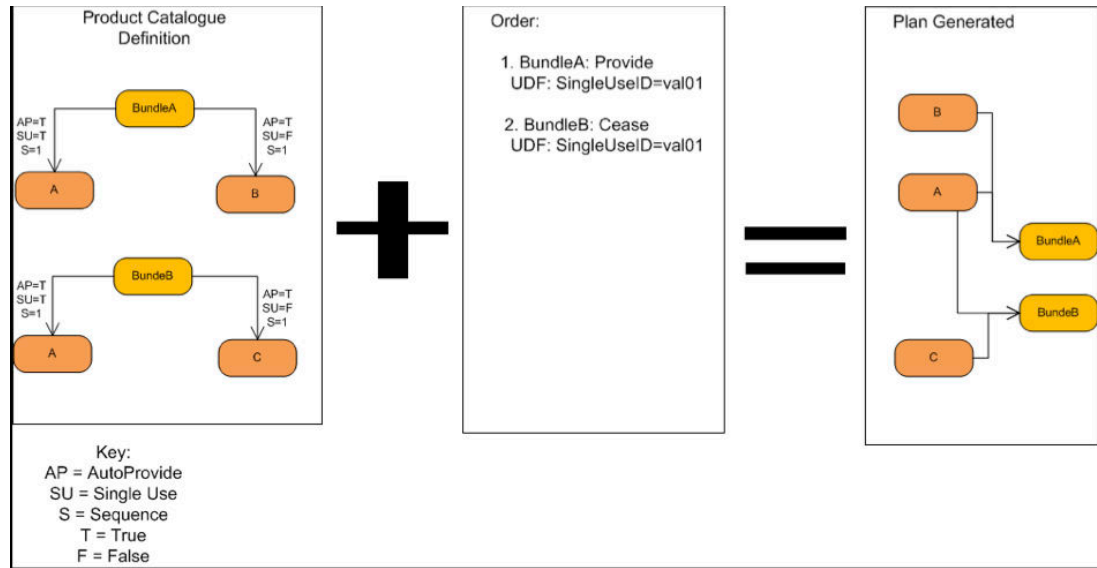
This functionality ensures that if the products have the same product ID and have been defined for single use with their order line actions as PROVIDE and CEASE then those products are not provisioned more than once. It deletes instance, which has its order line action as CEASE, mark the action as UPDATE and also ensure that the dependencies point to the PROVIDE instance, which remains in the plan. This is done for products with the same parent only. Single Use is modeled in product model by setting record attribute single use as true.

In this scenario the Cease instance of the product is removed from the plan. Bundle A has a dependency to both sub product A and B. BundleB has a dependency to both sub product A and C. The instance of A still left in the plan has a new line action of UPDATE. The User Defined Fields is not merged into the retained product.

The plan fragment and the plan description is set to the Update fragments from the product information.

In cases where the sub product A has dependent products all those dependent products is made dependent to the remaining instance of product A.

Single use (Provide-Cease)



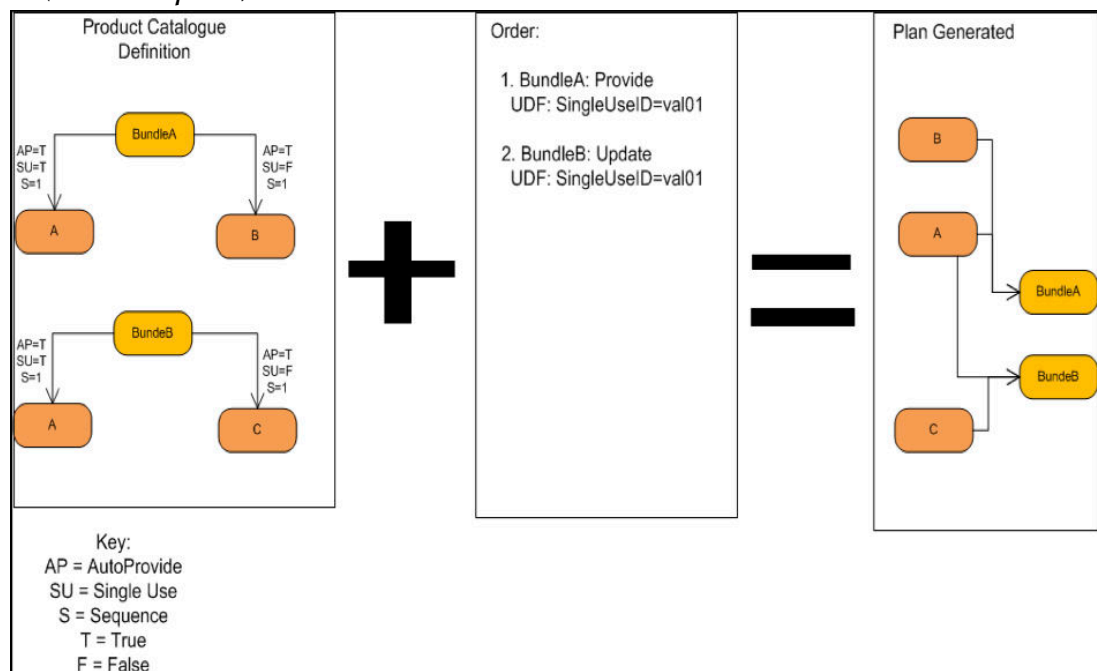
Update Single Use

This functionality ensures that if the products have the same product ID and have been defined for single use with their order line actions as PROVIDE and UPDATE then those products are not provisioned more than once. It deletes instance, which has its order line action as PROVIDE and also ensure that the dependencies point to the UPDATE instance, which remains in the plan. This is done for products with the same parent only.

In the following scenario the Update instance of sub product A remains in the plan. Bundle A has a dependency to both sub product A and B. BundleB has a dependency to both sub product A and C. The instance of product A retains the line action of UPDATE. The User Defined Fields is not merged into the retained product.

The plan fragment and the plan description is set to the Update fragments from the product information

Single use (Provide-Update)

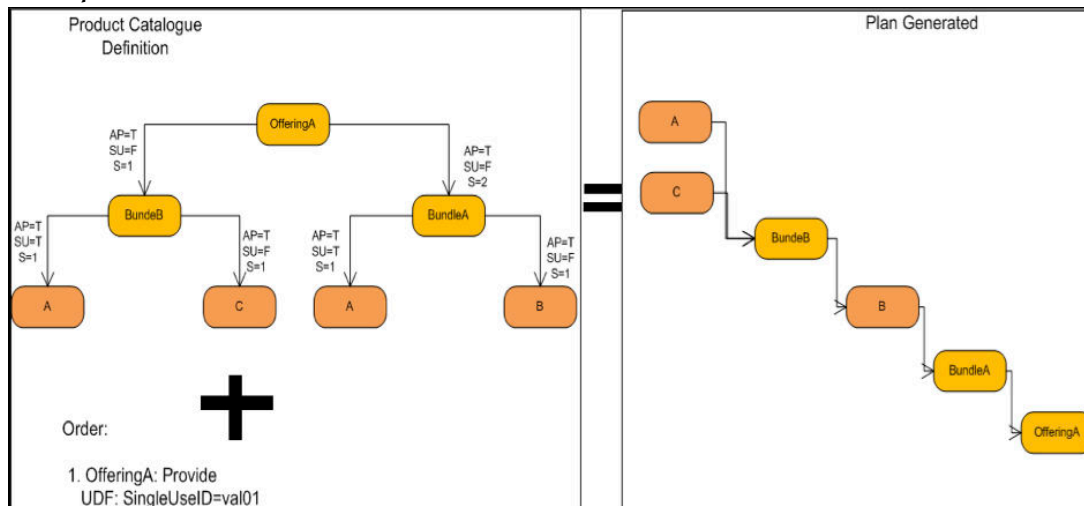


Sequenced Single Use

In this scenario OfferingA contains both BundleA and BundleB, which have been sequenced 2 and 1 respectively. Since both bundles contain sub product A, this product is merged into a single instance. BundleB is the only product to contain a dependency to sub product A since it is the 1st product to be provisioned in the plan. BundleA has the dependency to sub product A deleted.

The User Defined Fields is merged as mentioned in the above scenarios. The lineIDs and EOL attributes are merged as well with a comma as a separator.

Single use (Sequenced)



Unconditional Removal of Child Products

The hierarchy single use flag in `ConfigValues_AOPD.xml` enables the unconditional removal of child products from the Plan. To enable this functionality, set the following flags:

1. `HierarchySingleUse = true` (in Automated Order Plan Development).
2. Merge inventory in Automated Order Plan Development request = true (in the Order Management Server user interface Configurator/ Member1/Order Management Server AFI Configuration/Orchestrator-Automated Order Plan Development Integration Configuration).

The `HierarchySingleUse` flag needs to be set for the Inventory Single Use functionality to work irrespective of Unconditional Removal of products. Unconditional removal of products removes all the child products of a product deleted due to single use. The `MergeInventory` flag must be set to true even if products are removed conditionally for inventory single use.

After implementing these settings, the `HierarchySingleUse` use case works for `SingleUse` and `InventorySingleUse` cases.



The `HierarchySingleUse` global variable is disabled by default (`HierarchySingleUse = false`) to continue with the default implementation. The `HierarchySingleUse` flag works for all Single Use cases i.e. `SingleUse`, `CeaseSingleUse`, `UpdateSingleUse` and `InventorySingleUse`.

TIBCO Order Management provides an inbuilt integration with TIBCO Fulfillment Subscriber Inventory. To use this component, configure the properties in the `ConfigValues_OMS.xml` file described in "Post Installation Task: Integrating with TIBCO Fulfillment Subscriber Inventory" in the *TIBCO Order Management - Long Running Installation and Configuration* guide.

Product Affinity (Plan Item Level)

Through the Product Affinity between different products on the same order, the products to be grouped and fulfilled together through the execution of a single plan item occur. It can be termed as an order fulfillment optimization.

Generally, a plan item corresponding to an order line specifies a product to be fulfilled in the order. If an affinity is specified between the products that are either being fulfilled implicitly as mandatory children, or

ordered explicitly as separate order lines, the individual plan items are grouped together into a single affinity plan item during plan optimization in Automated Order Plan Development. Thus, the corresponding products are fulfilled through the execution of this single plan item.

The product affinity is specified in the product catalog in one of the following two different ways:

- By specifying the affinity type and action-specific plan fragments attributes in the AffinityGroup tab in PRODUCT repository
- By assigning the plan fragments by using ProductHasXXPlanFragment relationships and specifying the affinity specific relationship attributes

The XX in relationship name refers to actions, such as PROVIDE, CEASE, UPDATE, and CANCEL.

Automated Order Plan Development recognizes the affinity and combines the plan items corresponding to the order lines depending on the following two main conditions:

- If the plan fragments defined in the product catalog for the ordered products are the same
- If the affinity type defined in the product catalog for the ordered products is the same (InLink or CrossLink)

user-defined field Data Handling

Affinity groups together plan items for different order lines into a single plan item. Automated Order Plan Development is also responsible for populating the User Defined Fields that are associated with these plan items. The potential exists for the same user-defined field to be present on different order lines, all values must be available in the plan and the relevant order lines identified.

The following data handling rules must be implemented:



The following table lists the Sample Order Line Data representing the order lines being affinity grouped. The Sample Plan Item Data represents the output affinity grouped plan item for those order lines.

Sr No	Rule	Outcome	Sample Order Line Data	Sample Plan Item Data
1	user-defined field exists on only one of the order lines being affinity grouped	user-defined field name is the original user-defined field name concatenated with the order line number. Value is the original user-defined field value.	Order Line = 1 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 2 <i>Does not contain ServiceID user-defined field</i>	user-defined field Name = ServiceID:1 user-defined field Value = 1234

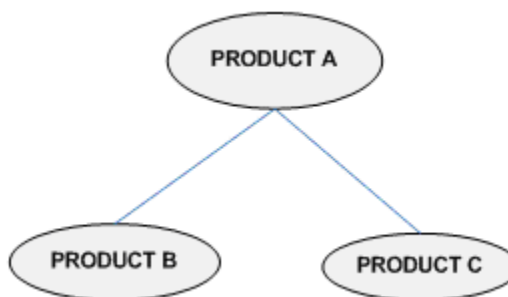
2	user-defined field exists on more than one of the order lines being affinity grouped, but not all order lines. user-defined field value is the same on all order lines.	user-defined field name is the original user-defined field name concatenated with the order line number as a comma-separated list. Value is the original user-defined field value.	Order Line = 1 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 2 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 3 <i>Does not contain ServiceID user-defined field</i>	user-defined field Name = ServiceID:1,2 user-defined field Value = 1234
3	user-defined field exists on all order lines being affinity grouped. user-defined field value is the same on all order lines.	user-defined field name is the original user-defined field name. Value is the original user-defined field value.	Order Line = 1 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 2 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 3 user-defined field Name = ServiceID user-defined field Value = 1234	user-defined field Name = ServiceID:1,2,3 user-defined field Value = 1234
4	user-defined field exists on more than one order line being affinity grouped. user-defined field value is different on different order lines.	user-defined field is created for each unique user-defined field value, with the corresponding name containing the original user-defined field name concatenated with the order line numbers as a comma-separated list.	Order Line = 1 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 2 user-defined field Name = ServiceID user-defined field Value = 1234 Order Line = 3 user-defined field Name = ServiceID user-defined field Value = 6789	user-defined field Name = ServiceID:1,2 user-defined field Value = 1234 user-defined field Name = ServiceID:3 user-defined field Value = 6789

TIBCO Order Management - Long Running supports the following types of product affinities:

Inlink

The *Inlink Affinity* can be defined between the products at the same level in a bundle.

Inlink Affinity



As shown in the figure, the InLink affinity can be defined between the Product B, and Product C for PROVIDE action by specifying the affinity type as **InLink**. The PROVIDE plan fragment is defined as **PC_PROVIDE_BC**.

For the InLink affinity, the **LinkID** user-defined field having the same value must be passed in the order lines.

In addition to the two conditions, Automated Order Plan Development also checks the following conditions for the InLink affinity:

1. If a value of the **LinkID** user-defined field in the plan items, which is propagated from the order lines to be merged, is exactly the same.
2. If the dependentOn (parent product) plan item for the plan items to be merged is exactly the same.

If these conditions are fulfilled, the plan items are combined into a single *affinity plan item* containing the plan fragment from any of the merging plan items, since it is the same for all of them.

Crosslink

The Crosslink Affinity is defined between the products at any levels across the bundles.

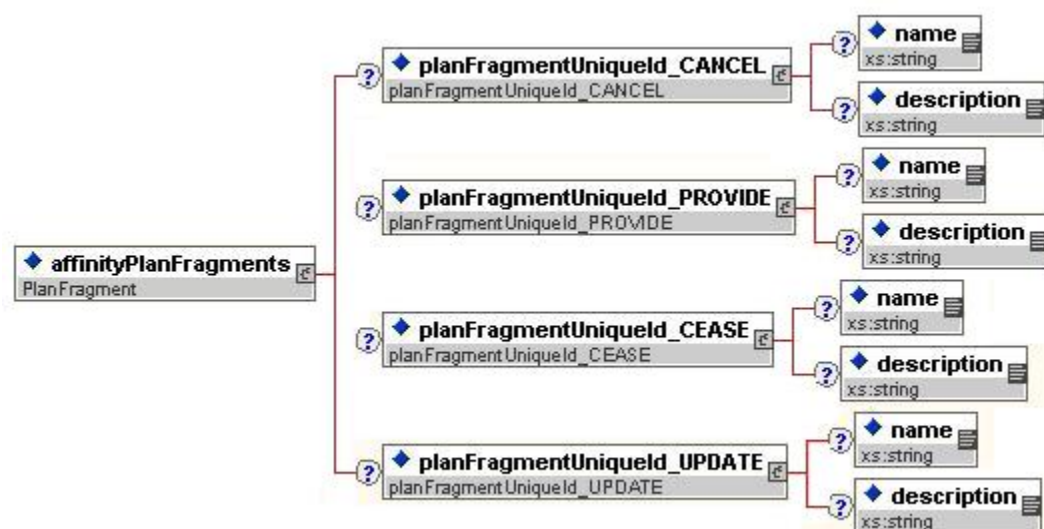
In the [Product Model diagram](#), the products **Telephone** and **COMBOX** can have CrossLink affinity between them. when order fulfillment process, both these technical products would be installed and configured in one go through the affinity grouped single plan item.



Automated Order Plan Development does not check any additional conditions for CrossLink affinity.

Affinity applies to the order plan development and this element is used to determine what plan fragments are executed for the product when affinity grouping is active. Affinity Plan Fragments XSD is illustrated as:

Affinity Plan Fragments XSD



The following table explains the Affinity Plan Fragments Data Model.

Element Name	Element Type	Description	Example
planFragmentUniqueId_CANCEL	String (Optional)	Plan fragment cancel type.	
planFragmentUniqueId_CANCEL/ name	String (Optional)	Name of the plan fragment to execute when canceling this product.	EP_BUNDLE_CANCEL NO_RECIPROCAL_ACTION
planFragmentUniqueId_CANCEL/ description	String (Optional)	Description of the plan fragment to execute when canceling this product.	Product 1 Cancel
planFragmentUniqueId_PROVIDE	String (Optional)	Plan fragment provide type.	
planFragmentUniqueId_PROVIDE / name	String (Optional)	Name of the plan fragment to execute when providing this product.	EP_BUNDLE_PROVIDE
planFragmentUniqueId_PROVIDE / description	String (Optional)	Description of the plan fragment to execute when providing this product.	Product 1 Provide
planFragmentUniqueId_CEASE	String (Optional)	Plan fragment cease type.	

planFragmentUniqueId_CEA SE/ name	String (Optional)	Name of the plan fragment to execute when ceasing this product.	EP_BUNDLE_CEASE
planFragmentUniqueId_CEA SE / description	String (Optional)	Description of the plan fragment to execute when ceasing this product.	Product 1 Cease
planFragmentUniqueId_UPD ATE	String (Optional)	Plan fragment update type.	
planFragmentUniqueId_UPD ATE/ name	String (Optional)	Name of the plan fragment to execute when updating this product.	EP_BUNDLE_UPDATE
planFragmentUniqueId_UPD ATE/ description	String (Optional)	Description of the plan fragment to execute when updating this product.	Product 1 Update

Affinity Sequencing

Affinity Sequencing is used to support the scenario for maintaining sequencing during affinity grouping. Affinity Sequencing was introduced during affinity RulesEngine (BusinessEvents) selects plan items at random, which are then merged into a single plan item. Since items are selected at random during this process, sequencing is not maintained for plan items that must be in a sequence.

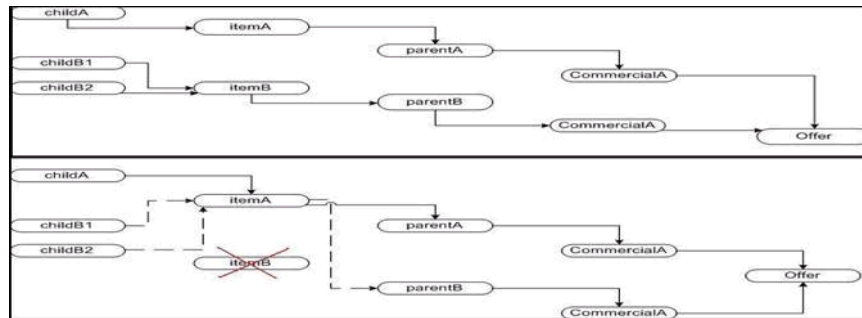
To make products available for affinity sequencing:

- Affinity TYPE value for all products where sequence must be respected must be set to "SequencedAffinity" in the affinity type
- All order lines where affinity components are known to exist must have a user-defined field defined as AffinitySequence and the value must be an integer

Depending on the AffinitySequence values being compared, the following actions are possible:

1. itemA.AffinitySequence = itemB.AffinitySequence
 - If both items have dependent children the children from itemB is copied to itemA
 - itemB parent is updated with the plan item ID from itemA, thus making itemA dependent to its own parent and the itemB parent
 - user-defined field values from itemB is merged into itemA
 - Any item, which has a reference to itemB have that reference replaced with a reference to itemA
 - The instance of itemB is deleted from the plan

Parallel Scenario

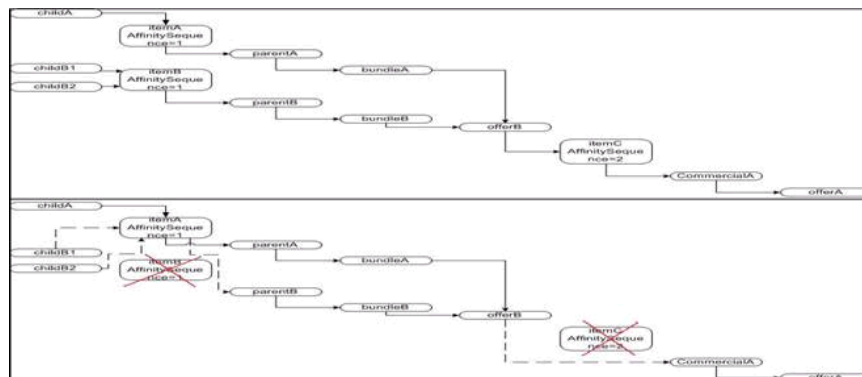


The figure depicts the scenario where the items to be affinity grouped are running in parallel. One of the items in this case itemB is deleted from the plan. The dependent items to itemB, which are childB1 and childB2 are made dependent to itemA. Then itemA is made dependent to parentB, which is the parent to itemB.

2. $\text{itemA.AffinitySequence} < \text{itemB.AffinitySequence}$

- itemB is merged into itemA
- user-defined field values from itemB is merged into itemA
- Any item, which has a dependency to itemB have that reference removed
- all the children from itemB is made dependent to itemB parent(s)
- itemB is deleted from the plan

Sequenced Scenario

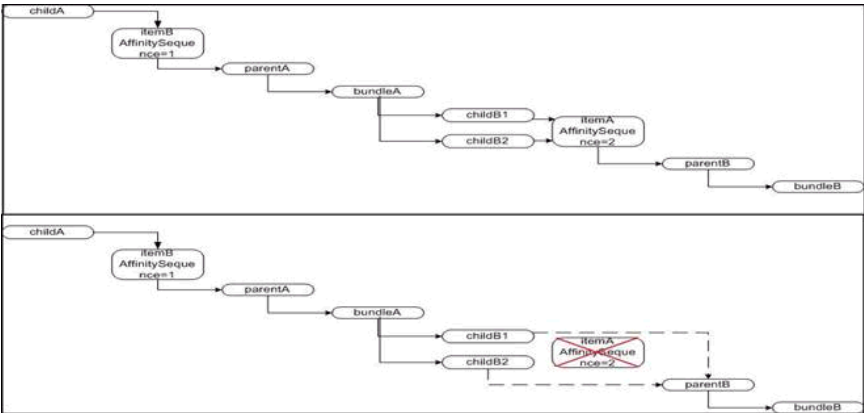


The figure depicts an offering, which has items that are in parallel and in sequence that have to be affinity grouped. For items that are in parallel they are handled similar to the figure 1. For the item that is in sequence itemC. It is dependent item offerB is made dependent to CommercialA, which is the parent to itemC.

3. $\text{itemA.AffinitySequence} > \text{itemB.AffinitySequence}$

- itemA is merged into itemB
- User Defined Field's from itemA is merged into itemB
- if itemA has dependent children those children are copied into itemB and the parent item of itemA
- Any item, which has a reference to itemA have that reference erased
- itemA is deleted from the plan

Items in Sequence



For all the above actions the following occurs in all of them:

- EOL, Plan description and Line ID values are merged into comma-separated values from itemA and itemB
- The planID is updated with the affinity plan ID



When an order is submitted, the order lines for items, which have Affinity must have the AffinitySequence defined and updated.



To not merge certain User Defined Fields during Affinity Sequencing, those User Defined Fields must be added as a comma-separated values in the global variable CharacteristicsWithoutAffinityPostfix in the rules engine (Automated Order Plan Development).

Conditional Affinity

Conditional Affinity combines InLink and CrossLink affinities in a single affinity type and provides additional flexibility. Affinity grouping enables different plan items to be grouped together based on the evaluation of the XPATH expression defined at the product catalog. The two affinity grouping types are:

Inlik Affinity	Crosslink Affinity
Affinity groups plan items having the same parent product share a common LinkID user-defined field value and have the same affinity plan fragment name	Affinity groups plan items having the same affinity plan fragment name

The additional configuration fields and rules in conditional affinity are:

Field	Description
AffinityType	Determines the type of affinity implemented. <ul style="list-style-type: none">• InLink• CrossLink• Sequenced Affinity• ConditionalAffinity

Field	Description
AffinityCondition	<p>Valid for Conditional type only. A String field containing an XPATH expression that evaluates to true or false based on data is in the order:</p> <ul style="list-style-type: none"> • If the expression is true, the product plan item is affinity-grouped • If the expression is false, then the product plan item is not affinity-grouped • If the field is blank, assume that the value is true • If the XPATH expression evaluates to anything other than the true or false, Automated Order Plan Development fails and returns an exception <p>The XPATH expression evaluates against the following data fields on the order:</p> <ul style="list-style-type: none"> • Order Header user-defined field Name and Value • Order Line ProductID • Order Line Action and ActionMode • Order Line user-defined field Name and Value <p>The XPATH expression can also be defined against the following plan data fields:</p> <ul style="list-style-type: none"> • planItem productID • planItem user-defined field name value • planItem Action

Field	Description
AffinityCorrelation	<p>Valid for Conditional type only. The XPATH is evaluated on the Plan data and the order data. A String field containing an xpath expression based on a data is in the following order:</p> <ul style="list-style-type: none"> • All plan items that evaluate to the same AffinityCorrelation are grouped together • The field is functionally similar to the LinkID method of correlating plan items in the InLink affinity. However, it lets correlation based on complex conditions without a restriction on the user-defined field names • If the field is blank, a default LinkID value is shared by all other blank configurations • If the XPATH expression evaluates to an empty string, the XPATH expression is blank, or assume a default LinkID <p>The XPATH expression evaluates against the following order data fields:</p> <ul style="list-style-type: none"> • Order Header user-defined field Name and Value • Order Line ProductID • Order Line Action and ActionMode • Order Line user-defined field Name and Value <p>The XPATH expression can also be defined against the following plan data fields:</p> <ul style="list-style-type: none"> • planItem productID • planItem user-defined field name value • planItem Action
AffinityParentGroup	<p>Valid for Conditional type only. A Boolean field containing the value true or false:</p> <ul style="list-style-type: none"> • If set to true, the plan items with products sharing the same immediate parent product are grouped together • If set to false, the parent product is not considered for grouping
AffinityActionGroup	<p>Valid for Conditional type only. A Boolean field containing the value true or false:</p> <ul style="list-style-type: none"> • If set to true, then only plan items with products that share the same action are grouped together • If set to false, then the action is not considered for grouping

Field	Description
AffinityActionValue	<p>AffinityActionValue is considered for grouping when AffinityActionGroup is set to true. This is valid for Conditional type only. String field containing an XPATH expression that evaluates to a String based on data is in the following order: The XPATH expression must evaluate to one of the following:</p> <ul style="list-style-type: none"> • PROVIDE • UPDATE • CEASE • Empty String <p>If the XPATH expression evaluates to anything other than these actions, then Automated Order Plan Development fails and returns an exception.</p> <ul style="list-style-type: none"> • If the field is blank, or the return value from the XPATH expression is an empty string, the remaining action rules must be applied. <p>The XPATH expression can evaluate against the following data fields on the order:</p> <ul style="list-style-type: none"> • Order Header user-defined field Name and Value • Order Line Action and ActionMode • Order Line user-defined field Name and Value <p>The XPATH expression can also be defined against the following plan data fields:</p> <ul style="list-style-type: none"> • planItem productID • planItem user-defined field name value • planItem Action
AffinityProvide	Provide plan fragment name for affinity grouped plan item. Only plan items with the Provide action and the same value in this field are grouped together
AffinityUpdate	Update plan fragment name for affinity grouped plan item. Only plan items with the Update action and the same value in this field are grouped together
AffinityCease	Cease plan fragment name for affinity grouped plan item. Only plan items with the Cease action and the same value in this field are grouped together
AffinityCancel	Cancel plan fragment name for affinity grouped plan item. Only plan items with the Cancel action and the same value in this field are grouped together

In the case where plan items with different actions are grouped together due to affinity, the following logic is used to determine what action to apply to the plan item. The following rules apply:

1. If AffinityActionValue is specified, then the action of the plan item is the result of evaluating this xpath.
2. If AffinityActionValue is blank, or evaluates to an Empty String, then the remaining rules apply:
 - a. If all order lines have the same action, then the plan item action is the same as the order lines.
 - b. If order lines have different actions, then:

- a. If at least one order line has PROVIDE action, then the plan item has PROVIDE action.
- b. Otherwise if at least one order line has CEASE action, then the plan item has CEASE action.
- c. Otherwise, the plan item has UPDATE action.

For details, see *Fulfillment Catalog Product Catalog Guide*.

1) If XPath is defined against plan data, the format must be <Actual XPath> containing string \$var/PlanItem. For example, if you want to define the XPath for user-defined field name-value pair MSISDN=123, the XPath can be \$var/PlanItem[productID='GSMLine']/udfs[name='MSISDN']/value/text().



XPath evaluates data from the planItem. Refer to the *sample planItem xml*.

2) If XPath is defined against the order data, the format must be <Actual XPath> containing string \$var/Order. Refer to *Sample order XML*.

3) Default order data is considered for evaluation if XPATH does not contain \$var/PlanItem.



Refer to *Sample XPATHs* for XPATH definitions.

When Automated Order Plan Development returns a plan it indicates the action of the plan item. Under normal circumstances, this maps directly to the action of the associated order line that caused the creation of the plan item. In the case where plan items with different actions are grouped together, the following logic is applicable to determine what action to apply to the plan item.

1. If AffinityActionValue is specified, then the action of the plan item is the result of evaluating this xpath.
 - If AffinityActionValue is blank, or evaluates to an Empty String, then the remaining rules apply
2. If all order lines have the same action, then the plan item action is the same as the order lines.
3. If order lines have different actions, then:
 - If at least one order line has PROVIDE action, then the plan item has PROVIDE action.
 - Otherwise if at least one order line has CEASE action, then the plan item has CEASE action.
 - Otherwise, the plan item has UPDATE action.

Conditional Affinity Sample

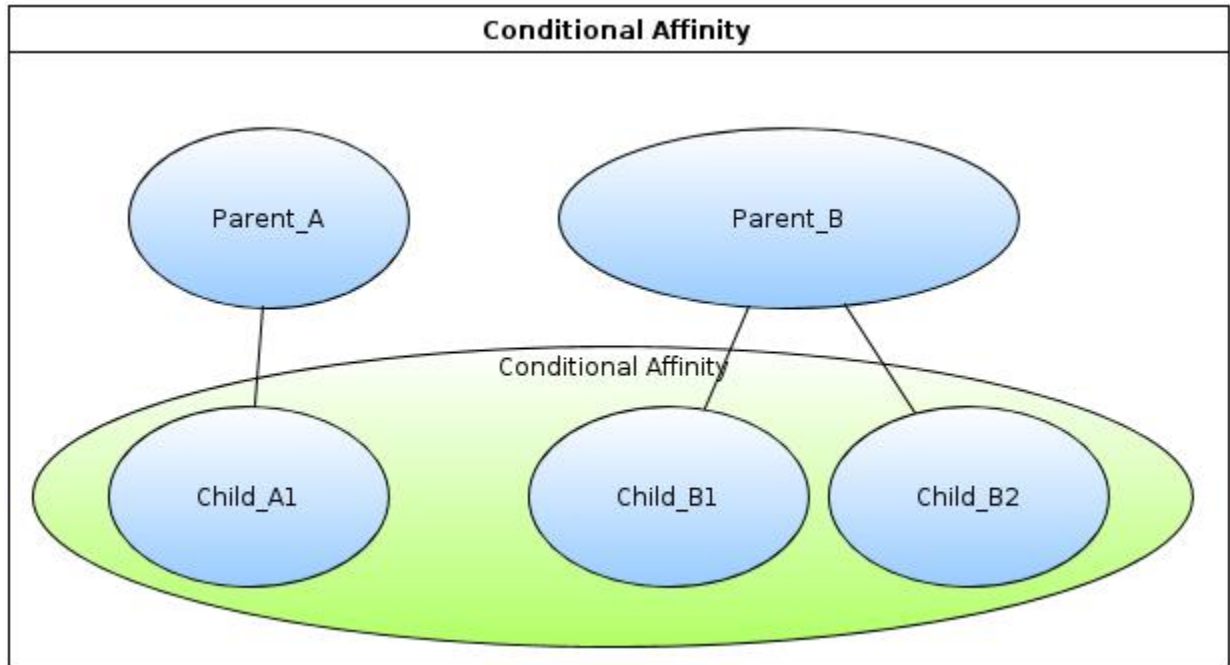
A product model is having two parents:

- Parent_A
- Parent_B

Consider a product model where conditional affinity is defined at all the child products:

- Child_A1
- Child_B1
- Child_B2

Conditional Affinity



The XPATH defined in the product model is evaluated against the submitted order schema.

The following table describes the attribute-based conditional affinity scenarios:

Attribute	Sample XPATH Expressions	Order Payload
AffinityCondition	<code>exists(\$var/Order/OrderHeaderUDF[name=UDFNAME and value="UDFVALUE"])</code>	<pre><ord1:udf> <ord1:name>UDFNAME</ord1:name> <ord1:value>UDFVALUE</ord1:value> </ord1:udf></pre>
AffinityCorrelation	<code>\$var/Order/orderlines[productID='Child_A1']/OrderlinesUDF[name='UDFNAME']/value/text()</code>	<pre><ord1:line> <ord1:lineNumber>1</ord1:lineNumber> <ord1:productID>Child_A1</ord1:productID> <ord1:quantity>1</ord1:quantity> <ord1:uom>1</ord1:uom> <ord1:action>PROVIDE</ord1:action> <ord1:actionMode>New</ord1:actionMode> <ord1:udf> <ord1:name>UDFNAME</ord1:name> <ord1:value>UDFVALUE</ord1:value> </ord1:udf> </ord1:line></pre>

Attribute	Sample XPATH Expressions	Order Payload
AffinityParentGroup	Child_B1 and Child_B2 have immediate parent. The two is affinity grouped when: AffinityParentGroup=true	
AffinityAction Value The affinityAction Group must be true	\$var/Order/orderlines[productID='Child_A1']/OrderlinesUDF[name='UDFNAME']/value/text()	<pre> <ord1:line> <ord1:lineNumber>1</ord1:lineNumber> <ord1:productID>Child_A1</ord1:productID> <ord1:quantity>1</ord1:quantity> <ord1:uom>1</ord1:uom> <ord1:action>PROVIDE</ord1:action> <ord1:actionMode>New</ord1:actionMode> <ord1:udf> <ord1:name>UDFNAME</ord1:name> <ord1:value>UDFVALUE</ord1:value> </ord1:udf> </ord1:line> </pre>

Configurable Handling of CrossLink + ProductComprisedOf Conflicts and Single Use + ProductComprisedOf Conflicts

Affinity can violate the product model ProductComprisedOf action-based sequencing when the provisioning of two or more products must be grouped together through a single affinity plan item for execution but have different parents, which provisioning must be sequenced in a specific order. The affinity plan item is executed for all parents irrespectively of the ProductComprisedOf action-based sequencing, which breaks product model and can lead to circular dependencies and missing dependencies.

System config parameters must be added to trigger the following behavior:

- **Error:** If Affinity and ProductComprisedOf conflict, stop and report an error.
- **Ignore:** If Affinity and ProductComprisedOf conflict, ignore Affinity rule and apply only ProductComprisedOf.
- **Apply:** If Affinity and ProductComprisedOf conflict, process with both rules applied but add dependencies.



This applies to all other Affinity types.

SingleUseHandling: Error | Ignore | Apply

- **Error:** If single use and ProductComprisedOf conflict, stop and report an error.
- **Ignore:** If single use and ProductComprisedOf conflict, ignore SingleUse rule and apply only ProductComprisedOf.
- **Apply:** If single use and ProductComprisedOf conflict, process with both rules applied but add dependencies.

The default is *Apply*.

Sort Plan

The sort plan functionality was implemented to sort the plan according to the subscriber for batch orders with multiple subscribers contained within the plan.

The sort plan function is defined to sort the plan according to an attribute defined within the order. This function makes sure that products that belong to similar grouping attributes follow each other in the GUI.

In scenarios where bulk orders for multiple subscribers are submitted into TIBCO Order Management - Long Running, the subscriber ID is used as the grouping mechanism. All the order lines have a user-defined field defined with the name SubscriberID. Once the entire plan has been generated all the plan items are sorted according to the value for the subscriber ID user-defined field. This ensures that all products for an individual subscriber follow one after the other. This is followed by the next subscriber and so on.

Attribute-Based Decomposition

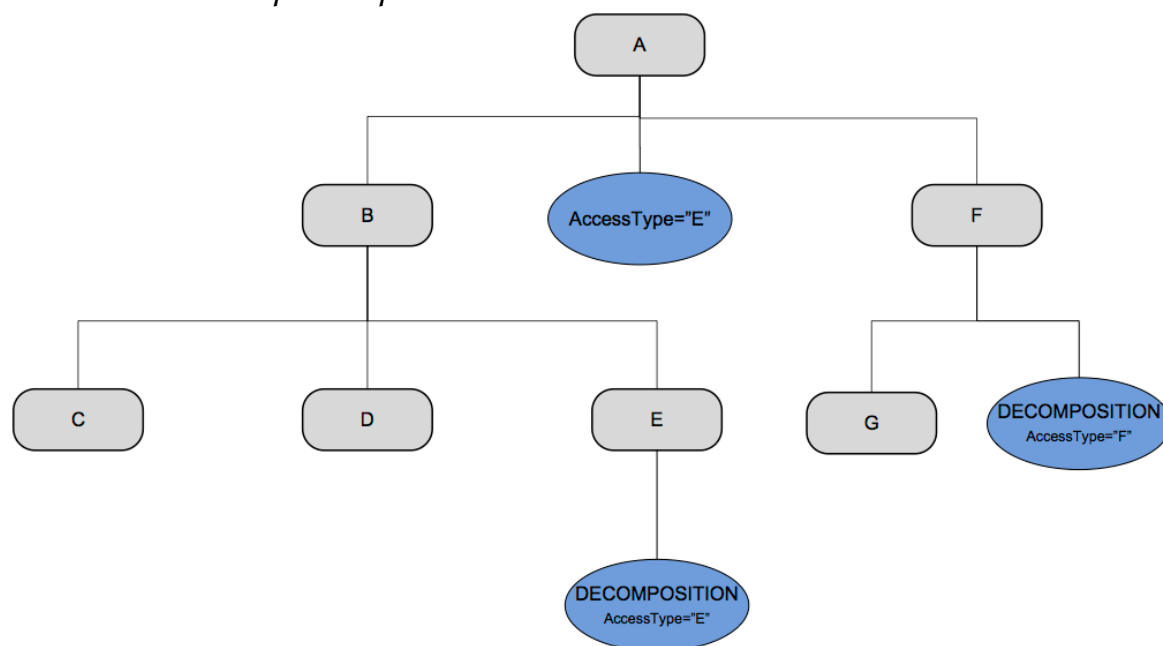
This functionality provides the ability to define decomposition rules along the relationship path for products. The decomposition rule is defined as XPATH logic, which grants the ability to apply the defined logic anyway along with the order. E.g. We can define that Copper Access or Fiber Access process component must only be in the plan if the Access Type in the order is Copper or Fiber.

For attribute-based decomposition can be applied the following conditions have to be satisfied:

1. DECOMPOSITION attribute needs exist in the product where the XPATH logic can be defined.
2. The XPATH logic must contain an exists (this is due to the XPATH logic is evaluated to either true or false). The logic can either be for a check of the User Defined Fields or an existence of a particular product within the order.

A simple scenario for Attribute based decomposition is described below:

Attribute based decomposition product



A bundled offering ("A" in the above figure) contains a characteristic for the type of access that the subscriber can specify during order entry. In our figure above this is the fulfillment of Product "E" or Product "F". The bundled offering has an associated characteristic named "AccessType" where the value can either be "E" or "F". For the fulfillment of access type E or F a unique execution plan is generated. The "DECOMPOSITION" characteristic for "F" and "E" contain a relationship value with "AccessType" set to either "F" or "E".

The Decomposition characteristic can contain complex XPATH logic, which can be used to determine, which branch of the tree must be included in the final execution plan for the offering.

Our design takes into account the new product catalog characteristic called "DECOMPOSITION". The decomposition engine processes this characteristic and determine, which branch in the product hierarchy is

required for the final execution plan. If in our order access type “E” is specified then branch “F” is not included in the execution plan and “E” is included. If access type “F” is specified then “E” is not included in the execution plan.

XPath for attribute-based decomposition can be used against the User Defined Fields. The User Defined Fields can come from order line or from product model characteristics. Automated Order Plan Development configuration flag "includeproductmodelcharacteristics" is used to include product model characteristics for xpath execution. By default, product model characteristics are not considered.

By default, all the order lines from the order are considered to check against whom the Xpath must be executed. Conditionally it can also be made to execute only against the order line from which product is being decomposed. Automated Order Plan Development configuration "includeonlyproductorderline" needs to be set to true in this case. By default, it is false.

The xpath expression is relative to the element "Product". A sample xpath is `exists($var//Product[udf[name='AccessType' and value='copper']])`

ProductDependsOn and ProductRequiredFor Relationships

ProductDependsOn relationship: ProductDependsOn (PDO) is a product dependency relationship to sequence the associated target and source plan items. The flexible product decomposition occurs through ProductDependsOn relationship. This establishes a relationship between two products and is evaluated during the decomposition process.

ProductRequiredFor relationship: The ProductRequiredFor (PRF) relationship is a prerequisite relationship for a product to add a target plan item.

The ProductDependsOn and ProductRequiredFor relationships enable you to create product offers without defining sequencing for the products. You can create ProductDependsOn relationship to lower level products instead of using ProductComprisedOf links.

The ProductDependsOn functionality provides a base behavior that permits to sequence plan items corresponding to products related by ProductDependsOn when:

1. ProductDependsOn source and ProductDependsOn target product instances have no LINKID defined.
2. ProductDependsOn source and ProductDependsOn target product instances have LINKID defined and have the same LINKID value.

The feature can extend the base behavior and sequence additionally plan items corresponding to products related by ProductDependsOn when:

1. ProductDependsOn source product instance has LINKID defined and ProductDependsOn target product instances have no LINKID defined.
2. ProductDependsOn source product instance has no LINKID and target product instances have a LINKID defined.

A ProductDependsOn source product instance can relate to multiple ProductDependsOn target product instances by using base and extended cases so that following use cases are possible:

- A ProductDependsOn source product that has a LINKID defined and is related to a ProductDependsOn target instances that has same LINKID defined shall be also related to ProductDependsOn target product instances that have no LINKID defined.
- A ProductDependsOn source product that has no LINKID defined and is related to a ProductDependsOn target instance that has no LINKID defined shall be also related to ProductDependsOn target product instances that have a LINKID defined. By default, for ProductDependsOn sequencing, the base behavior is enabled. To enable the extended behavior set the "EnableBiDirectionalLinkID" to true, by default it is false.

By default, only one instance of required product per LinkID is available in plan. If there is a requirement to override this behavior to include multiple instances of required product with the same LinkID, then the

Automated Order Plan Development flag "AllowMultipleRequiredProducts" must be set to true. By default, it is false.



The ProductRequiredFor adds the required plan item without dependency, if you create only ProductRequiredFor.

The ProductDependsOn and ProductRequiredFor relationships have the following two relationship attributes:

- Source Action
- Target Action

The ProductRequiredFor relationship also has the third relationship attribute named ocvValidationReq. This is a Boolean flag for validation. Based on validation flag, Automated Order Plan Development adds product configured with the ProductRequiredFor relationship in the plan.

The ProductDependsOn relationship also has the third relationship attribute named 'sequenceDirection'. The valid values of this attribute are either 'AFTER' or 'BEFORE'. This attribute is paired with the provided values of SourceAction and TargetAction. For each SourceAction and TargetAction, there is a value defined for the sequenceDirection attribute.

- A 'BEFORE' sequence direction creates a dependency of the target product on the source product.
- An 'AFTER' sequencing direction creates a dependency of the source product on the target product. This is the default.

If no value is provided in the sequenceDirection attribute, the attribute defaults to 'AFTER,' and the functionality works as it did before the introduction of sequenceDirection relationship attribute. The backward compatibility is occurred as a result.

The value defined in the sequenceDirection attribute creates a dependency of the target product on the source product or it creates a dependency of the source product on the target product.

If a ProductDependsOn Source Product has a dependency on child products, then those child products have a dependency on the ProductDependsOn target product by default. If there is a requirement to override this default behavior and set the dependency of the source product directly on the target product, then the value of the flag "IgnorePDOFirstChildDependency" needs to be set to true in the Automated Order Plan Development configuration file. By default, this value is false.



ProductDependsOn relationship can be made conditional by using XPATH statement stored in optional product characteristic DECOMPOSITION_DEPENDS_ON.

ProductRequiredFor relationship can be made conditional by using XPATH statement stored in optional product characteristic DECOMPOSITION_REQUIRED_FOR.

Source and Target Attribute Values

The following table describes the different possible combinations:

SourceAction	TargetAction
PROVIDE	PROVIDE
PROVIDE	UPDATE
PROVIDE	CEASE
PROVIDE	CANCEL
UPDATE	PROVIDE

SourceAction	TargetAction
UPDATE	UPDATE
UPDATE	CEASE
UPDATE	CANCEL
CEASE	PROVIDE
CEASE	UPDATE
CEASE	CEASE
CEASE	CANCEL
CANCEL	PROVIDE
CANCEL	UPDATE
CANCEL	CEASE
CANCEL	CANCEL

You can also define source action and target action to match the following combination by using uppercase, comma-separated values. For example:

SourceAction: PROVIDE,PROVIDE,UPDATE,CEASE,CANCEL,CEASE

TargetAction: UPDATE,CANCEL,PROVIDE,UPDATE,PROVIDE,UPDATE

You can also define sequenceDirection to match the following combination by using uppercase, comma-separated values. For example:

SourceAction: PROVIDE,PROVIDE,UPDATE,CEASE,CANCEL,CEASE

TargetAction: UPDATE,CANCEL,PROVIDE,UPDATE,PROVIDE,UPDATE

SequenceDirection: AFTER,BEFORE,AFTER,BEFORE,BEFORE,AFTER



There cannot be any space between the commas and the values.

Dependency between planitems occurs when both the following occur:

- The sequenceDirection attribute has valid values, i.e. either 'AFTER' or 'BEFORE.'
- The number of sequenceDirection attributes matches with the number of Source Actions and the number of Target Actions.



There is only one target action for any given source action.

The following table explains the ProductDependsOn and ProductRequiredFor relationships and their impact on orders and plans.

Product Configuration	Order	Plan
Product A has a ProductRequiredFor relationship with Product B having source action and target action PROVIDE and PROVIDE	OL1=ProductA	Two plan item (A and B) do not depend on each other
Product A has ProductRequiredFor and ProductDependsOn relationship with B and ProductRequiredFor and ProductDependsOn has source action and target action PROVIDE and PROVIDE	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide)	planItemA depends on planItemB
Product A has ProductRequiredFor and ProductDependsOn relationship with B and ProductRequiredFor and ProductDependsOn has source action and target action PROVIDE and PROVIDE	OL1=ProductA	planItemA depends on planItemB
Product A has ProductDependsOn relationship with B having source action and target action PROVIDE and PROVIDE	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide)	planItemA depends on planItemB

The following table explains ProductDependsOn with Sequence direction and their impact on orders and plans.

Product Configuration	Order	Plan
Product A has ProductDependsOn relationship with B having SA & TA as PROVIDE & PROVIDE. SequenceDirection is AFTER.	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide)	Two planitem having planItemA depends on planItemB
Product A has ProductDependsOn relationship with B having SA & TA as PROVIDE & PROVIDE. SequenceDirection is BEFORE.	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide)	Two planitem having planItemB depends on planItemA
Product A has ProductDependsOn relationship with B having SA & TA as PROVIDE & PROVIDE. SequenceDirection is AFTER. Product B has ProductDependsOn relationship with C having SA & TA as PROVIDE & PROVIDE and SequenceDirection is BEFORE.	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide) OL3=ProductC(Action=Provide)	Three planitems having planItemA depends on planItemB and planItemC depends on planItemB

Product Configuration	Order	Plan
Product A has ProductDependsOn relationship with B having SA & TA as PROVIDE & PROVIDE. SequenceDirection is BEFORE. Product B has ProductDependsOn relationship with C having SA & TA as PROVIDE & PROVIDE and SequenceDirection is AFTER.	OL1=ProductA(Action=Provide) OL2=ProductB(Action=Provide) OL3=ProductC(Action=Provide)	Three planitems having planItemB depends on planItemA and planItemB depends on planItemC

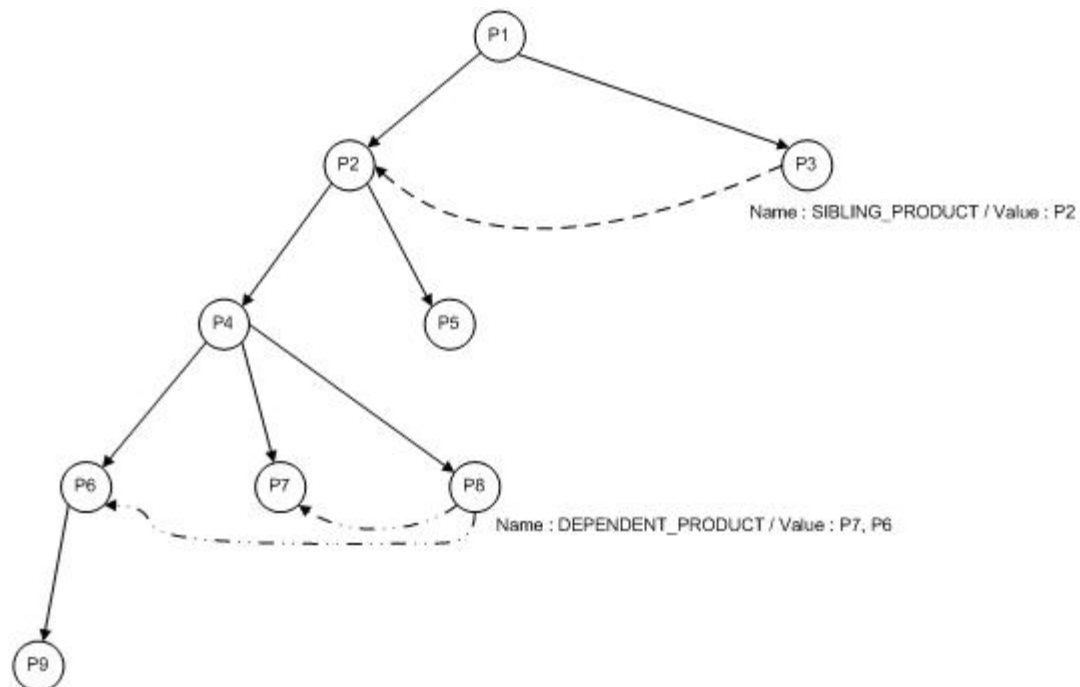
Dependent and Sibling Products

TIBCO Order Management provides the ability in the product catalog to indicate that a product is dependent on its peer [DEPENDENT_PRODUCT] or peer's hierarchy [SIBLING_PRODUCT].

The only difference between dependent products and sibling products is that the dependent product would not add the peer product's children to be included in the subsequent southbound service calls when sibling product adds the peer's children on the southbound service calls.

The diagram shows the product model.

Data Model



P8 has a dependent product link to P7 and P6. This means that the process component corresponding to P8 can use the output User Defined Fields of P7 and P6 during the execution provided P7 and P6 have been ordered directly or indirectly in the order and corresponding process components have been executed.

P3 has a sibling product link to P2. This means that the process component corresponding to P3 can use the output User Defined Fields of P2, P4, P5, P6, P7, P8 and P9 during the execution provided P2 has been ordered directly or indirectly in the order and corresponding process components have been executed.

The 6 product characteristics as explained in the table below must be added in the product model defined in TIBCO Fulfillment Catalog or offline model xml. The dependent or sibling link can be defined for a product by creating the **Characteristic** relationship with one of the above relevant products [as per the

scenario] with the value of **RelationshipValue** attribute as the comma-separated IDs of the dependent or sibling products.

Name	Description
DEPENDENT_PRODUCT	Dependent product characteristic for PROVIDE scenario
DEPENDENT_PRODUCT_CEA SE	Dependent product characteristic for CEASE scenario
DEPENDENT_PRODUCT_UPD ATE	Dependent product characteristic for UPDATE scenario
SIBLING_PRODUCT	Sibling product characteristic for PROVIDE scenario
SIBLING_PRODUCT_CEA SE	Sibling product characteristic for CEASE scenario
SIBLING_PRODUCT_UPDATE	Sibling product characteristic for UPDATE scenario

For example, Dependent link for P8 in case of PROVIDE scenario can be specified by creating a Characteristic relationship between P8 and DEPENDENT_PRODUCT with the value of RelationshipValue as "P6, P7".

Shared Attributes

This section describes the Shared Attributes and its samples test scenarios.

Shared Attributes are used when two Products (parent to child and sibling) share the same attribute and its corresponding value and an update in the value of one needs to be reflected in the other. If an attribute is deemed as a shared attribute and when the value was passed on the order then during decomposition value is copied to the other products based on the EvaluationPriority set on the other products.

EvaluationPriority

Multiple products can have the same shared attribute. Hence, if value for a shared attribute needs to be merged with same shared attribute in other products, user needs to define the EvaluationPriority, which indicates the priority of merging the specified characteristic from the target Product.

During plan development process, Automated Order Plan Development checks if characteristic (i.e. attribute) is of type 'Shared'; if yes then it checks the EvaluationPriority for the characteristic. If products mentioned on the EvaluationPriority have the same shared attribute, then the value for the characteristic is taken from the product. If none of the products mentioned on the EvaluationPriority have the same shared attribute OR EvaluationPriority is not defined, then the value is taken from order line (if passed in order line) or product model.

Second part of the Shared attribute definition mentions that update in the value of shared attribute in one product needs to be reflected in other products having same shared attribute.

During execution, the process component might have to update the attribute (user-defined field) values. To update the value of a user-defined field, the process component calls setPlanItem on Transient Data Store mentioning the User Defined Fields to be updated. Process component sends the following details for the user-defined field:

1. **name** - name of the user-defined field to update
2. **value** - updated value
3. **flavor** - 'output' (this indicates that process component has updated the value from set/get methods), Input (this indicates that process component has updated the value from order line), Config (this indicates that process component has updated the value from Product model)

4. **type** - Shared (if user-defined field is of type Shared)

On the subsequent calls to getPlanItems on Transient Data Store (process component might make this call to get details such as User Defined Fields for plan items from Transient Data Store), Transient Data Store checks if requested plan items have any user-defined field (i.e. attributes) with type as 'Shared'. If Shared User Defined Fields are present then Transient Data Store checks the EvaluationPriority for that user-defined field.

For the products mentioned on the EvaluationPriority, for each product (in the order of priority) Transient Data Store checks if it contains the user-defined field with same name and flavor = output. If Transient Data Store finds such user-defined field then value from this user-defined field is returned. If EvaluationPriority is not defined OR products mentioned on the EvaluationPriority do not contain the user-defined field with same name and "output" flavor then value from order line/product model is returned (i.e. merging is not done).

Below are the sample of EvaluationPriority:

- For single product, product ID is followed by priority with colon in between them.

```
<productId>:<priority>
```

Example:

```
<ns0:evaluationPriority>SIM_TECNICO_BP1:1</ns0:evaluationPriority>
```

- For multiple products, sets of product-priority are separated by comma.

```
<productId>:<priority>,<productId>:<priority>
```

Example:

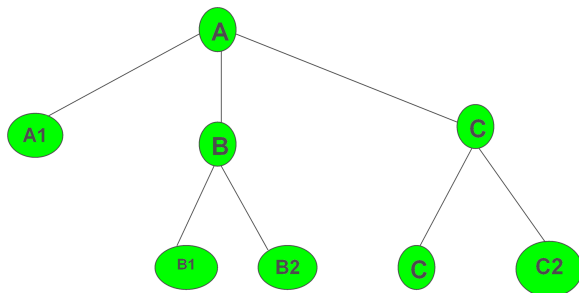
```
<ns0:evaluationPriority>SIM_TECNICO_BP1:1,SIM_TECNICO_BP2:2</ns0:evaluationPriority>
```

Shared Attributes - Sample Test Scenarios

This section describes the simple cases to test shared attributes in different scenarios.

1. Publish Product Model. The processes must be running in Test harness.

Here is an example for shared attributes with value of one reflecting in the other.



Scenario 1:

For the above product model structure, submit order for SharedAttribute_B and SharedAttribute_B1. Refer to the [Order Submission](#) topic for more information on the order submission process. Send new value for the shared attribute in the user-defined field format and values for both the attributes.

The value of B must reflect in B1, which conforms to the explanation of the Shared Attributes.

Scenario 2:

Submit order and send new value for the shared attribute (in the user-defined field format). Through order submission, send values for both the attributes, SharedAttribute_B and SharedAttribute_B1. Using SetPlanItemRequest service, set Shared Attributes value of B.

In this case also, the value of B must reflect in B1. Using the GetPlanItemrequest service for B1 returns a new value, which is reflected in B, thus corresponding value and an update in the value of one is reflected in the other.

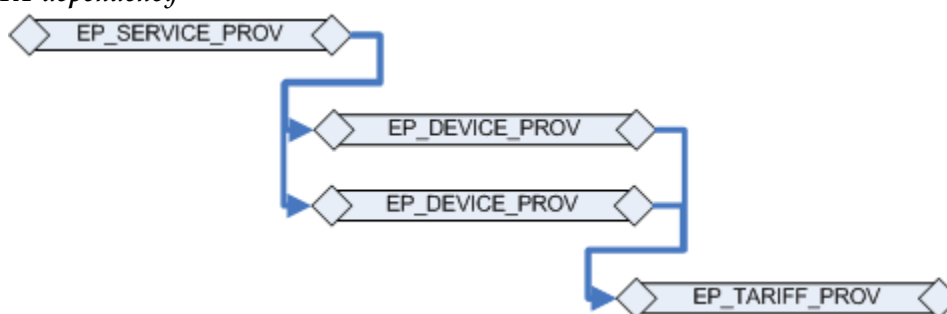
Intermediate Milestones Dependencies

The actual fulfillment of a product is done by orchestrating the back-end process components. By default, any process component has two milestones:

1. START
2. END

These milestones represent the starting and the end parts of it. There is a direct dependency between the process components due to sequencing of the products in the catalog. This dependency is of type END-to-START, or once a process component is completely executed, then only the dependent process component can start its execution as shown in the following figure:

END-to-START dependency



The process component EP_DEVICE_PROV can start only when EP_SERVICE_PROV is completed and EP_TARIFF_PROV can start only when EP_DEVICE_PROV is completed.

TIBCO Order Management also supports the following complex types of dependencies between the executing process components:

- [Milestone to START Dependency](#)
- [END to Milestone Dependency](#)
- [Milestone to Milestone Dependency](#)
- [Milestone without Dependency](#)
- [Conditional Milestones Dependency](#)

These dependencies are supported with the implementation of Intermediate Milestones within the process component in addition to the START and END.

The functionality provides a base behavior that permits plan items to be sequenced corresponding to products related by MDO when:

1. MDO related product instances have no LINKID defined.
2. MDO related product instances have LINKID defined and have same LINKID value.

This feature can extend the base behavior and sequence additionally plan items corresponding to products related by MDO when:

1. MDO related parent product instance has LINKID defined and child product instances have no LINKID defined.
2. MDO related parent product instance has no LINKID and child product instances have a LINKID defined.

An MDO related parent product instance can relate to multiple child product instances using base and extended cases so that following use cases are possible:

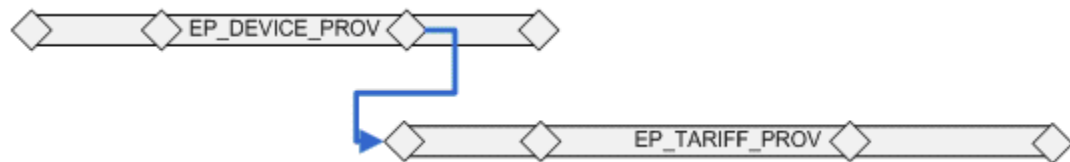
- An MDO related parent product that has a LINKID defined and is related to a child instances that has same LINKID defined shall be also related to MDO related child product instances that have no LINKID defined.
- An MDO related parent product that has no LINKID defined and is related to a child product instance that has no LINKID defined shall be also related to MDO related child product instances that have a LINKID defined.

Milestone to START Dependency

START milestones of a process components has a dependency on the completion of an intermediate milestone(s) in another process component(s).

The following figure shows the Milestone to START dependency:

Milestone to START Dependency

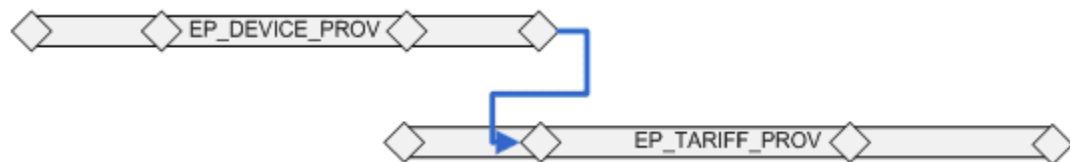


END to Milestone Dependency

Intermediate milestone(s) in a process component has a dependency on the completion of the END milestone(s) in another process component(s).

The following figure shows the END to Milestone dependency:

END to Milestone Dependency



Milestone to Milestone Dependency

Intermediate milestones in a process component has a dependency on the completion of the intermediate milestones in another process components.

The following figure shows the milestone to milestone dependency:

Milestone to Milestone Dependency

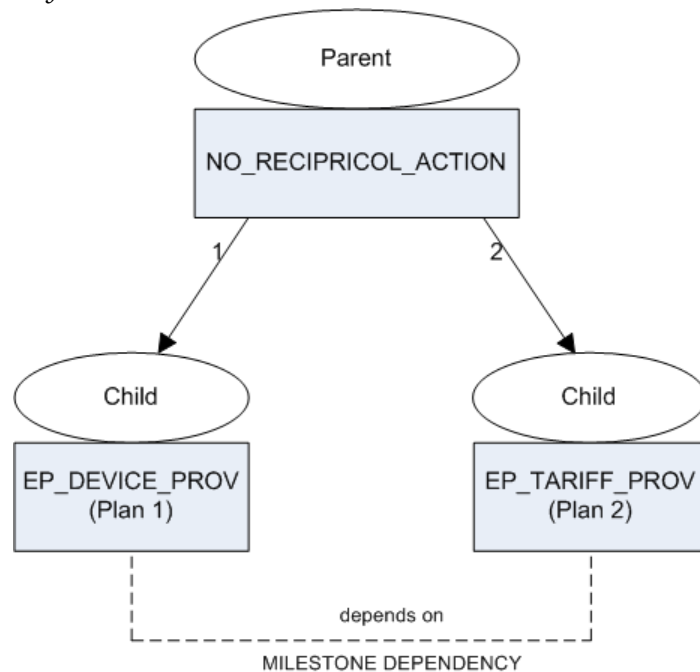


ProductComprisedOf Sequence and Intermediate Milestone Dependency

Sequencing is an indication of the order in which the plan items are executed. If there is not an intermediate milestones dependency set, a default START-END dependency is created. When using the ProductComprisedOf sequence and intermediate milestones at the same time, the intermediate milestone dependencies takes a precedence over the conventional End->Start dependencies.

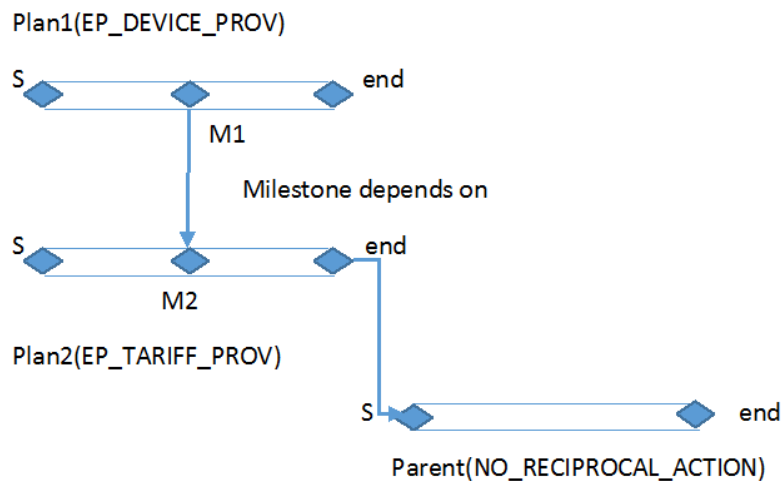
The following figure shows a START->END dependency with an intermediate milestone dependency:

Start Milestone Dependency



1. The plan for an order is generated in the usual manner. There are no intermediate milestone dependencies yet. The following is the plan based on the sequences:
Start_Plan1_End --> Start_Plan2_End --> Start_NO_RECIPROCAL_ACTION_End.
2. The generated plan is evaluated against the intermediate milestone dependencies defined in the product model so as to rearrange the dependencies. In this step as per the product model, the milestone 'Milestone_Dependency_2' of Plan2 depends on the milestone_M1 of Plan1.
Start_Plan1End ---> Milestone DependencyPlan2_End
3. There is a check for whether to add the existing dependencies or not. If the intermediate milestone dependency in a plan item is on the same plan fragment as the one in existing dependency, it ignores the existing dependency on that plan fragment, for example, the start of Plan2 does not depend on the end of Plan1 anymore.

The following is the final plan:



Start_Plan1_End ---> Start_Milestone DependencyPlan2_End --->
StartNO_RECIPROCAL_ACTION_End

The intermediate milestone dependencies take a precedence over the conventional End-->Start of dependencies.

Milestone without Dependency

There can be intermediate milestones defined in a process component, which does not have any dependencies. However, milestones in another process component might depend on any of these milestones.

The following figure shows the Milestone without any dependency:

Milestone without Dependency



These types of dependencies help manage the complex order fulfillment process. For instance, start activating the broadband service once the broadband device fulfillment reaches a certain status, say, INSTALLED.

The product model schema has been updated to support the milestones and dependency definitions. See *TIBCO® Fulfillment Catalog Product Catalog* Guide for newly added repository and relationships to support the intermediate milestones dependencies.

Conditional Milestones Dependency

The dependency between intermediate milestones can be conditional. This is specified using the relationship attribute called Condition in TIBCO Fulfillment Catalog. It is represented in the product model as illustrated below:

Conditional Milestones Dependency

```
<ns0:plan>
  <ns0:name>EP_TEST_PROVIDE_11</ns0:name>
  <ns0:description>Product 11 PROVIDE</ns0:description>
  <ns0:action>PROVIDE</ns0:action>
  <ns0:affinity>false</ns0:affinity>
  <ns0:milestone>
    <ns0:name>START</ns0:name>
    <ns0:dependency>
      <ns0:planName>EP_TEST_PROVIDE_10</ns0:planName>
      <ns0:milestoneName>MILE1</ns0:milestoneName>
      <ns0:type>InLink</ns0:type>
      <ns0:condition>Match:Database=Middleware</ns0:condition>
    </ns0:dependency>
  </ns0:milestone>
</ns0:plan>
```

In this sample, the START milestone of EP_PROVIDE_11 is dependent on MILE1 of EP_PROVIDE_10, only if the specified condition is satisfied.

The condition syntax can be one of following three types:

- Parent user-defined field Syntax
- Child user-defined field Syntax
- Match Parent-Child user-defined field Syntax



There is no provision to specify the XSLT statement in the condition as it is there for Attribute Based Decomposition.

Note the following definitions:

- **Parent:** The plan fragment whose milestone has a dependency on another plan fragment. The parent user-defined field is referred to a user-defined field passed in the order line in the order for that product, which is propagated into the plan item for that order line.
- **Child:** The plan fragment on whose milestone a milestone in another plan fragment depends. The child user-defined field is referred to a user-defined field passed in the order line in the order for that product, which is propagated into the plan item for that order line.

In the plan illustrated above, EP_TEST_PROVIDE_11 [START] is dependent on EP_TEST_PROVIDE_10 [MILE1]. It is assigned to PROD11 as PROVIDE plan fragment. PROD11 is the parent and PROD10 is child.

Parent user-defined field Syntax

Value:Parent(ParentUDFName=ExpectedValue)

The condition is satisfied only if there is a user-defined field in the parent plan item with the same value as passed in the condition as "ExpectedValue".

Child user-defined field Syntax

Value:Child(ChildUDFName=ExpectedValue)

The condition is satisfied only if there is a user-defined field in the child plan item with the same value as passed in the condition as "ExpectedValue".

Match Parent-Child user-defined field Syntax

Match:ParentUDFName=ChildUDFName

The condition is satisfied only if the value of the user-defined field [ParentUDFName] in parent plan item is equal to the value of the user-defined field [ChildUDFName] in child plan item.

Order Amendment

Order Amendment allows the order, which is currently being fulfilled, to be modified for different parameters such as action, requiredByDate, and User Defined Fields in the existing order lines. It also allows adding a new order line in the request. The parameters and their reason for change are mentioned as follows:

- The parameter values passed in the original request was incorrect. The corrected values can be passed by sending an amendment request.
- The parameter values require a modification as per the change request from the end user. For example, the bandwidth of a product named Internet Bundle is passed as a user-defined field named Bandwidth in the order line. The bandwidth in the original order was 1 Mbps. When the order was being fulfilled, the customer requested the bandwidth to be updated to 2 Mbps. This is done by sending an amendment request by changing the value of the user-defined field named Bandwidth to 2 Mbps.
- An additional product required fulfillment when the current one is being fulfilled.

An order can be amended when it is in one of the following states:

START	If the order is amended in these states then it is called a <i>pre-plan development</i> amendment, because the execution plan has not yet been created. In this scenario, the execution plan generated for the amendment request is considered and executed.
SUBMITTED	
FEASIBILITY	
OPD	
PREQUALIFICATIONFAILED	
EXECUTION	If the order is amended in these states it is <i>post-plan development</i> amendment, because the execution

SUSPENDED	plan was already created and had begun execution. In this scenario, the existing plan requires modification and merging with the plan that has been generated as per the amendment request. <i>Post-plan development</i> amendment is the most frequently used amendment.
ERROR_HANDLER	

An order cannot be amended when it is in either of the FINAL states:

- COMPLETE
- CANCELLED
- WITHDRAWN

Amendment Workflow

Since an order amendment involves the modification of the current execution plan, a predefined process is adopted. The predefined process is as follows:

1. Upon accepting an order amendment request, the Orchestrator first tries to suspend the current execution plan by sending the suspend requests (`PlanItemSuspendRequest` message) to all the plan items that are in EXECUTION state. Based on the implementation of the process components, and the point at which the process component is executed, the process components might send a successful suspend response (`PlanItemSuspendResponse` message) or a successful completion response (`PlanItemExecuteResponse`). Any one of the responses is acceptable by the Orchestrator.
2. Once the execution plan (and order) reaches the SUSPENDED state, the Orchestrator sends a plan generation request to Automated Order Plan Development to generate the execution plan as per the order lines in the amendment request.
3. The new execution plan generated by the core Automated Order Plan Development is merged with the existing plan to add, or to modify the plan items as per the changes in the amendment request.
4. Based upon the modification rule characteristics defined in the product model, the compensatory plan items are added in the new execution plan to let the undoing of the tasks that were performed by the earlier corresponding plan items. If required, the REDO plan items are also added in the new execution plan to let the redoing of the tasks that needs to be performed by a particular plan item.
5. Upon receiving the consolidated execution plan for the amendment request from Automated Order Plan Development, the Orchestrator activates the SUSPENDED plan and starts orchestrating it as per the latest dependencies.
6. All SUSPENDED plan items is activated, either for cancellation (`cancelWithNoRollback` or `cancelAndRollback`) or resume execution (`resumeExecution`) by sending the `PlanItemActivateRequest` messages.
7. Any compensatory and redo plan items, created during the amendment process, is executed in the same way as the regular plan items by sending the `PlanItemExecuteRequest` messages, so as to either complete or cancel the order.

Modeling of the Required Characteristics in Fulfillment Catalog

As per the requirement of the amendment use case, some or all of the following characteristics are required to be available in the product model published to TIBCO Order Management. The modeling of these characteristics and relating them with the required products, needs to be done in TIBCO Fulfillment Catalog at the design time. Refer the generic steps given below for modeling.



This section covers just the high level modeling steps specific to the characteristics required for amendments. Refer TIBCO Fulfillment Catalog documentation for details.

1. Create the following records in CHARACTERISTIC repository:
 - EPMR_ACTION_PROVIDE
 - EPMR_ACTION_Cease
 - EPMR_ACTION_UPDATE
 - EPMR_ACTION_WITHDRAW
 - COMPENSATE_PROVIDE
 - COMPENSATE_Cease
 - COMPENSATE_UPDATE
2. For more granular EPMR actions based on the plan item statuses, the user can add additional characteristics mentioned below in generic format. Note that these characteristics are used only in case of the new and improved user-defined field modification functionality. Refer the New Characteristics sub-section in OrderLine user-defined field change.
 - EPMR_ACTION_<<action>>_UDF_CHANGE_<<Plan Item Status>>. For example, EPMR_ACTION_PROVIDE_UDF_CHANGE_SUSPENDED.
3. Create the Characteristic relationship between the records in PRODUCT repository and one or more EPMR_ACTION_* records in the CHARACTERISTIC repository mentioned in point 1 and 2 above. The logically valid value for the RelationshipValue attributes in all these Characteristic relationships can be one of the four EPMR actions - COMPENSATE, RESTART, COMPENSATE_RESTART, or IGNORE. Refer the [Execution Plan Modification Rules \(EPMR\)](#) topic for the significance of these four actions. For example, the technical product Router can have a Characteristic relationship with EPMR_ACTION_PROVIDE characteristic, with the value of RelationshipValue attribute as COMPENSATE_RESTART.
4. Create the Characteristic relationship between the records in PRODUCT repository and one or more COMPENSATE_* records in the CHARACTERISTIC repository mentioned in point 1 above. The logically valid value for the RelationshipValue attributes in all these Characteristic relationships must be the ID of the plan fragment record that is required to be executed as the compensation task for corresponding action. For example, the technical product Router can have a Characteristic relationship with COMPENSATE_PROVIDE characteristic, with the value of RelationshipValue attribute as the planFragmentID Router_Cancel.

Types of Amendment

At a high level, order amendments can be classified into the two heads and further into each sub-type as described below:

1. Changes at order line level
 - a. Action Change - Changing the action in one or more order lines.
 - b. RequiredByDate Change - Changing the requiredByDate in one or more order lines.
 - c. user-defined field Change - Changing the user-defined field(s) in one or more order lines.
2. Changes at the order header level
 - a. RequiredByDate Change - Changing the requiredByDate at order header level.
 - b. OrderLine Addition - Adding a new order line in the order.
 - c. OrderPriority Change – Changing the orderPriority at the order header level.

The amendment types at the order line level are MUTUALLY EXCLUSIVE. Only one of the amendment types is allowed for a particular order line in an amendment request. E.g. the action and User Defined Fields cannot be simultaneously updated in an order line. If multiple amendment types are tried simultaneously for an order line in a single amendment request, an exception with the appropriate code and an appropriate message is thrown. For example, if action and requireByDate both are changed in an

order line in the amendment request, an exception with error code "TIBCO-AFF-OMS-100064" and error message "Action and requiredByDate cannot be modified simultaneously in an order line", is thrown.

However, different amendment types can be applied in different order lines as part of the same amendment request. That is, action change in one order line and user-defined field or requiredByDate change in another one can be done simultaneously.

OrderLine Action Change

In this amendment type, the fulfillment action in an order line can be changed as part of the amendment request. E.g. action was PROVIDE in an order line in the original order request and changed to UPDATE in the amendment request. Since CANCEL can also be passed as the action in one or all the order lines in the amendment request, order line cancellation and entire order cancellation are the sub-types of this amendment type.

OrderLine Cancellation

To cancel a particular order line, CANCEL must be passed as action in the amendment request. The PENDING plan items associated to the order line are directly CANCELLED. Execution Plan Modification Rules are applied on the plan items that were COMPLETE or SUSPENDED before the amendment so as to compensate them, as per the Execution Plan Modification Rules action defined in the product model. Once all the associated plan items are either canceled or compensated, the order line is marked as canceled by changing its status to CANCELLED.

Entire Order Cancellation

To cancel the entire order, CANCEL must be passed as action in all the order lines in the amendment request. All the PENDING plan items in the execution plan are directly CANCELLED. Execution Plan Modification Rules are applied on the plan items that were COMPLETE and SUSPENDED before the amendment to compensate them, as per the Execution Plan Modification Rules action defined in the product model. Once all the plan items are either canceled or compensated, all the order lines and also the order is marked as canceled by changing the statuses to CANCELLED. The Execution Plan Modification Rules are applied in case of order line or entire order cancellation, based on the Boolean value (true/false) of the roll back user-defined field passed in the order header. The modification rules is applied if the roll back UDF's value is true, otherwise it is not applied.



The default behavior is always to roll back, which is, if the roll back user-defined field is not passed in the order header, it is considered to be true.

An order can also be canceled using the CancelOrderRequest SOAP service and from Order Management Server user interface.

Preconditions for Action change

Following are the preconditions for the order line action change amendment type.

1. The number of order lines in the amendment request must match with those in the original order request.
2. The lineID, productID, requiredByDate, and User Defined Fields in all the order lines in the amendment request must match with those in the original order request.

When the fulfillment action in an order line is changed, the plan items associated with that order line in the existing plan are handled in different ways.

1. The action in the amendment request is set as the fulfillment action in all PENDING plan items.
2. Execution Plan Modification Rules (EPMR) are applied to all SUSPENDED or COMPLETE plan items to take the appropriate actions on these plan items such as compensating the earlier tasks and/or redoing the tasks from beginning.

For any action change in the amendment request other than CANCEL, the Execution Plan Modification Rules characteristic corresponding to the action in the original plan item, from the product being fulfilled by that plan item, is considered when applying the modification rule.

OrderLine Action in Original Request	OrderLine Action in Amendment Request	Execution Plan Modification Rules Characteristic Considered
PROVIDE	Any, except for CANCEL and PROVIDE	EPMR_ACTION_PROVIDE
UPDATE	Any, except for CANCEL and UPDATE	EPMR_ACTION_UPDATE
CEASE	Any, except for CANCEL and CEASE	EPMR_ACTION_CEASE

On the other hand, if CANCEL is passed as the order line action in the amendment request, the EPMR_ACTION_WITHDRAW characteristic from the product being fulfilled by the corresponding plan item is considered always, regardless of the action in the original request.

Based on the value of the Execution Plan Modification Rules characteristic that was considered, the modification rules are applied on the required plan items. See topic [Execution Plan Modification Rules \(EPMR\)](#) to understand the effect of each action.



If the Execution Plan Modification Rules characteristic to be considered (Example: EPMR_ACTION_PROVIDE) is not present in the corresponding product model, COMPENSATE_RESTART is considered as the default EPMR action, only if the flag CompensateRestartForNoEPMRChar is set in Automated Order Plan Development configurations. See the topic [Amendment Configuration Flags](#) to understand the significance of each flag.

Once the EPMR action is applied on all the required plan items and the compensatory and/or redo plan items are generated, the dependencies in the parent plan items are updated appropriately. See topic [Impact on Dependencies](#) to understand how the dependencies are modified. The amendment plan is sent out to Orchestrator so as to fulfill the order sent in the amendment request.

RequiredByDate Change

RequiredByDate for an order defines the time at which the order plan must be executed. It can be mentioned at both the order header level or/and the order line level. In terms of dependency in the order plan, it generates a time dependency (with absolute time) for a plan item along with dependency on other executing plan items (point dependency) if any. Once the absolute time is reached, time dependency is considered as satisfied.

Following are the preconditions for the order line requiredByDate change amendment type:

1. The number of order lines in the amendment request must match with those in the original order request.
2. The lineID, productID, action, and User Defined Fields in all the order lines in the amendment request must match with those in the original order request.

Following is the process of calculating a time dependency with respect to requiredByDate.

- If requiredByDate is set on the order level only, the start time dependency applies to all plan items with no leading dependencies
- If requiredByDate is set on the order line level only, the start time dependency applies to plan items for that order line

- If requiredByDate is set on the order header level and on the order line level, the following behavior applies:
 - If requiredByDate in Order Header is later than requiredByDate in order line, then the start time used is the one at order level
 - If requiredByDate in Order Header is earlier than requiredByDate in line item, then the start time used is the one at order line level

RequiredBydate Amendment type allows for changing the required date for an order when it is not in its FINAL stages as mentioned earlier. The following matrix defines the conditions to identify a RequiredByDate change amendment type:

Original header date	Original line date	New header date	New line date	IsAmendment
Past Dated	Past Dated	past dated but greater than originalheader date	past dated but greater than originalheader date	No
Past Dated	Past Dated	Same as original	Future Dated	Yes, for that particular Order Line
Past Dated	Past Dated	Future Dated	Same as original	Yes, for all order lines
Future Dated	Past Dated	Back Dated	Same as original	Yes, for all order lines
Future Dated	Past Dated	Future date than original	Same as original	Yes, for all order lines
Past Dated	Future Dated	Same as original	Same as original	No
Past Dated	Past Dated	Future Dated	Future Dated	Yes, for all order lines. The time dependency is calculated as explained earlier.
No Date	Past Date	Back dated	Same as original	No
No Date	Future Date	Back dated	Same as original	No
No Date	No Date	Future Dated	Future Dated	Yes, for all order lines. The time dependency is calculated as explained earlier.

The default behavior in 2.1.1 for required by date change is not to create compensation or restart any plan items. Below matrix defines the amendment behavior based on plan item status

Plan Item Status	Description
Pending	Plan item dependency time is updated so the plan item triggers at the amended required by date.
Suspended	Not permitted. Any required by date changes are ignored. As the plan item is already started, it is not possible to change the start date.
Complete	Not permitted. Any required by date changes are ignored. As the plan item is already completed, it is not possible to change the start date.

The value of roll back user-defined field in order is ignored in this case as no compensation or restart plan items are created.

OrderLine user-defined field Change

OrderLine user-defined field change is an order amendment type where the amended order lines have changed only their corresponding User Defined Fields. All other attributes remain unchanged. This application is able to identify if the orders have changed only with respect to User Defined Fields by inspecting the orderlines to identify if the User Defined Fields have been modified, added or removed. This way of identifying the amended user-defined field is different from TIBCO Order Management 2.0.1 and prior, where user-defined field only change was identified by checking value of special UDF (MODIFICATION_IDENTIFYING_ATTR) passed in the order line. Starting from TIBCO Order Management 2.1.0, there is no need to pass MODIFICATION_IDENTIFYING_ATTR UDF to tell the application, which user-defined field is being modified.

Here is the sample orderline showing the changes between TIBCO Order Management 2.1.0 and TIBCO Order Management 2.0.1 and prior:

Sample OrderLine in TIBCO Order Management 2.0.1

```
<ord1:line>
  <ord1:lineNumber>1</ord1:lineNumber>
    <ord1:productID>MODEM</ord1:productID>
    <ord1:productVersion>1.0</ord1:productVersion>
    <ord1:quantity>1</ord1:quantity>
    <ord1:uom>UOM</ord1:uom>
    <ord1:action>PROVIDE</ord1:action>
    <ord1:actionMode>MOVE</ord1:actionMode>
    <ord1:requiredByDate>2011-04-30T13:20:00-05:00</
ord1:requiredByDate>
    <ord1:inventoryID>1</ord1:inventoryID>
    <ord1:notes>NOTES</ord1:notes>
    <ord1:slaID>SLA_ID</ord1:slaID>
    <ord1:udf>
      <ord1:name>MODIFICATION_IDENTIFYING_ATTR</ord1:name>
      <ord1:value>Region</ord1:value>
    </ord1:udf>
    <ord1:udf>
      <ord1:name>Region</ord1:name>
      <ord1:value>Antarctica</ord1:value>
    </ord1:udf>
</ord1:line>
```

Sample Order Line in TIBCO Order Management 2.1.0

```
<ord1:line>
  <ord1:lineNumber>1</ord1:lineNumber>
  <ord1:productID>MODEM</ord1:productID>
  <ord1:productVersion>1.0</ord1:productVersion>
  <ord1:quantity>1</ord1:quantity>
  <ord1:uom>UOM</ord1:uom>
  <ord1:action>PROVIDE</ord1:action>
  <ord1:actionMode>MOVE</ord1:actionMode>
  <ord1:requiredByDate>2011-04-30T13:20:00-05:00</
ord1:requiredByDate>
  <ord1:udf>
    <ord1:name>Region</ord1:name>
    <ord1:value>Asia</ord1:value>
  </ord1:udf>
</ord1:line>
```

Identifying user-defined field only Amendment

TIBCO Order Management checks the following conditions to identify user-defined field only change amendment scenario. All the following conditions, which are mentioned, holds true for user-defined field Only Change Amendment:

- Number of order lines in initial order must match number of order lines in amended order.
- The product Id in order line in Initial Order must match with the product Id in corresponding order line of amended order.
- The Action in order line in Initial Order must match with the Action in corresponding order line of amended order.
- The RequiredByDate in order line in Initial Order must match with the RequiredByDate in corresponding order line of amended order.

New Execution Plan Modification Rules Characteristics

Starting with the version number 2.1.0, the application provides more granular Execution Plan Modification Rules actions to be configured for user-defined field modifications based on the status of the plan items, so as to have more control when generating the COMPENSATE or REDO plan items.

The format of new Execution Plan Modification Rules characteristics are:

1. EPMR_ACTION_<<action>>_UDF_CHANGE: Using this format Execution Plan Modification Rules action can be configured on per Amendment Type. The supported values of <<action>> are:
 - a. PROVIDE
 - b. CEASE
 - c. UPDATE
 - d. WITHDRAW

The following is an example of the characteristic, configured in product model with Execution Plan Modification Rules:

```
<ns0:characteristics>
  <ns0:name>EPMR_ACTION_PROVIDE_UDF_CHANGE</ns0:name>
  <ns0:description>Characteristic</ns0:description>
  <ns0:instanceOptional/>
  <ns0:instanceCeaseSequence/>
  <ns0:instanceUpdateSequence/>
  <ns0:instanceSequence/>
  <ns0:instanceMin>0</ns0:instanceMin>
  <ns0:instanceMax>0</ns0:instanceMax>
  <ns0:evaluationPriority/>
  <ns0:value>
```

```

        <ns0:type>PROVIDE</ns0:type>
        <ns0:discreteValue>COMPENSATE_RESTART</ns0:discreteValue>
        <ns0:mandatoryValue>true</ns0:mandatoryValue>
    </ns0:value>
    <ns0:simpleRule>
        <ns0:name>EPMR_ACTION_PROVIDE_UDF_CHANGE</ns0:name>
        <ns0:ruleSetOutcome>Characteristic</ns0:ruleSetOutcome>
    </ns0:simpleRule>
</ns0:characteristics>

```

2. EPMR_ACTION_<<action>>_UDF_CHANGE_<<Plan Item Status>>: Using this format, Execution Plan Modification Rules action can be configured on per Amendment Type and Plan Item Status . The supported values of Plan Item Status are:
 - a. COMPLETE
 - b. SUSPENDED
 - c. PENDING
 - d. EXECUTION

The following is an example of the characteristic, configured in product model with Execution Plan Modification Rules:

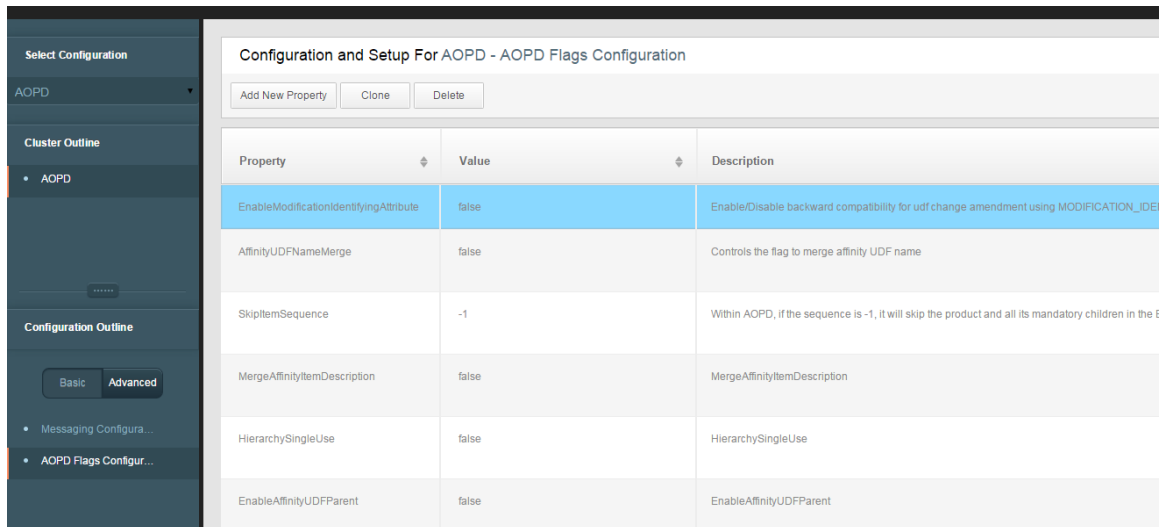
```

<ns0:characteristics>
    <ns0:name>EPMR_ACTION_PROVIDE_UDF_CHANGE_SUSPENDED</ns0:name>
    <ns0:description>Characteristic</ns0:description>
    <ns0:instanceOptional/>
    <ns0:instanceCeaseSequence/>
    <ns0:instanceUpdateSequence/>
    <ns0:instanceSequence/>
    <ns0:instanceMin>0</ns0:instanceMin>
    <ns0:instanceMax>0</ns0:instanceMax>
    <ns0:evaluationPriority/>
    <ns0:value>
        <ns0:type>PROVIDE</ns0:type>
        <ns0:discreteValue>COMPENSATE_RESTART</ns0:discreteValue>
        <ns0:mandatoryValue>true</ns0:mandatoryValue>
    </ns0:value>
    <ns0:simpleRule>
        <ns0:name>EPMR_ACTION_PROVIDE_UDF_CHANGE</ns0:name>
        <ns0:ruleSetOutcome>Characteristic</ns0:ruleSetOutcome>
    </ns0:simpleRule>
</ns0:characteristics>

```

Backward Compatibility with TIBCO Order Management 2.0.1

TIBCO Order Management 3.0.0 supports the use of MODIFICATION_IDNETIFYING_ATTR udf to denote the user-defined field being changed through the use of a flag. This flag, called EnableModificationIdentifyingAttribute, can configured from the Configurator UI is as shown in the following figure:



Property	Value	Description
EnableModificationIdentifyingAttribute	false	Enable/Disable backward compatibility for udf change amendment using MODIFICATION_IDE
AffinityUDFNameMerge	false	Controls the flag to merge affinity UDF name
SkipItemSequence	-1	Within AOPD, if the sequence is -1, it will skip the product and all its mandatory children in the t
MergeAffinityItemDescription	false	MergeAffinityItemDescription
HierarchySingleUse	false	HierarchySingleUse
EnableAffinityUDFParent	false	EnableAffinityUDFParent

The default value of this flag is FALSE.

Predefined User Defined Fields

Changes in following User Defined Fields are ignored by the application:

- ORDERLINE
- GLOBAL_PRODUCT_NAME
- EOL
- ACTION
- M_EPS_UDFS



The changes done only in the User Defined Fields at the order header level in an amendment request doesn't have any impact on the existing plan in terms of the creation of compensatory and redo plan items. There are no changes in the dependencies between the plan items either. However, the amendment plan contains the updated value of the User Defined Fields. The plan items, which go into EXECUTION post amendment is able to get the updated value of the header level User Defined Fields using GetPlan JMS data interface or GetOrderExecutionPlan service.

OrderPriority Change

The orderPriority change in an amendment request doesn't have any impact on the existing plan in terms of the creation of compensatory and redo plan items. There is no changes in the dependencies between the plan items either. However, once the amendment plan is activated, the updated value of the orderPriority is passed as *JMSPriority* in all further outbound JMS messages corresponding to that order so as to prioritize the fulfillment of the order.

OrderLine Addition

In this amendment type, one or more new order lines can be added as part of the amendment request to fulfill the additional products. The plan items corresponding to the newly added order lines are added into the execution plan and the dependencies are updated accordingly. At a high level, there are two main cases.

1. If an optional child product from a ProductComprisedOf relationship was ordered in the original request and the parent product is then ordered in a new order line in the amendment request, the newly created plan item for parent product have a dependency on the existing plan item of the child product.

For example, if B is an optional child product of A, in case of the above mentioned scenario; the newly added plan item for A have a dependency on the plan item of B, which was there in the original plan. This is done regardless of the status of the plan item for child product B.

This is logical and would have been the case even when both products were ordered in the original request itself. Once the amendment plan is activated, the plan item for the parent product goes into EXECUTION after the plan item for child product is successfully completed. If the plan item for child product was already COMPLETE, the plan item for parent product goes into EXECUTION immediately.

2. On the other hand, if a parent product from a ProductComprisedOf relationship was ordered in the original request and the child product is then ordered in a new order line in the amendment request, the dependency of the child plan item is added based on the status of the parent plan item, as explained in the following points:
 - a. If the parent plan item was PENDING before the amendment, the dependency is added into the existing parent plan item.
 - b. If the parent plan item was SUSPENDED or COMPLETE before the amendment, a REDO plan item is generated against it. The original parent plan item is canceled if it was SUSPENDED. A REDO plan item is generated based on the Execution Plan Modification Rules action configurations as mentioned in the following points:
 - a. If the value configured in the Execution Plan Modification Rules characteristic for the corresponding order line action is either RESTART or COMPENSATE_RESTART. E.g. In case of PROVIDE action, the value of EPMR_ACTION_PROVIDE characteristic is referred.
 - b. Or the required Execution Plan Modification Rules characteristic is missing in the product model and the CompensateRestartForNoEPMRChar flag is enabled in Automated Order Plan Development configurations.

See [Execution Plan Modification Rules \(EPMR\)](#) for details about Execution Plan Modification Rules actions. The dependency of the child plan item is added into the newly created REDO plan item for the parent product. Once the amendment plan is activated, the newly added plan item for the child product goes into EXECUTION. After it is successfully completed, the REDO plan item for parent goes into EXECUTION.

Preconditions for OrderLine Addition

Following is the only precondition for the order line addition amendment type.

The lineID, productID, requiredByDate, and User Defined Fields in all the order lines in the amendment request must match with those in the original order request.



Unlike addition, the deletion or removal of an existing order line is not allowed and supported in an amendment request. For canceling the fulfillment of the product in an order line, the order line action must be changed to CANCEL instead.

Execution Plan Modification Rules (EPMR)

The execution plan, for the amendment types mentioned earlier, is modified based on the predefined rules that are specified as values in the following characteristics in the product model:

1. EPMR_ACTION_PROVIDE
2. EPMR_ACTION_Cease
3. EPMR_ACTION_UPDATE
4. EPMR_ACTION_WITHDRAW

Only one of the appropriate characteristics, based on the action passed in the original order, is referred to when applying the modifications on the execution plan. For user-defined field change functionality, additional set of characteristics can be defined to have a granular control based on the status of the plan item.

As mentioned in earlier, these rule actions are applied on the plan items that are either in SUSPENDED or COMPLETE state. There are four standard Execution Plan Modification Rules actions, which are explained in

the following paragraphs. Only one of the four actions can be specified as the value for a particular Execution Plan Modification Rules characteristic for a particular product.

COMPENSATE_RESTART

This Execution Plan Modification Rules action is assigned as the value of an Execution Plan Modification Rules characteristic if a compensatory and a redo plan item is to be created against an existing plan item as a part of the amendment processing.

Compensatory Plan Item

The purpose of a compensatory plan item is to compensate, or reverse, the tasks that were done by the existing plan item before the amendment request was initiated. The important aspect of a compensatory plan item are as follows:

1. The planItemId of the compensatory plan item is derived using the planItemId of the existing plan item and has the following format: COMP-<amendment number>_<planItemId of the existing plan item>. For example, if the planItemId of the existing plan item is 04ceddc8-60fc-4800-82b9-4f3382400000 and a compensatory plan item is created against it during the first amendment request, the planItemId assigned to that compensatory plan item is COMP-1_04ceddc8-60fc-4800-82b9-4f3382400000.
2. The action and planFragmentUniqueID (processComponentID) in the compensatory plan item is assigned on the basis of the action in the existing plan item, which is described in the following table:

Action in Existing Plan Item	Action in Compensatory Plan Item	processComponentID in Compensatory Plan Item
PROVIDE	CEASE	Value of Characteristic COMPENSATE_PROVIDE
UPDATE	UPDATE	Value of Characteristic COMPENSATE_UPDATE
CEASE	PROVIDE	Value of Characteristic COMPENSATE_Cease



If the required COMPENSATE_<ACTION> characteristic (for example COMPENSATE_PROVIDE) is not present in the product model, the regular plan fragment specified for CANCEL action is assigned.

3. The compensatory plan item, by default, have a simple END-START point dependency on the existing plan item being canceled, as shown in the following figure. This is to ensure that the compensatory task must be started only after the existing task, being executed, is activated and canceled.

Dependency between the compensatory plan item COMP-1_P1 and the existing plan item P1 that is canceled



To enable the backward compatibility of having no dependency in the compensatory plan items in TIBCO Fulfillment Order Management 2.0.x, the flag NoDependencyInCOMPPlanItems must be set in

Automated Order Plan Development configurations. Refer topic [Amendment Configuration Flags](#) to understand the significance of each flag.

4. The action and the processComponentID in the existing plan item is set to CANCEL and NO_RECIPROCAL_ACTION respectively so as to cancel the existing plan item. Note that the Orchestrator changes the processComponentID to NO_RECIPROCAL_ACTION only for the PENDING plan items that are canceled. The processComponentID for the SUSPENDED plan items remain the same.

A compensatory plan item, if requiring creation, is created always against a regular plan item in the first amendment. However, in the subsequent amendment requests, it can be created against a regular or a REDO plan item from the earlier amendment based on the execution plan at that point of time, as shown in the following figure. A compensatory plan item is never created against another compensatory plan item that was created during the last amendment.

Dependency between the compensatory plan item COMP-2_REDO-1_P1 created during second amendment and the REDO plan item from the last amendment REDO_P1, which is canceled.



Redo Plan Item

The sole purpose of a redo (restarting) plan item is to restart or redo the tasks that were supposed to be done in the existing plan item before the amendment request was initiated. The important aspects of a redo plan item are as follows:

1. The planItemId of the redo plan item is derived using the planItemId in the existing plan item and has the following format: REDO-<amendment number>_<planItemId of the existing plan item>. For example, if the planItemId of the existing plan item is 04ceddc8-60fc-4800-82b9-4f3382400000 and a redo plan item is created against it during the first amendment request, the planItemId assigned to that redo plan item is REDO-1_04ceddc8-60fc-4800-82b9-4f3382400000.
2. The action in the redo plan item is the one that is passed in the corresponding order line in the amendment request.
3. The planFragmentUniqueID (processComponentID) in the redo plan item is the same as the one in the existing plan item, except in case of order line action change amendments. In such cases, the plan fragment associated with the action in the amendment request is assigned in the redo plan item.
4. The redo plan item always have a simple END-START point dependency on the compensatory plan item that is created, as shown in the following figure. This ensures that the designated tasks is restarted only after the compensatory tasks are finished.

Dependency between the REDO plan item REDO_P1 and the compensatory plan item COMP-1_P1



A redo plan item, requiring creation, is always created against a regular plan item in the first amendment. However, in the subsequent amendment requests, it can be created against a regular or a REDO plan item from the earlier amendments based on the execution plan at that point of time, as shown in the following figure. A redo plan item is never created against a compensatory plan item that was created during the last amendment.

Dependency between the redo plan item REDO-2_REDO-1_P1 created during second amendment and the REDO plan item from the last amendment REDO_P1, which is canceled



In case of OrderLine Cancellation or Entire Order Cancellation, even if the Execution Plan Modification Rules action is COMPENSATE_RESTART, only compensatory plan item is created. There is no need to create a redo plan item as the order line, or the entire order is canceled.



RESTART is not a logical Execution Plan Modification Rules action in case of an order line or an entire order cancellation. If RESTART action is encountered when processing the order line or the order cancellation, no action is taken on the corresponding plan item. A relevant message is logged, instead, to report the same.

COMPENSATE

This Execution Plan Modification Rules action is assigned as the value of an Execution Plan Modification Rules characteristic if only a compensatory plan item is to be created against an existing plan item as a part of the amendment processing.

See [Compensatory Plan Item](#) for understanding all the aspects of a compensatory plan item.

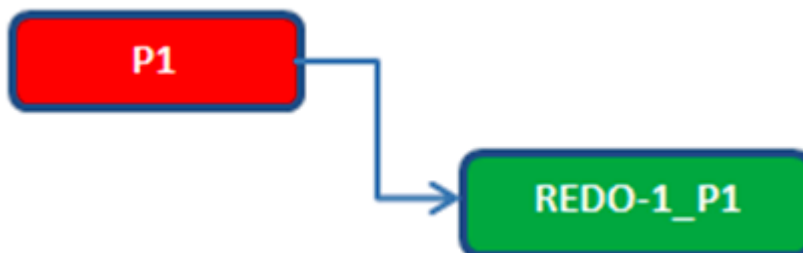
RESTART

This Execution Plan Modification Rules action is assigned as the value of an Execution Plan Modification Rules characteristic if only a redo plan item is created against an existing plan item as a part of the amendment processing.

See [Redo Plan Item](#) for understanding all the aspects of a redo plan item.

The redo plan item always have a simple END-START point dependency directly on the existing plan item that is going to be activated and canceled, due to the non-existence of a compensatory plan item as shown in the following figure. This is to ensure that the designated task must be restarted only after the cancellation of the existing task

Dependency between the REDO plan item REDO_P1 and the existing plan item P1, which is canceled



IGNORE

This Execution Plan Modification Rules action is assigned as the value of an Execution Plan Modification Rules characteristic, if no explicit action is required to be taken against an existing plan item as part of the amendment processing. In case of OrderLine Cancellation or Order Cancellation, the planFragmentUniqueID (processComponentID) of the plan item is set to NO_RECIPROCAL_ACTION.

No Execution Plan Modification Rules Characteristic in Product

In case a product model does not contain the required Execution Plan Modification Rules characteristic, then behavior of amendment to generate redo or compensate plan item can be controlled using the flag, CompensateRestartForNoEPMRChar, in Automated Order Plan Development configurations. See the topic [Amendment Configuration Flags](#) for this flag.

Amendment Configuration Flags

The following are the flags available in Automated Order Plan Development configurations to tweak some of the functionalities around order amendments:

1. EnableModificationIdentifyingAttribute

This flag, if true, enables the OrderLine user-defined field modification functionality using the MODIFICATION_IDENTIFYING_ATTR characteristic as it was in 2.0.x.

2. NoDependencyInCOMPPlanItems

This flag, if true, enables the backward compatibility to 2.0.x of having no dependency of the existing plan item being canceled, in the compensatory plan item. The compensatory plan item immediately goes into execution along with the activation request of the existing plan item.

3. EnableDateShiftCompRedo

This flag, if true, enables the backward compatibility to 2.0.x version of creating compensatory and redo plan items in case of requiredByDate change (Date Shift) type amendments. This value of roll back user defined field controls the behavior at runtime. The default value of roll back is true and the behavior is:

- Compensation and Restart plan items is created as per the Execution Plan Modification Rules characteristics for suspended and completed plan items.
- The original completed and suspended plan items do not have the new requiredByDate. New requiredByDate (date shift) is set for the corresponding "Redo" plan items.
- In case of pending items, the requiredByDate of that pending plan item is changed to the new requiredByDate. No Compensate or Redo Plan items is generated.

If mentioned as false, then

- Compensation and restart plan items are not created.
- Completed and suspended plan items do not contain the changed required by date. Only pending plan items have the new date.

The behavior of requiredByDate amendments for compensation and restart plan items for Execution Plan Modification Rules characteristics is consistent with implementation of other amendment types configured for 2.0.x in this release.

4. CompensateRestartForNoEPMRChar

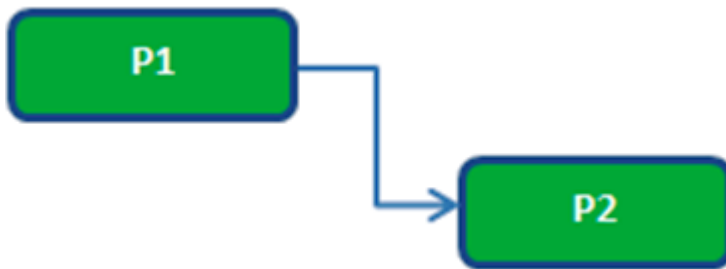
This flag, if true, considers COMPENSATE_RESTART as the Execution Plan Modification Rules action in case of the required Execution Plan Modification Rules characteristic not present in the product model. If this flag is false and the required Execution Plan Modification Rules characteristic is also not present in the product model, no action is taken on the plan items associated to that product, which are in COMPLETE or SUSPENDED state.

Impact on Dependencies

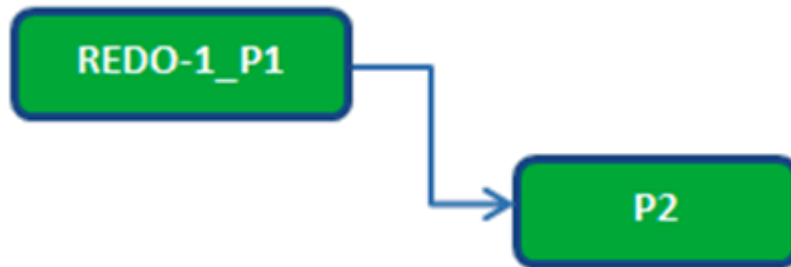
If both compensatory plan items and redo plan items or either of them are created against one or more existing plan items when processing an amendment request, the dependencies in the overall execution plan are impacted. The dependency on the existing plan item being canceled is implicitly added in the compensatory and/or redo plan item when they are created. There can be some additional dependencies in the redo plan items in certain cases. Also, the dependencies in other regular plan items are modified, if required. The following points explain these modifications:

1. If a parent plan item in PENDING state, which is not being canceled, has a dependency on such a child plan item against which a COMP and REDO plan items have been created, the dependency on the existing plan item is removed and a new dependency is added on the REDO plan item, as shown in the following figures. The REDO plan item has higher priority over the COMP plan item when replacing the dependency on the corresponding existing plan item. If the REDO plan item does not exist, the dependency on the existing plan item is replaced with a dependency on the COMP plan item. This keeps the dependency structure in the amendment plan in-line with the earlier plan.

Dependency on plan item P1 in PENDING plan item P2 in the original plan



Dependency on the first level REDO plan item of P1 in PENDING plan item P2 in the amendment plan

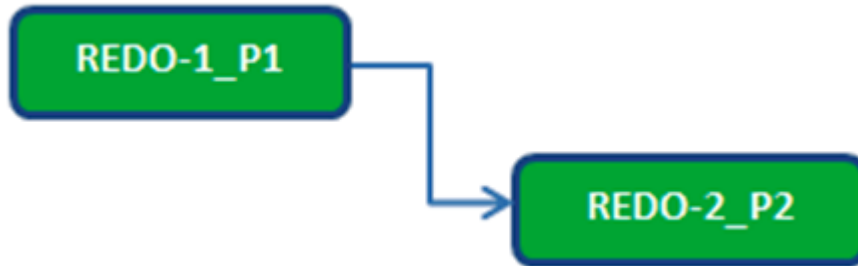


2. If REDO plan items have been created against an existing parent and child plan item, then the same dependency is maintained between the corresponding REDO plan items. The parent REDO plan item have a dependency on the child REDO plan item, in addition to the dependency on the existing original plan item being canceled, as shown in the following figures:

Dependency on plan item P1 in plan item P2 in the original plan

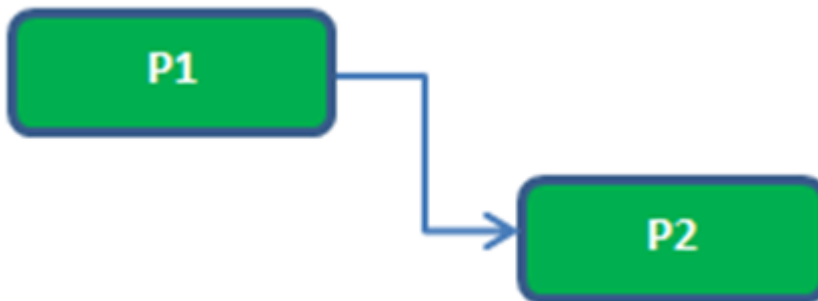


Dependency on REDO plan item P1 in REDO plan item P2 in the amendment plan



3. If COMP plan items have been created against an existing parent plan item and child plan item, a reverse dependency is maintained between the corresponding COMP plan items in the amendment plan. The child COMP plan item has a dependency on the parent COMP plan item, as shown in the figure. It is done to ensure that the exact compensation of the plan items, which is, the compensation of parent plan item occurs first, followed by the compensation of child plan item.

Dependency on plan item P1 in plan item P2 in the original plan



Dependency on COMP plan item P2 in COMP plan item P1 in the amendment plan



Multiple Amendments

TIBCO Order Management allows multiple amendments of an order however not concurrently. This means that the order, which is currently being amended cannot be amended again at the same time. Once the existing amendment request is successfully processed by TIBCO Order Management Long Running and the new plan is activated, the order can be very well amended again, provided the amendment conditions are satisfied. Each amendment request for an order is processed in the same way as explained in section [Amendments Workflow](#).

The planItemId assigned to a compensatory or a redo plan item created during an amendment contains the amendment number as one of the prefixes. Refer sections [Compensatory Plan Item](#) and [Redo Plan Item](#) for the knowing the planItemId format.

Order Priority

This section describes about the order priority process.

Understanding Order Priority

The OrderPriority enables the user to set priority on submitted orders. This information is then used by JMS broker to deliver the high priority orders to downstream components on priority. The order priority is also propagated to downstream process components. The priority value ranged from 0 to 9. The priority information or priority value to process any given order is set in the JMSHeader field of the JMS message, which is sent to the following Order Management Long Running components:

- Orchestrator engine
- Automated Order Plan Development engine

The JMS broker then delivers the order based on a priority.

The process of order prioritization works at a queue level and can be controlled by JMS broker.



The order priority process or order prioritization cannot be controlled once the message is delivered to the Orchestrator engine or Automated Order Plan Development engine. The priority value can be changed before JMS broker sends the order request to the engines.

Order Schema Changes

The order schema allows you to submit the orderPriority information with the order. The orderPriority is at the orderHeader level and the same priority is applied to all the orderLines.

The schema snippet is as follows:

```
<xs:element name="orderPriority" minOccurs="0" default="4">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="9" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

The orderPriority can take values in range from 0 to 9 to make consistent and map with JMSPriority message header values.



The default value of the orderPriority field is 4.

Lower Priority Orders

When any order is processed based on the given priority, it results in a situation where a lower priority orders might never get a chance to be processed because of high priority orders.

The orders with the lower priority can be processed by a mechanism known as the Flow Control mechanism.

The Enterprise Messaging Service (EMS) allows the user to control the flow of messages to a destination. Each destination can specify a target maximum size for storing all the pending messages. When the target is reached, Enterprise Message Service blocks message producers when new messages are sent. This effectively slows down message producers until the message consumers can receive the pending messages.

Custom Action

In the Fulfillment Order Management 2.0.0 release, you can define the set of Actions to provide a way to define any number of unique fulfillment actions. Until this release, only four set of actions were supported.

Custom actions are loaded into Automated Order Plan Development as Action Models and is referred to at the time of plan generation.

Custom action enables you to submit an order for custom actions, depending on which Automated Order Plan Development assigns the planfragment.



The planfragment is selected based on the `PlanFragmenthasCustomAction` relationship from the product datamodel.

Manual Order Plan Development

If Manual Order Plan Development is enabled and the order has the configured property for identifying the order for Manual Order Plan Development, then the plan for that particular order can be manually modified in OPD state.

The orders for Manual Order Plan Development can be manually modified in regular fulfillment flow and the amendment flow as well.

The following operations can be performed within the TIBCO Order Management user interface:

- Searching Orders for Manual Order Plan Development
- Modifying the Plan in Draft Mode through Grid View
- Modifying the Plan in Draft Mode through Gantt View
- Saving the Modified Plan
- Saving and Executing the Modified Plan
- Discarding Changes
- Modifying Plan in case of Amendments

Searching Orders for Manual Order Plan Development

To search orders for Manual Order Plan Development perform the following steps:

1. Browse the **Orders** tab and click the **Filter** icon.

[Browse Orders](#)

2. Select **Manual Order Plan Development** in the OPD Source and click **Save**. All the orders, which are in OPD for manual changes, and the orders, which executed Manual Order Plan Development is listed.

Filter

Condition

([OPD Source] [=] 'MOPD')

Attributes	Operator	Value
Order ID	=	AOPD
Order Ref ID	like	MOPD
Customer ID	in	
Status		
Submitted Date		
Originator		
Fulfillment Engine		
OPD Source		

Buttons: Add Condition, Reset, Save, Reset Condition, Cancel


3. Click the order for which plan has to be manually modified. Order details for that particular order is populated and user gets an option **Show Manual Plan** on the top bar. [Show Manual Plan](#)

Browse Orders

[Add Order](#)
[Show Manual Plan](#)
[Show Activity Log](#)

Search (DPO Source) (+) (WORD)

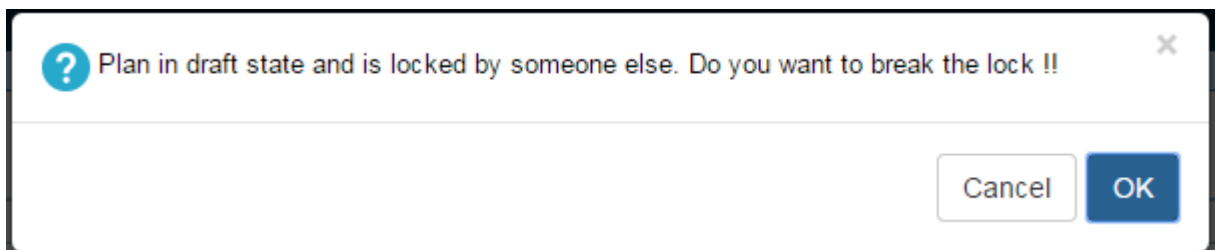
Order Ref ID	Customer ID	Status	Submitted Date	Originator	Fulfillment Engine
ORD_25062014_01	DUMMY_PARTY	OPD	06/25/2014 11:59:31 UTC+5:30	OMS	AFO
ORD_18062014_Simp001	DUMMY_PARTY	OPD	06/18/2014 17:23:35 UTC+5:30	OMS	AFO
ORD_13062014_01	DUMMY_PARTY	OPD	06/13/2014 17:48:28 UTC+5:30	OMS	AFO
ORD_12062014_01	DUMMY_PARTY	OPD	06/12/2014 13:57:22 UTC+5:30	OMS	AFO

4. Click **Show Manual Plan** and the Automated Order Plan Development generated plan is shown for modifications. Initially the plan shown for editing is in non-draft mode. The user needs to bring the plan in draft mode to get the options for modifying the plan. All the plan and plan-items are in START state and milestones are in PENDING state. 

[Plan](#)
[Custom Headers](#)

Plan ID	Description	StaticBundle
5a6714b-b260-4bfb-bd4f-cbc6	Orchestrator	06/25/2014 11:59:37 UTC+5:30
3823782a-e6b3-46b3-8934-a172	1282533e-0944-48b8-9d53-6d01f3d1c4	ORD_25062014_01
	Status	06/25/2014 11:59:37 UTC+5:30
	Is Amendment?	06/27/2014 17:20:04 UTC+5:30
	Risk Region	None

5. Click the **Draft Plan** icon and you receive options for modifying the plan. There might be a possibility that some other user is already accessing the plan and modifying it. In such cases you are prompted for confirmation on breaking the lock from the other user who is accessing it. If you choose to break the lock then the unsaved changes of the other user is removed.



Modifying the Plan in Draft Mode through Grid View

Click **Draft Plan** in grid view. You can see the Add and Remove options on top of the tree in the grid.

You can perform the following actions:

1. Create a new plan item
2. Create a new milestone
3. Delete plan item
4. Delete milestone
5. Modify plan item
6. Modify milestone
7. Creating new dependencies
8. Deleting dependencies
9. Validations

Create a New Plan Item

Click **Plan** in the tree and click **Add**. A new plan-item is created. For each plan-item you can have option of modifying or adding following values:

Plan Item Tab

In the **Plan Item** tab you can associate a product and action to the new plan item.

Plan Item	Custom Headers	Order Line	Process Info	Sections
Product ID	CFS_TV		Description	EP_PF_CFS_TV_Provide
Action	PROVIDE		Status	START

When you click inside the **Product ID** text box or the icon beside it, you are given list of all the products available in the Order Management Server repository.

You can browse through the entire list of products by using pagination or can directly search for the desired product by keying in the product ID in the search box. This search is case sensitive. When you get the desired product in the list, select the product and click the **OK** button. The selected product is displayed in the **Product ID** text box and all the **Action** or **Owner** associated to the selected product is fetched in the **Action** drop-down box.

If you click on the cancel button in the Product ID popup window, no product is selected. Either the Product ID text box is displayed blank or the earlier product is retained in the text box.

Product Id	Product Description
PROD14	PROD14
PROD15	PROD15
PO_PROD1_TFOM-506	PO_PROD1_TFOM-506
CFS_PROD1_TFOM-506	CFS_PROD1_TFOM-506
CFS_TV	CFS_TV
Product_TV	Product_TV
TARIFF選機語	TARIFF選機語
ROUTER選機語	ROUTER選機語
MODEM選機語	MODEM選機語
BPO_DUO	BPO_DUO

As you associate the correct action or owner to the newly created plan-item, **Process Info** tab and the **Sections** tab is populated for the plan item.



If the originally submitted order consists of an Affinity Plan Item, the products is displayed comma separated in Product Textbox. If the user modifies the product by clicking on the textbox or the icon beside it, the user is not able to revert and get back the affinity products through Manual Order Plan Development.

Custom Headers Tab

When the user creates a new plan item, the default or system defined custom headers (User Defined Fields) are already created for the user and you are not allowed to modify these. Values for these User Defined Fields are populated as per the selections in different tabs of the plan item.

milestone sections defined in the plan fragment model. Adding a new intermediate milestone is always dependent on sections (sections of milestones) defined in the plan fragment model. This rule is used to add a new milestone. Every plan item is associated to a product and an action or owner by this information we identify the plan fragment, which can be associated to the plan item and if this plan fragment has correct intermediate milestones defined then you are able to create new intermediate milestones. In case there is no intermediate milestone section defined for this particular plan fragment associated to the plan item then you won't be able to create any new milestone.



You can confirm the sections, which are defined for the plan fragment by clicking the plan item in the tree and selecting **Sections** tab.

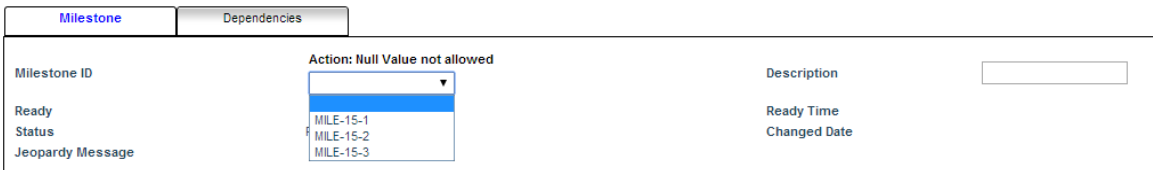
Considering that the plan fragment you have associated has correct intermediate milestones defined then, select the milestone folder in the tree and click **Add**.

A new milestone is created with a Dummy<id> name in the tree and you have to select the milestone ID from the available milestone ID list in the drop-down box.



For every intermediate milestone assigning a correct milestone ID from available milestone list and creating at least one dependency to or from this intermediate milestone is mandatory. Any intermediate milestone created must be part of the flow of the plan that is the intermediate milestone must have at least one dependency on it, or from it.

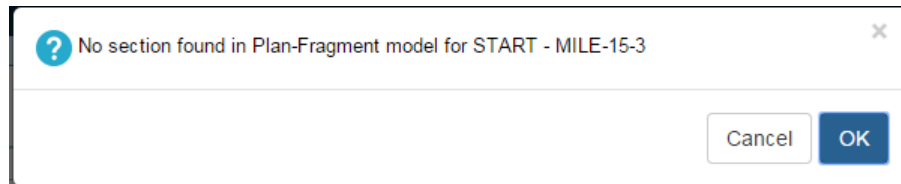
When selecting the milestone ID from milestone ID list, you are expected to select the milestone ID whose section with the existing milestone exists in the plan fragment model.



You have Mile-15-1, Mile-15-2, and Mile-15-3 in the milestone ID list and in the plan fragment model you have defined following sections:

START – Mile-15-1, Mile-15-1 – Mile-15-2, Mile-15-2 – Mile-15-3, Mile-15-3 – END, Mile-15-1 – END, Mile-15-2 – END.

Now you are trying to create first intermediate milestone and you selected Miles-15-3 as the milestone ID.



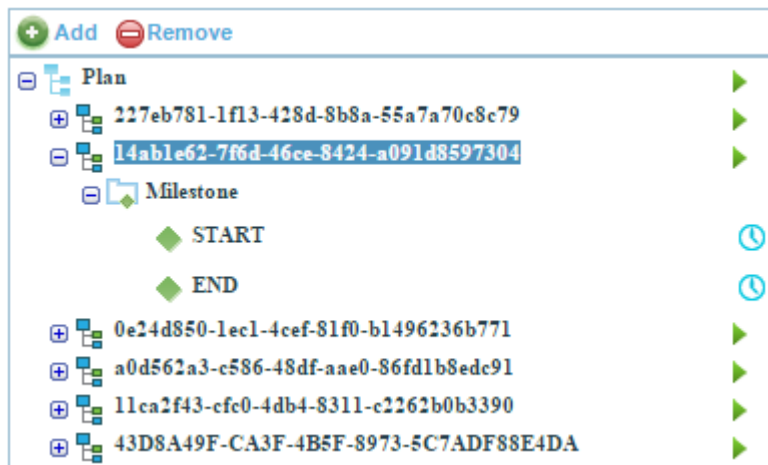
You get the error message showing all the sections that were not found in the plan fragment model when creating the specified milestone, in this case message shows it didn't find START – Mile-15-3 section so Mile-15-3 won't be accepted as a first milestone between START and END.

When you try to select Mile-15-1 as the new milestone-id, it won't give any error as there is existing section for START – Mile-15-1 and Mile-15-1 – END.

Delete Plan Item

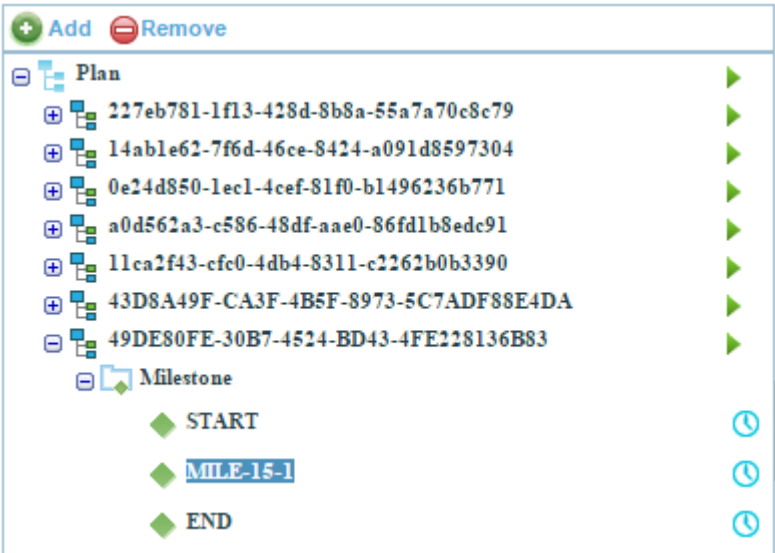
Deleting any plan item would mean deleting all the associated milestones and dependencies on or from those milestones.

Select the plan item you want to delete and click **Remove**.



Delete Milestone

You cannot delete START or END milestone. You can select any intermediate milestone and delete it. Deleting any intermediate milestone deletes all the dependencies on or from that milestone.



Modify Plan Item

When the plan is in draft mode, you can click any plan item in the tree and modify the plan item.

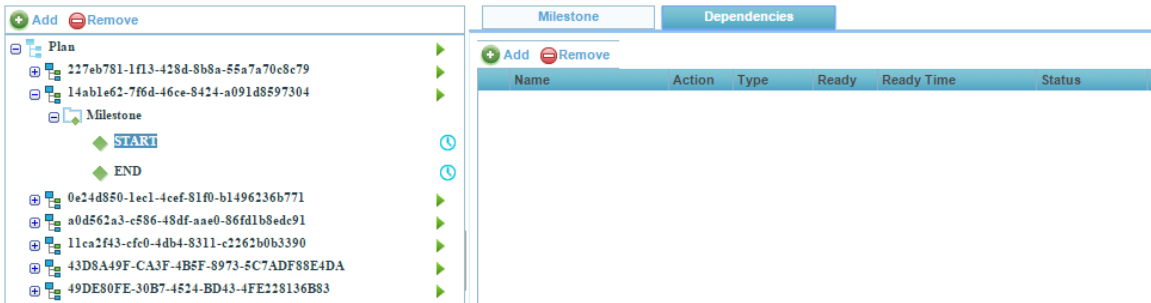
Modify Milestones

When the plan is in draft mode, you can click any milestone in the tree and modify the milestone.

Creating New Dependencies

Manual OPD supports the creation of two types of dependencies: Point and Time.


Select any milestone other than END milestone in the plan tree and select the **Dependencies** tab. You can see Add and Delete options in the dependencies tab. When you create a dependency you are creating a Dependent-On type of relationship between the milestones, which means when you select a milestone from the tree, as shown in the example START milestone was selected in the tree and when creating the dependency you are trying to define that this selected START milestone is Dependent-On which other milestone from other plan-items.



You cannot create any new dependency from END milestone.

Select a valid milestone (other than END milestone) in the tree and click on “Add” in dependencies tab, an empty dependency row is created and you are expected to create a point or time dependency.

Creating Point Dependencies


Click the Add new point dependency  icon to create a new point dependency. A popup is displayed with all the plan-items containing milestones on which you can create the dependency.

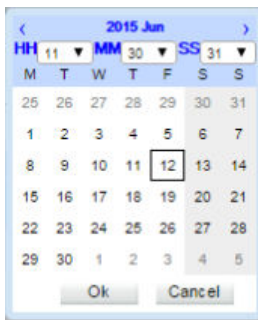
Choose the appropriate milestone from the tree on which you want to create the dependency. In the dependency-tree widget you are able to select all the milestones other than the START milestone. You cannot create dependency on START milestone.

Select the appropriate milestone from dependency-tree widget and click the **OK** button. A point dependency is created in the empty row created earlier.

Milestone		Dependencies									
<div><div><div><div></div><div>Add</div></div><div><div></div><div>Remove</div></div></div></div>											
Name	Action	Type	Ready	Ready Time	Status	Status Changed	Time Delta	Plan	Plan Item	Milestone	
<div><div><div></div><div>D4C9DED8-A9C6-488C-A54D-638BAFC44E19</div></div></div>	<div><div><div></div><div></div><div></div></div></div>		false								

Creating Time Dependencies

Click on the Add new time dependency  icon to create a new time dependency. A date time picker popup is displayed.



Select the appropriate date and time to create the time dependency.

After selecting the appropriate date and time from the picker, click the **OK** button and a time dependency is created in the empty dependency row you had created earlier.

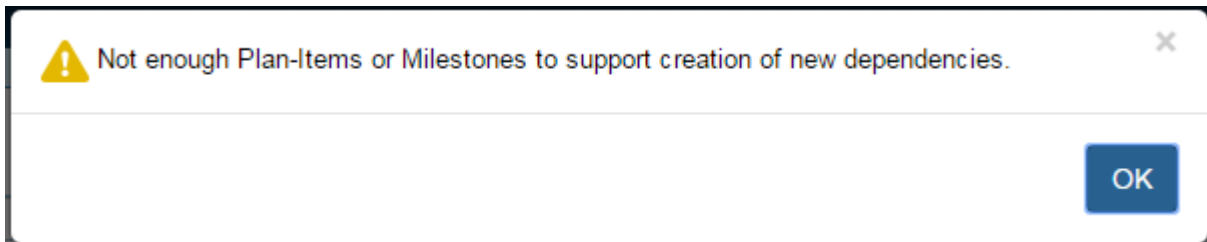
Milestone		Dependencies								
<div><div><div><div></div><div>Add</div></div><div><div></div><div>Remove</div></div></div></div>										
Name	Action	Type	Ready	Ready Time	Status	Status Changed	Time Delta	Plan	Plan Item	
<div><div><div><div></div><div></div></div><div>0AD31621-5A92-4770-937C-8F915B97C163</div></div></div>	<div><div><div></div><div></div></div><div></div></div>	START_ON	false							

If you want to create multiple point dependencies from the selected milestone in the tree then you must create multiple rows in the dependency table. You can only create a single dependency (point or time) per row created in dependency table. For example, if you created three point dependencies from a milestone, then you have to create a row in dependency table per point dependency, which would look like the dependency table shown in the following screenshot:

Milestone		Dependencies								
<div><div><div><div></div><div>Add</div></div><div><div></div><div>Remove</div></div></div></div>										
Name	Action	Type	Ready	Ready Time	Status	Status Changed	Time Delta	Plan	Plan Item	
<div><div><div></div><div>D4C9DED8-A9C6-488C-AS4D-638BAF44E19</div></div></div>	<div><div><div></div><div></div></div><div><div></div><div></div></div></div>		false							
<div><div><div></div><div>90B24D8-8114-43B9-B491-D26CCAB38BB4</div></div></div>	<div><div><div></div><div></div></div><div><div></div><div></div></div></div>		false							
<div><div><div></div><div>6CF0396F-2D99-49F8-AB50-049F9F2E5243</div></div></div>	<div><div><div></div><div></div></div><div><div></div><div></div></div></div>		false							

You can add Time-Dependency only to START milestone and cannot add multiple Time-Dependencies on the same milestone.

If you are to create more than one dependency on the selected milestone and you don't have enough milestones on which you can create dependency, then you get the following warning message:



Deleting Dependencies

Any dependency can be deleted by selecting the checkbox of the row and clicking **Remove**.

Milestone	Dependencies						
<div><div><div><div><div></div><div>+</div></div><div>Add</div></div><div><div><div></div><div>✖</div></div><div>Remove</div></div></div></div>							
Name	Action	Type	Ready	Ready Time	Status	Status Changed	Time Delta
<input checked="" type="checkbox"/> D4C90EDB-A9C6-488C-A54D-638BAFC44E19			false				
<input type="checkbox"/> 990B24D8-8114-43B9-B491-D26CCAB38BB4			false				
<input type="checkbox"/> 6CF0396F-2D99-49F8-AB50-049F9F2E5243			false				

Validations

There are different client side validations at different levels of user modifications. In case of any validation failure an icon is shown at plan item or milestone level, along with proper error message on hover of the icon.

The following are the validations for plan item:

1. Products, which were ordered in the order must always be part of the modified plan. For example, if you had ordered BPO_DUO in order line 1, then during modification the plan must always contain a plan-item with product BPO_DUO, here BPO_DUO can be associate to any available order line.
2. Order lines, which were part of the original order must also be part of the modified plan. For example, if there were 2 order lines in the order and as part of the modified plan, plan-items must be associated to both the order lines.
3. Product Id cannot be not null, each plan item must be associated to one product.
4. Action Value cannot be null, each product of the plan item must be associated to an action.
5. When you create a new user-defined field in **Custom Header** tab, name and value must not be null.
6. At least one order line must be associate with each plan item.

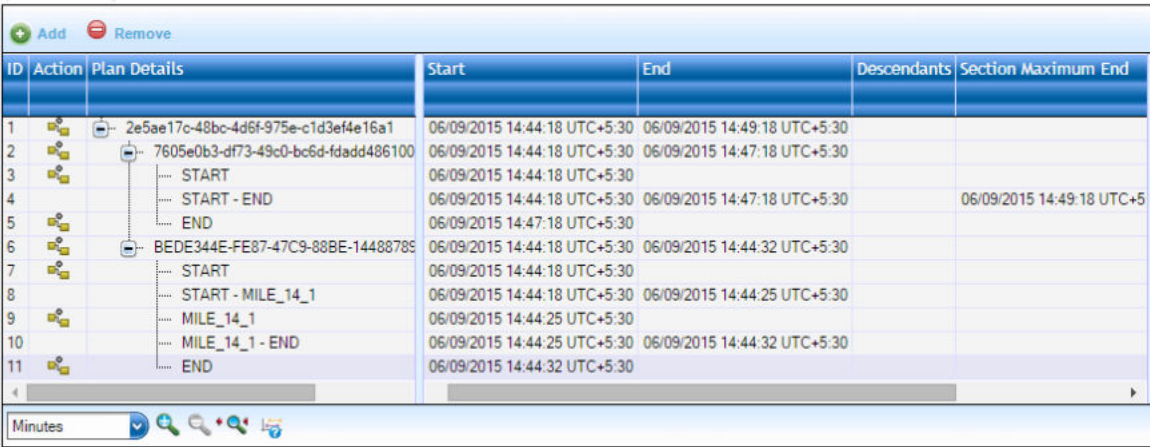
Following are validations for milestone:

1. Milestone id cannot be null. Each milestone when created or modified must be associated to a milestone id.
2. Dependency row when created must not be empty. Each dependency row when created must have a point or time dependency associated to it.
3. Proper section must exist in the plan fragment model when creating a new milestone.
4. The intermediate milestone must have at least one dependency on it or from it.

Modifying the Plan in Draft Mode through Gantt View

You can modify a Manual Order Plan Development plan from Gantt view as well. Toggle to Gantt view by


clicking the icon in the tool bar for the plan. Click **Draft Plan** in Gantt view and you are get **Add and Remove** options on the top bar of the Gantt chart.




ID	Action	Plan Details	Start	End	Descendants	Section Maximum End
1		2e5ae17c-48bc-4d6f-975e-c1d3ef4e16a1	06/09/2015 14:44:18 UTC+5:30	06/09/2015 14:49:18 UTC+5:30		
2		7605e0b3-df73-49c0-bc6d-fdadd486100	06/09/2015 14:44:18 UTC+5:30	06/09/2015 14:47:18 UTC+5:30		
3		START	06/09/2015 14:44:18 UTC+5:30			
4		START - END	06/09/2015 14:44:18 UTC+5:30	06/09/2015 14:47:18 UTC+5:30		06/09/2015 14:49:18 UTC+5:30
5		END	06/09/2015 14:47:18 UTC+5:30			
6		BEDE344E-FE87-47C9-88BE-14488789	06/09/2015 14:44:18 UTC+5:30	06/09/2015 14:44:32 UTC+5:30		
7		START	06/09/2015 14:44:18 UTC+5:30			
8		START - MILE_14_1	06/09/2015 14:44:18 UTC+5:30	06/09/2015 14:44:25 UTC+5:30		
9		MILE_14_1	06/09/2015 14:44:25 UTC+5:30			
10		MILE_14_1 - END	06/09/2015 14:44:25 UTC+5:30	06/09/2015 14:44:32 UTC+5:30		
11		END	06/09/2015 14:44:32 UTC+5:30			

You can perform all the modification options described in section using the Gantt chart option. We have tried to keep the Manual Order Plan Development modifications similar in Grid and Gantt view. There are some differences in how to access the edit widgets in Gantt.


Create a New Plan Item

Click **planID** to select it in the **Plan Details** column and click the Add icon  on the top toolbar. You can see a popup for filling the details of the new plan item.


Create a New Milestone

Click the plan item to select the plan item in the **Plan Details** column and then click Add icon  on the top toolbar. You can see the popup for filling the new milestone details.


Delete Plan Item

Select a plan item in **Plan Details** column and click the Remove icon  on the top toolbar and the selected plan item would be deleted. Deleting a plan-item deletes all the associated milestones and the dependencies on, or from the milestones.


Delete Milestone

Select a milestone in Plan Details column and click the Remove icon  on the top toolbar. The selected milestone would be deleted. You cannot delete START or END milestone. You can select any intermediate milestone and delete it. Deleting any intermediate milestone deletes all the dependencies on or from the milestone.

Modify Plan Item

When the plan is in draft mode, you can click the  icon in the **Actions** column corresponding to the plan-item you want to modify. It opens the popup with the plan item details.


Modify Milestones

When the plan is in draft mode, you can click the  icon in **Actions** column corresponding to the milestone you want to modify. It opens the popup with the milestone details.

Creating New Dependencies


Since the dependency is created on the milestone, you can create a dependency (point or time) when creating a new milestone, or by modifying an existing milestone. Creating a dependency when creating a new milestone opens the milestone details popup. Retrieve the details and visit the **Dependencies** tab. Creating a dependency when modifying an existing milestone, copy the milestone details from the popup and then visit the **Dependencies** tab.

Validations

If there is a validation error then the Validation Failure icon  is shown in the **Status** column of the Gantt chart.


Saving the Modified Plan

You can save your plan modification for future editing purposes. Click Save Manual Plan icon  on the

plan toolbar  to save the modified plan's copy for later use. In case there are some client side validations then the plan won't be saved and upon clicking the save icon you can see an appropriate message. If the plan is saved successfully then you can see the success message on the user interface.

Saving and Executing the Modified Plan

After completing the plan modifications, you have to start the execution of the plan. Click Save and Execute


Manual Plan  icon to save the final copy of modified plan and start execution of the plan. Final copy of the plan is saved only when there are no validation errors at the UI side in the plan. The plan is sent for execution only when server side validations are passed. See *TIBCO Order Management Long Running Concepts and Architecture Guide* for details on 'Server Side Validations'.

The plan being modified must always be the latest copy to either Save or Save and Execute.

For example, if user1 is in process of modifying the plan and user2 takes the lock, modifies and saves it, the user1 has a stale copy of the plan. Hence, when user1 tries to save the plan, a message is displayed: `New copy of plan available in database, please update. Cannot Save the plan.` The changes are not saved. user1 has to reload the plan and make the modifications again.

Discarding Changes

To discard the modifications made on the plan, click the Discard Changes  icon on the plan toolbar

. The changes made by you are discarded. You are presented with a confirmation box. Click **OK** and your changes are disappeared and you are presented with last saved copy of the plan. Discarding changes is possible only when the plan is in OPD state in the user interface.

Modifying Plan in EXECUTION State

In case you have to modify the plan, which is already in EXECUTION state, you must submit an order amendment with Manual Order Plan Development user-defined field as per the configurations. Order Amendment can be submitted either through SOAP Service or through User Interface.

Once the order amendment is submitted with appropriate Manual Order Plan Development user-defined field, the order remains in SUSPENDED state and the corresponding plan in SUSPENDED state is available for manual modifications.

Steps for modifying the plan in EXECUTION state are as follows:

1. Suspend the order you want to modify.
2. Initiate an order amendment with Manual Order Plan Development user-defined field and also with the changes you want to make to the order (if any).
3. Order is in SUSPENDED state and is ready for plan modifications in the UI.
4. Search for the Manual Order Plan Development plan as described in the topic [Searching Orders for Manual Order Plan Development](#) and perform the plan modifications.
5. After completion of the modifications, click **Save & Execute Manual Plan** to resume the plan's execution after modifying the amended plan.

Jeopardy Management System

This section describes the functions of TIBCO® Order Management Jeopardy Management System feature.

Jeopardy Management System

Service level agreements (SLAs) are commonly used to ensure the Quality of Service (QoS). The conventional way to manage a service is to measure the Quality of Service and then determine whether the requirements have been met. This means that problems are detected and then corrective action is taken. By contrast, the Jeopardy Management module relies on predicting the result of Quality of Service compliance of process components that are part of order fulfillment ecosystem. Therefore, it is frequently possible to take corrective action before a problem occurs, thus minimizing its impact.

Jeopardy Management

The jeopardy management process involves three main activities:

1. Monitoring the Quality of Service
2. Reporting the Quality of Service
3. Predicting the Quality of Service

The objectives of Jeopardy Management are as follows:

1. Continuous collection of performance data and status information of all execution plan
2. Detect SLA violation
3. Predict Jeopardy Conditions for execution plan
4. Perform consequential actions
 - a. Send notification
 - b. Invoke web service

The Jeopardy Management System enables you to manage the risk associated with plan tasks falling behind schedule, and to prevent them from jeopardizing the timely fulfillment of an order. The Jeopardy Management System is a key component of Order Management. Jeopardy management is the process of monitoring execution of a set of tasks in a plan to fulfill a customer order. In this application, execution plans are generated by decomposing orders based on the product model. Plans are orchestrated based on a schedule, and when a plan goes or predicted to go outside the expected design of the schedule then the system notifies the stakeholders as early as possible to take the corrective steps.

A plan is composed of a series of plan items. Each plan item has at least two milestones:

- START milestone
- END milestone

Plan items might also have intermediate milestones that represent points of interest during execution of that plan item. Service-level agreement of service provider that executes plan items are specified through process component model or Plan fragment model, which stipulate among other things - the provided Services performance. SLAs have a typical duration and a maximum allowed duration to fulfill a plan item.

To manage the risk associated with plan tasks falling behind schedule, the jeopardy manager performs the following risk-management tasks:

1. Managing Critical Paths: Jeopardy management computes and keeps an account of critical paths through an execution plan. Two of these critical paths correspond to the typical and maximum durations of process components. The third type of critical path is based on the actual duration to date, once the execution plan has started processing. The critical paths are used to predict the completion date

and time of the execution plan. By viewing the critical paths in the UI GANTT Chart, you can determine whether your execution plan is progressing normally or if it is in danger of being in jeopardy. The Order Management Server UI highlights each plan item as per the following legend scheme:

- a. Plan item instance is marked in the NORMAL region, if its execution is started on time and completed within expected time lines.
- b. Plan item instance is marked in the HAZARD region, if its execution is started late and delayed than expected time lines but it does not lie on critical path.
- c. Plan item instance is marked in the CRITICAL region, if its execution is started late and delayed than expected time lines and lies on the critical path. The UI also highlights risk regions of a plan with color scheme.

Jeopardy Indications



2. Monitoring jeopardy conditions at each of the following levels:
 - a. Plan Task
 - b. Execution Plan
3. Perform consequential actions at each of the following levels:
 - a. Plan Task
 - b. Execution Plan
 - c. Milestone

The Order Management UI shows dashboard for jeopardy management containing the following elements along with Orders Summary, Amended Orders, Backlog Orders, Orders in Execution:

1. Orders In Jeopardy.
2. Jeopardy Live Alerts.
3. Jeopardy Recorded Alerts.

Understanding Plan

A plan is constituted of a series of plan items. Each plan item has at least two milestones - START and END. Plan items might also have intermediate milestones that represent points of interest during execution of that plan item.

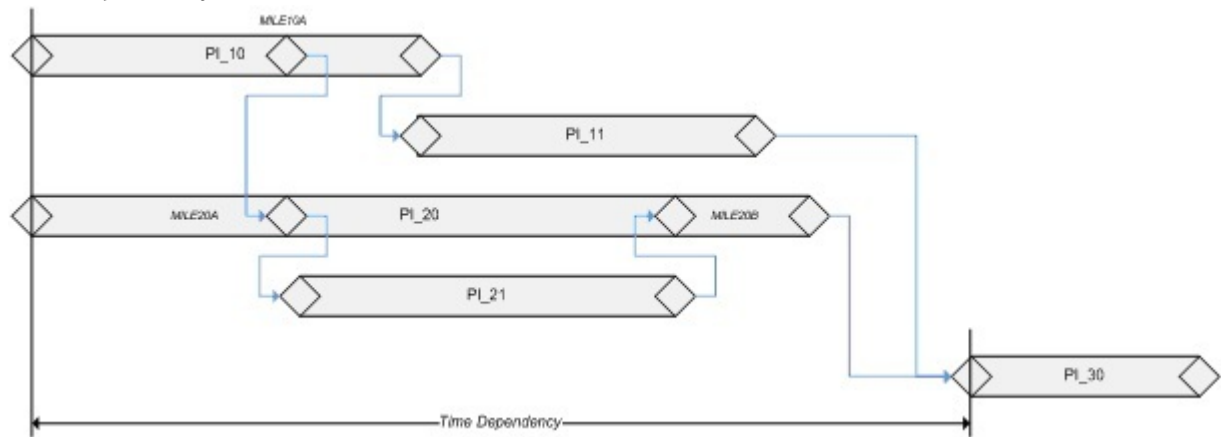
There are two types of milestone dependencies:

Point Dependency	Time Dependency
dependency on release of a given milestone in another plan item in the plan	dependency on a given date and time being exceeded



Execution of a plan item stops at a milestone until that milestone has been released. A milestone with no dependencies is released immediately. However, a milestone with attached dependencies is only released once all the point and time dependencies are satisfied.

Plan Dependency



For details on dependencies, see [Understanding Dependencies](#).

Understanding Critical Path

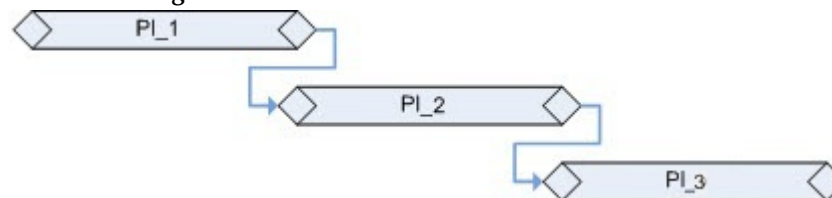
. The critical path is the longest sequence of plan item sections that determines end time of a plan. The critical paths are used to project the completion date and time of the execution plan.

Paths are computed using the dependencies between plan item sections.

Critical Path Calculation

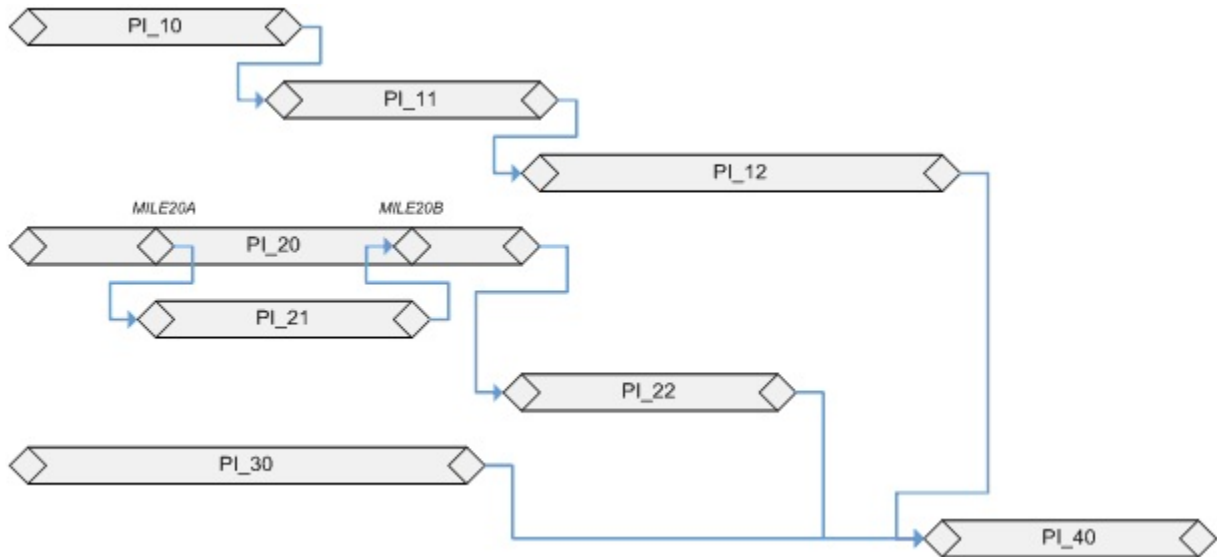
Critical path is used for both SLA and predictive jeopardy for monitoring a plan. A simple plan consists of a single execution path. For example:

Plan with Single Execution Path



Some plans have multiple execution paths as shown in the following figure:

Plan with Multiple Execution Paths



Understanding Dependencies

Dependency can be defined as a relationship between milestones in an execution plan. For example, Milestone B cannot start until Milestone A completes.

Milestone Dependencies

When a dependency on a milestone is determined, you can either depend on the milestone being started, or finished.

This means that Milestone 2 cannot be finished until Milestone 1 is started.

End Milestones

In case of an end milestone:

- another milestone cannot depend on the finish of an end milestone because there is no Finish (Release) on a Task Complete message
- the end milestone cannot be dependent on any other milestones

The following table lists the types of dependencies that can be set up between plan task milestones.

Dependency	Effect
Must Start On	The milestone must start on the date/time specified. If it cannot start for some reason (for example, because a previous plan task is late), a jeopardy condition is triggered.
Finish to Finish	One milestone is able to be released when the other milestone is released
Start to Finish One	One milestone can be released only when the other begins

Jeopardy Management for Execution Plans

If a given plan task or milestone is in jeopardy, it might or might not indicate that the overall execution plan is in jeopardy. The jeopardy manager also provides facilities for monitoring whether the whole execution plan is running on time, or taking longer to complete than expected. This is done by monitoring the forecast end date and time of the plan and comparing it against several threshold dates.

If you use start date scheduling or end date scheduling for your execution plans, you can set a different set of jeopardy conditions at plan level.

Jeopardy Management for Plan Task

A simple process component that has only start and end milestones consists only of one section, but more complex components are made up of several sections. A section is the interval between two milestones.

At the level of process component sections, you can configure jeopardy conditions that enable you to detect both if the task has taken longer to complete than it must have, and also to detect if a task that is under way or has not yet started is predicted to take longer to complete than scheduled.

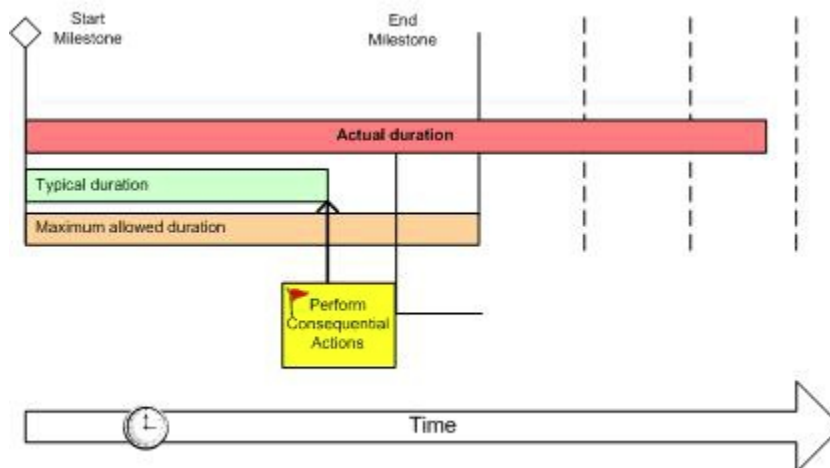
Jeopardy manager allows you to monitor the following durations:

- **Typical Duration:** you can specify this value when you create a process component, or when you define the plan task that uses the component in an execution plan.
- **Maximum Allowed Duration:** the maximum amount of time the activity represented by the task can take before it is considered to have overrun. You can specify this value when you create a process component, or when you define the plan task that uses the component in an execution plan.

There are critical paths identified in execution plans, constructed using the typical and maximum durations of the plan tasks included in those plans. If one of the process component sections being monitored has not completed before its defined typical duration, a monitor event is triggered (consequential action is performed). If the section has not completed before the end of its maximum allowed duration, another monitor event is triggered. For Example, a plan has taken longer than expected/predicted. The plan task has exceeded its typical duration, its maximum duration, and two subsequent monitoring intervals. A monitor event has been fired at each stage to notify you of the following:

- After Typical Duration
- After Maximum Duration

Typical and Maximum Durations



Jeopardy Risk Region for Plan

NORMAL	• Plan running within typical duration
Hazard	• Plan running exceeding typical duration
Critical	• Plan running exceeding maximum duration
Out of Scope	• Plan running beyond last acceptable completion date

Must Start On Dependencies

The must start on dependency indicates that an activity must start at a specific point in time. You can apply these dependencies to milestones that denote the start of such activities; in normal circumstances, when the execution plan is running on schedule, these are used to schedule activities at the right time, by releasing the relevant milestones. However, if it is predicted that it is not possible to release a milestone at the scheduled time, or if that time is reached and the milestone still cannot be released, the Jeopardy Management System recognizes the jeopardy condition.

Predictions are calculated during jeopardy detection cycle. Frequency of Jeopardy detection cycle is configurable.

Consequential Actions

Jeopardy Manager raises the jeopardy event. If a plan item is in jeopardy, the `PlanItemJeopardy` event is raised. If a Plan is in jeopardy, the `PlanJeopardy` event is raised.

To reduce the number of Jeopardy notifications sent out for a particular plan, jeopardy sends either a predictive notification or the actual notification at the plan level. For example, if the jeopardy sends out a predictive notification `AFF-JM-PLAN-0200` for the Plan to possibly exceed the typical duration, then the jeopardy does not send out the `AFF-JM-PLAN-0100` notification if the plan actually exceeds typical duration, as they are the notifications for the same risk region.

Jeopardy event message contains payload with information about the jeopardy condition. You can configure the consequential that you want to perform when these events are raised by the system that configures the Jeopardy Rules.

Jeopardy Rules can be configured through Order Management Server UI rule configuration option.

For each of the listed jeopardy conditions, you can take any of the following possible consequential actions:

- Alert notification.
- Fulfillment Action.

Jeopardy Pre-release Order Processing

The earlier versions of TIBCO Order Management (version 2.1.0 and earlier) used to store the order and the plan information for jeopardy in the active space data store. Now in the application, jeopardy has been designed to work in cache (server memory) mode. In the cache mode, jeopardy stores order and plan information in server cache (memory).

Because the implementation information of the earlier versions (version 2.1.0 and earlier) are not be available in the server cache, jeopardy ignores orders that were placed prior to the TIBCO Order Management 2.1.1 implementation, and are still in the execution status. Hence, it is recommended that all

existing orders, placed prior to the TIBCO Order Management 2.1.1 implementation, must be in their logical end status, which is COMPLETED, CANCELLED, or WITHDRAWN.

Predictive Jeopardy

Predictive jeopardy is measured on several metrics:

Plan Item Start Date	Plan Duration
For plan item milestones with both point and time dependencies, it is possible that the specified time dependency is not feasible based on the durations of the previously executed plan items that form the point dependencies on the same milestone	The overall duration of the plan can be calculated by performing a critical path analysis on the plan items that compose the plan
Plan item start date predictive jeopardy determines whether the plan item is later than the specified start date due to the other dependencies	Plan duration predictive jeopardy determines whether the execution duration of the plan exceeds the design duration of the plan

Jeopardy Events

The following are the types of jeopardy events:

Plan Item Jeopardy

The following table lists the jeopardy conditions for the plan item:

Plan Item Jeopardy Conditions	Description
AFF-JM-PLANITEM-0100	Plan item has exceeded typical duration
AFF-JM-PLANITEM-0110	Plan item has exceeded maximum duration
AFF-JM-PLANITEM-0120	Plan item has exceeded required start
AFF-JM-PLANITEM-0200	Plan item start is predicted to exceed required start and is increasing
AFF-JM-PLANITEM-0210	Plan item start is predicted to exceed required start and is decreasing
AFF-JM-PLANITEM-0220	Plan item is no longer predicted to exceed required start

Plan Jeopardy

The following table lists the jeopardy conditions for plan:

Plan Jeopardy Conditions	Description
AFF-JM-PLAN-0100	Plan has exceeded typical duration
AFF-JM-PLAN-0110	Plan has exceeded maximum duration
AFF-JM-PLAN-0120	Plan has exceeded out of scope threshold

Plan Jeopardy Conditions	Description
AFF-JM-PLAN-0200	Plan is predicted to exceed typical duration and is increasing
AFF-JM-PLAN-0210	Plan is predicted to exceed typical duration and is decreasing
AFF-JM-PLAN-0220	Plan is no longer predicted to exceed typical duration
AFF-JM-PLAN-0230	Plan is predicted to exceed maximum duration and is increasing
AFF-JM-PLAN-0240	Plan is predicted to exceed maximum duration and is decreasing
AFF-JM-PLAN-0250	Plan is no longer predicted to exceed maximum duration

Order Selection for Jeopardy Management

Since Jeopardy Management System is designed to send an alert on the plan tasks that are falling behind schedule, and to prevent the plan tasks from jeopardizing the SLA for the entire order fulfillment, JeOMS makes some dynamic decisions when processing long and short running orders.

There are chances of orders missing SLA requirements. If such a scenario occurs, then getting alerts on the short running orders (orders expected to finish in 5 minutes or less) is not helpful as sending alerts to production support personnel, and the subsequent manual action on the alerts, takes more time than the lifespan of the order.

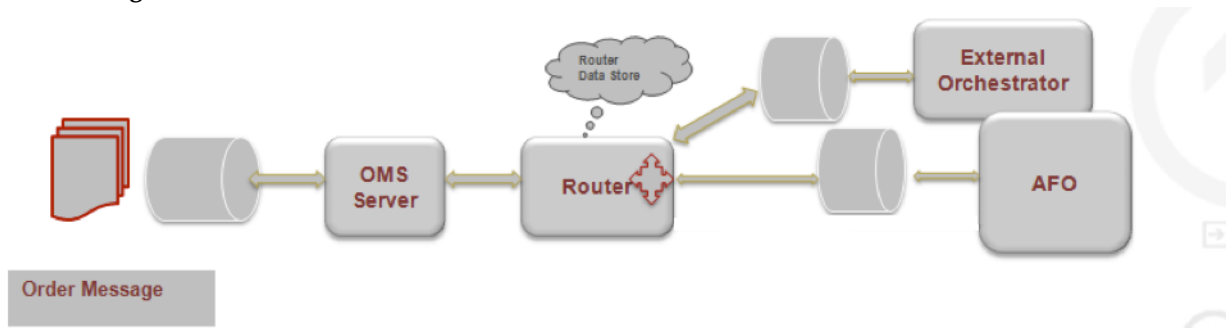
Jeopardy, therefore, has a stronger focus on the long running orders (orders that are expected to run for an hour or more). Jeopardy processes almost all the orders, but for short running orders, jeopardy might skip some alerts that are not expected to be handled when the risk level for that alert changes to a severe one.

Router

Here is the list of features of the Router:

1. Router redirects the order request to external Orchestrator based on routing parameter configured in the payload.
2. The routing parameter is extracted using XPATH expression.
3. Router invokes the Orchestrator services or using JMS.
4. Router also manages the table of order id of the order request and Orchestrator node that processed. Router makes sure that any subsequent request on the order is routed to same node. For example, there are two Order Management Server instances deployed in the cluster.
5. Router can optionally send only the order id in the payload properties (i.e. the order request is removed from the payload). The Orchestrator listener gets the payload with order Request from ActiveSpaces hibernate second level cache after consuming the order request.

Router Diagram



Internal Error Handler

Internal Error Handler marks the failed plan items in ERROR state and gives you the control to select appropriate action for the failed plan item.

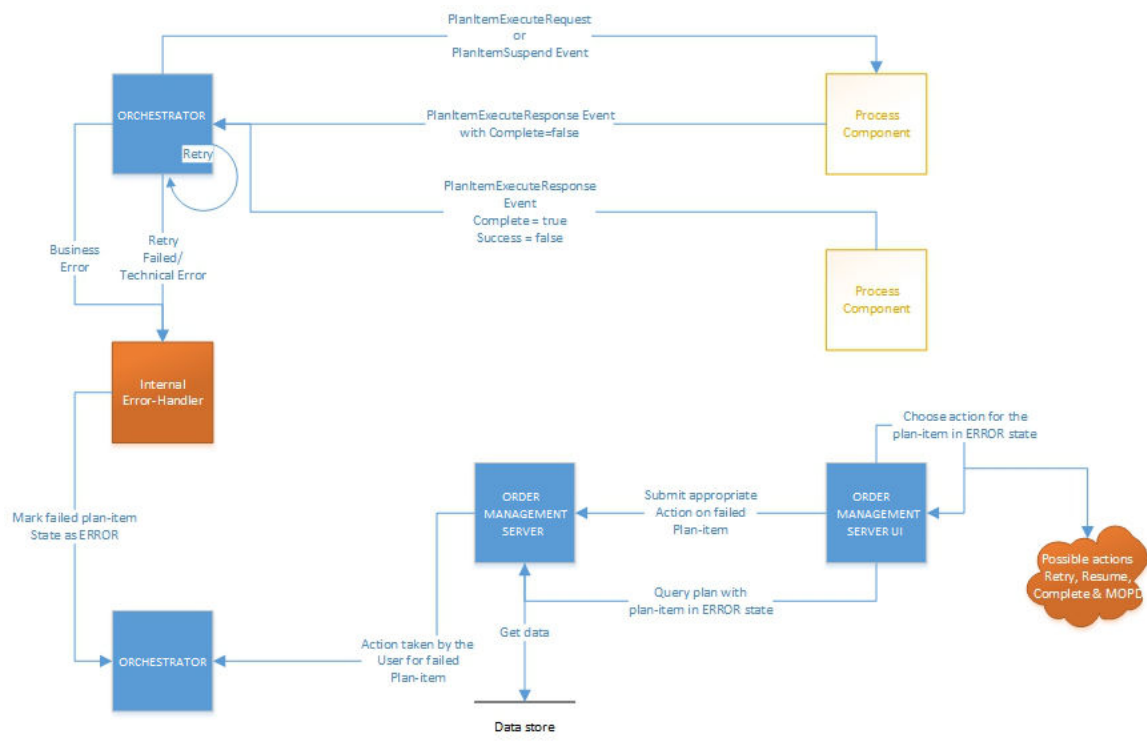
Internal Error Handler is an optional component and you can choose to configure internal error handler. By default, the external error handler is configured.

Internal Error Handler is designed to handle the failed plan-items. It handles the failed plan-items in two step process:

1. Mark the plan item state as ERROR for the plan item that failed.
2. Choose an appropriate action for the plan item in ERROR state from Order Management Server UI. You are able to choose appropriate action from the following options:
 - a. Retry
 - b. Resume
 - c. Complete
 - d. Manual Order Plan Development

Internal Error Handler Data Flow Diagram

The following is the data flow diagram for Internal Error Handler:



Understanding Data Flow in Internal Error Handler

The following steps help you understand the data flow in the Internal Error Handler:

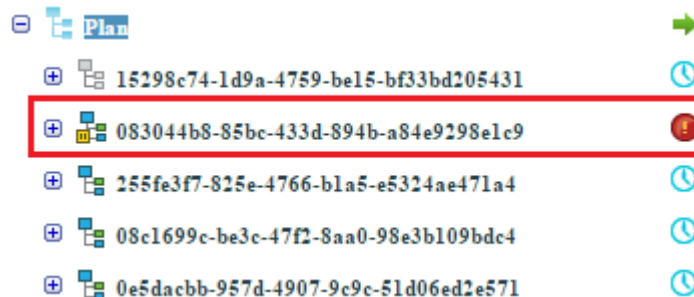
1. Orchestrator sends the PlanItemExecuteRequest or PlanItemSuspend event to the process component for each plan item.


2. In response, the process component sends PlanItemExecuteResponse event.
3. In PlanItemExecuteResponse event we have two flags: Completed and Success, and based on value of these flags the orchestrator takes appropriate action. The following tables illustrates the same:

	Complete	Success	Description
Technical Error	False	False/True	Orchestrator retries the process component call for the defined number of retries with the defined retry interval. If the process component call continues to fail, then it refers the plan item to the Plan Item Failed Handler.
Business Error	True	False	Orchestrator refers the plan item to the Plan Item Failed Handler.
Success	True	True	Processing continues as normal.

Steps 1 through 3 are part of existing implementation

4. The orchestrator invokes the plan item Error Handler according to your configuration. Going forward the system considers that you have configured Internal Error Handler, and refer the Internal Error Handler as Error Handler.
5. Plan item Error Handler changes the failed plan item state to ERROR.
6. Plan remains in EXECUTION with one or more plan items in ERROR state.
7. You are able to search for the plans with one or more plan items in ERROR state in Order Management Server UI.
8. After searching for the plan with plan item in ERROR state, you can view the plan details.
9. The following image shows the plan item in Error state:



The plan item have a tree icon with a pause on it, which indicates that the plan item is paused and needs manual intervention. The status icon is shown as  which indicates that the plan item is in ERROR state.

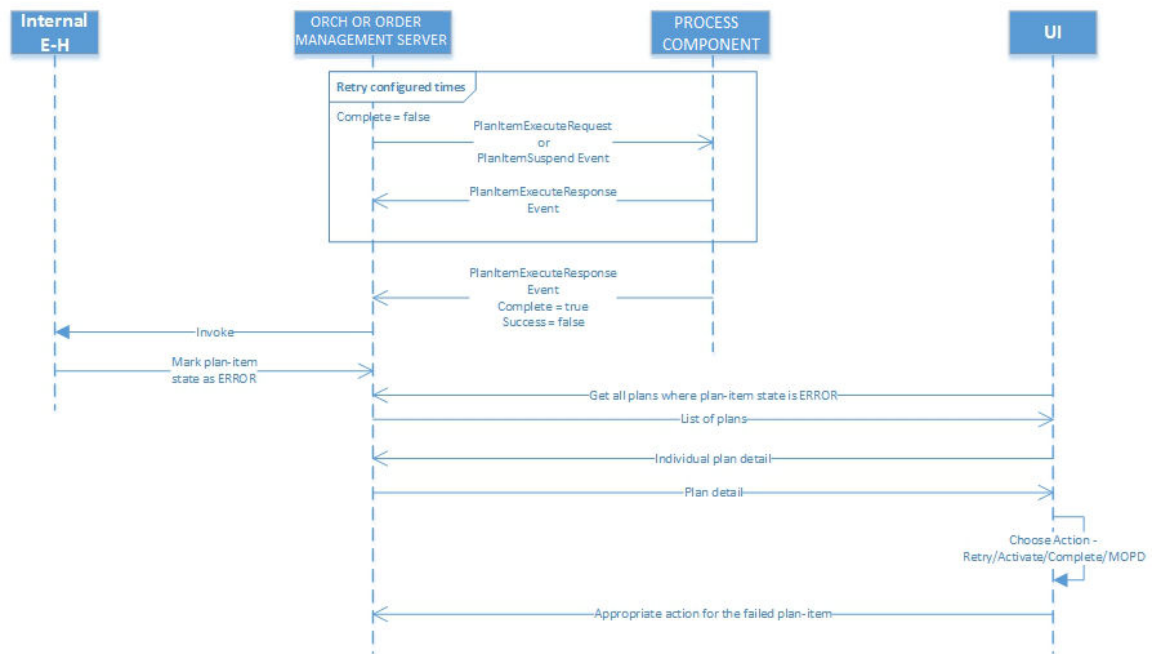
10. You can access the plan item in ERROR state and take appropriate action on it. Action on the plan item in ERROR state can be from one of the following options:
 - a. Retry
 - b. Resume
 - c. Complete
 - d. Manual Order Plan Development
11. You can take different actions on different failed plan items in the same plan. This would vary only if you have chosen Manual Order Plan Development as your action for the plan item in ERROR, since

Manual Order Plan Development as the error resolution would act on the entire plan rather than the individual plan item.

12. You have to submit the action taken for each failed plan item.
13. After your submission, the orchestrator would initiate the action on the respective plan items.

Internal Error Handler Sequence Diagram

The following is the sequence diagram to show the actual sequence of the flow of data in Internal Error Handler:



Searching for Plans with Plan Item in ERROR State

On the **Plan** tab you can search for the plans with any of its plan item in ERROR state. A new attribute has been added named **Plan Item Status** with all the possible values of the plan item.

Filter

Condition

([Plan-Item Status] [=] 'ERROR')

Attributes

- Order ID
- Order Ref ID
- Plan ID
- Description
- Status
- Originator
- Created on
- Changed on
- Plan-Item Status

Operator

- =
- like
- in

Value

- EXECUTION
- PENDING
- SUSPENDED
- ERROR_HANDLER
- ERROR

Add Condition Reset

Save Reset Condition Cancel

Modifying the Plan Item State

After you have searched the plans with plan item or plan items in ERROR state, you have to select the desired plan and select the plan item in ERROR state.

In Grid view you can choose the appropriate action on the plan item and submit the chosen action. When a plan item in ERROR state, it is shown with an appropriate icon, the icon represents the plan item in ERROR state and needs user intervention.

Plan	Plan Item	Status
15298c74-1d9a-4759-be15-bf33bd205431		
083044b8-85bc-433d-894b-a84e9298e1c9		ERROR
255fe3f7-825e-4766-b1a5-e5324ae471a4		
08c1699c-be3c-47f2-8aa0-98e3b109bdc4		
0e5dacbb-957d-4907-9c9c-51d06ed2e571		

Choosing Error Resolution for the Plan Item in Error State

After clicking the plan item you get the details for the plan item. This plan item have Status as ERROR and have option so that you can choose the appropriate Error Resolution and Comments for the specific plan item.

Plan Item	Custom Headers	Order Line	Process Info	Sections
Plan Item ID	083044b8-85bc-433d-894b-a84e9298e1c9			Description
Status	ERROR			A1 PROVIDE
Start TimeStamp	11/06/2014 18:17:00 UTC+5:30			Changed Date
Action	PROVIDE			End TimeStamp
Error Resolution				Non Executing
Comments				false

Error Resolution is a drop-down box that contains actions that you have to perform to fix the plan item in ERROR state. You have following choices for the plan item's Error Resolution:

- Retry

- Resume
- Complete
- Manual Order Plan Development

Plan Item	Custom Headers	Order Line	Process Info	Sections
Plan Item ID Status Start TimeStamp Action Error Resolution Comments	083044b8-85bc-433d-894b-a84e9298e1c9 ERROR 11/06/2014 18:17:00 UTC+5:30 PROVIDE <div> <div></div> <div>Retry</div> <div>Resume</div> <div>Complete</div> <div>MOPD</div> </div>		Description Changed Date End TimeStamp Non Executing	A1 PROVIDE 11/06/2014 18:17:08 UTC+5:30 false

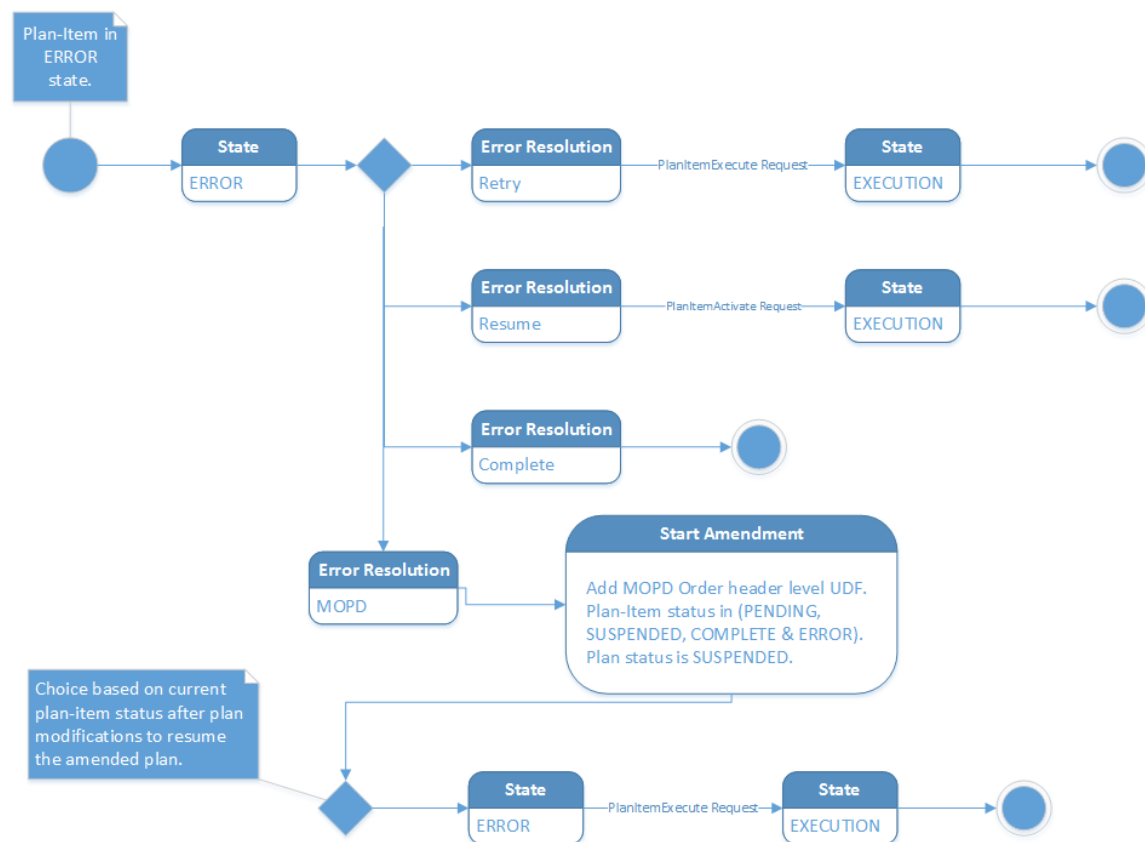
There can be more than one plan item in ERROR state and you have to choose error resolution for each plan item in ERROR state. In case you don't choose the error resolution for some plan items and submit the choice to the server, then the error resolution is executed as per your choice for those few plan items and the rest of the plan items remain in ERROR state.

Details of Each Resolution Choice

You can choose any appropriate action from the error resolution drop down box and the error resolution chosen is considered as a resolution choice for that particular plan item.

This changes only when you have chosen Manual Order Plan Development as the Error Resolution. When you choose Manual Order Plan Development as Error Resolution this resolution choice would be applicable to the entire plan and not just the plan item.

The following diagram shows the State Machine diagram for different states of the plan item after user chooses a resolution type for the plan item in ERROR state:



Error Resolution - RETRY

When you submit the error resolution with RETRY as the appropriate action, a new Plan Item Execute Request is sent for the plan item and the orchestrator moves the plan item state from ERROR to EXECUTION.

Error Resolution - RESUME

When you submit the Error Resolution with RESUME as the appropriate action, a new Plan Item Activate Request is sent for the plan item and the orchestrator moves the plan item state from ERROR to EXECUTION.

Error Resolution - COMPLETE

When you submit the error resolution with COMPLETE as the appropriate action, the plan item is marked as COMPLETE by the orchestrator. The orchestrator moves the plan item state from ERROR to COMPLETE.

Error Resolution - Manual Order Plan Development

You can select Manual Order Plan Development as the appropriate Error Resolution. Manual Order Plan Development action is treated at plan levels, which means if you choose Manual Order Plan Development, then the plan is considered for Manual modification.

And to consider the plan for Manual Order Plan Development we have to start by adding an Order-Header level user-defined field with Manual Order Plan Development flag. This would be done in the background without any user intervention as soon as you submit Manual Order Plan Development as the Error Resolution.

The following steps are performed after you choose to submit Manual Order Plan Development as the error resolution:

1. Order Header user-defined field is updated with Manual Order Plan Development flag.
2. Order amendment, with only user-defined field modification, is submitted.
3. Plan is moved to SUSPENDED state with plan items in (SUSPENDED, COMPLETE, PENDING, or ERROR) state.
4. Plan is presented to the user for modifications.
5. After user modifications are complete, user sends the modified plan for further execution.
6. Plan item in ERROR state is moved to EXECUTION and PlanItemExecute or PlanItemActivate Request is sent for this plan item depending upon the type of request that was sent for this plan item before it failed. All the other state transitions remain intact.

Submit the Error Resolution

After taking the error resolution on the plan item in ERROR state, you can submit your changes. With error resolution choice as Retry, Resume, or Complete, you can submit these error resolutions for the plan items in ERROR state all in one go, or you can submit each error resolutions for the plan item in ERROR state separately.

When you choose error resolution as Manual Order Plan Development for any of the plan item, it would impact the entire plan and not the individual plan item in ERROR state. So once you choose the error resolution as Manual Order Plan Development for any of the plan item in ERROR state, any other error resolution selection for other plan item in ERROR state have no impact. If there were three plan items in ERROR state and for one plan item you have chosen error resolution as Retry, and for another plan item, which is in ERROR state, you have chosen as Resume, and for the third plan item you chose Manual Order Plan Development as error resolution, then the resolution type for first two plan items have no impact and the Manual Order Plan Development process is initiated.

Plan ID: 107e2b81-937e-4223-bd8c-a4d518adecf9 (Plan Item: 083044b8-85bc-433d-894b-a84e9298e1c9)

Plan

16298c74-1d9a-4789-ba1f-bd33bd20f431

083044b8-85bc-433d-894b-a84e9298e1c9

Milestones

2666a3f7-82fa-4766-b1af-af324aa471a4

08c1699c-ba3c-47f2-8aa0-98a3b1b09bdc4

8a7daccb-957d-4907-9c9c-61d06e22a671

Plan Item

Custom Headers

Order Line

Process Info

Sections

Plan Item ID

Status

Start Time Stamp

Action

Error Resolution

Comments

083044b8-85bc-433d-894b-a84e9298e1c9

ERROR

11/06/2014 18:17:00 UTC+5:30

PROVIDE

Description

Changed Date

End Time Stamp

Non Executing

A1 PROVIDE

11/06/2014 18:17:08 UTC+5:30

false

If you have plan item or plan items in ERROR state, you have an option to submit the Error Resolution.

State Machine Pagination

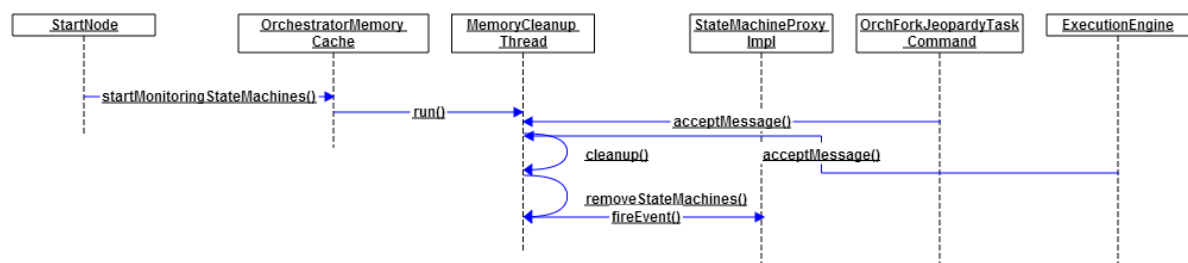
A state machine is initiated for every order submitted. With large number of state machine in heap memory there is possibility that the application might go out of memory. To control the memory usage of state machine, it is necessary to evict the state machines from heap memory.

Eviction of state machine from heap memory happens as follows:

1. Eviction happens only if property "Max No of StateMachines in Heap Memory" is configured as a value greater than 0.
2. Eviction cycle can be triggered in following two ways:
 - a. A Memory Cleanup thread monitors the eviction process after every predefined interval configured as `com.tibco.fom.orch.intervalMonitoring`. Within this interval before the eviction process starts it is possible that number of state machines in memory exceeds the threshold configured depending upon the order injection rate and other processing of the orders.
 - b. When there is any event coming to state machine for processing it invokes the eviction cycle to run and waits until the number of state machine is less than or equal to the threshold configuration defined. Once the eviction cycle completes and state machine counts is below the threshold defined further requests is processed.
3. As part of every eviction cycle, the application tries to find out the least used object. The application identifies a timestamp that can be used as a threshold called the critical timestamp and all the state machines with timestamps smaller than this is eligible for eviction. Memory Cleanup threads started at time of initialization of orchestrator node runs at a predefined interval to identify the state machines, which are eligible for eviction looking at the timestamp of all the state machines. State machines, which are idle for more than the configured idle time out are selected for eviction till the count of state machines matches to the configured threshold number, which can live in memory. To identify state machine for eviction, first timestamp of all the state machines are sorted in ascending order and then ignores the same no of state machines configured as threshold count having highest timestamp value. Out of these timestamps, which have the highest timestamp and count remains within threshold is selected not to be evicted; the smallest timestamp is selected as critical timestamp. Any state machine having timestamp smaller than the critical time stamp are eligible for eviction. To evict a state machine from memory, an event is submitted to respective state machine or BatchProcessor to clear the state machine entry from memory.
4. For any subsequent request, if state machine is not available in memory then state machine information is re retrieved back from backing store using the data stored as state machine context checkpoint. Using this checkpoint data, all the required data is populated and a state machine entry is created in memory for further use.
5. A cleanup thread keeps running after some interval, which removes the state machine context checkpoint data along with resourceWhen order reaches to its final state then state machine entry is deleted from memory.

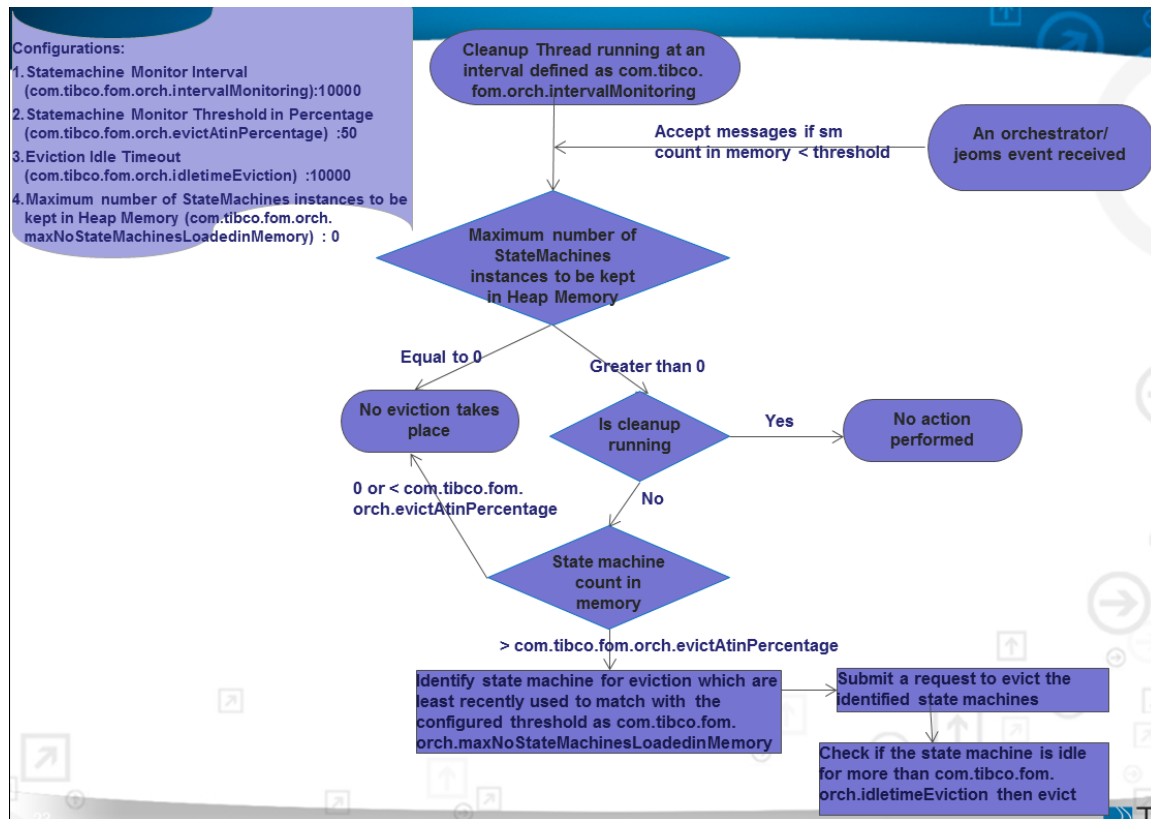
State Machine Pagination Sequence Diagram

The following diagram represents the State Machine Pagination Sequence Diagram:



State Machine Pagination Flow Diagram

The following diagram represents the State Machine Pagination Flow Diagram:



The following properties are not present in any of the config files in the \$OM_HOME/roles/configurator/standalone/config folder.

- com.tibco.fom.orch.intervalMonitoring
- com.tibco.fom.orch.evictInPercentage
- com.tibco.fom.orch.idleTimeEviction

These properties are present in the ConfigValues_OMS_INTERNAL.XML file, which can be found in \$OM_HOME/roles/omsServer/standalone/lib/configService-5.0.0-SNAPSHOT.jar. If you have to configure these values, the entries below can be copied into your ConfigValues_OMS.xml file.

```
<ConfValue description="Eviction Idle Timeout" isHotDeployable="true" name="Eviction
Idle Timeout" propName="com.tibco.fom.orch.idleTimeEviction" readonly="false"
sinceVersion="2.1" visibility="Basic">
<ConfNum default="10000" value="10000"/>
</ConfValue>
```

```
<ConfValue description="Statemachine Monitor Interval" isHotDeployable="true"
name="Statemachine Monitor Interval" propName="com.tibco.fom.orch.intervalMonitoring"
readonly="false" sinceVersion="2.1" visibility="Basic">
<ConfNum default="10000" value="10000"/>
</ConfValue>
```

```
<ConfValue description="Statemachine Monitor Threshold in Percentage"
isHotDeployable="true" name="Statemachine Monitor Threshold in Percentage"
propName="com.tibco.fom.orch.evictAtinPercentage" readonly="false" sinceVersion="2.1"
visibility="Basic">
<ConfNum default="50" value="50"/>
</ConfValue>
```

Order Capture System Overview

Order Capture System (OCS) is a new component in TIBCO Fulfillment Suite to create, manage, and submit TIBCO Fulfillment Orchestration orders based on what a subscriber already has.

This component is a web application which you can use to do the following:

- Select subscribers
- Browse validated products, services, or bundles from TIBCO Fulfillment Catalog
- Create orders for selected subscribers and submit them to TIBCO Order Management

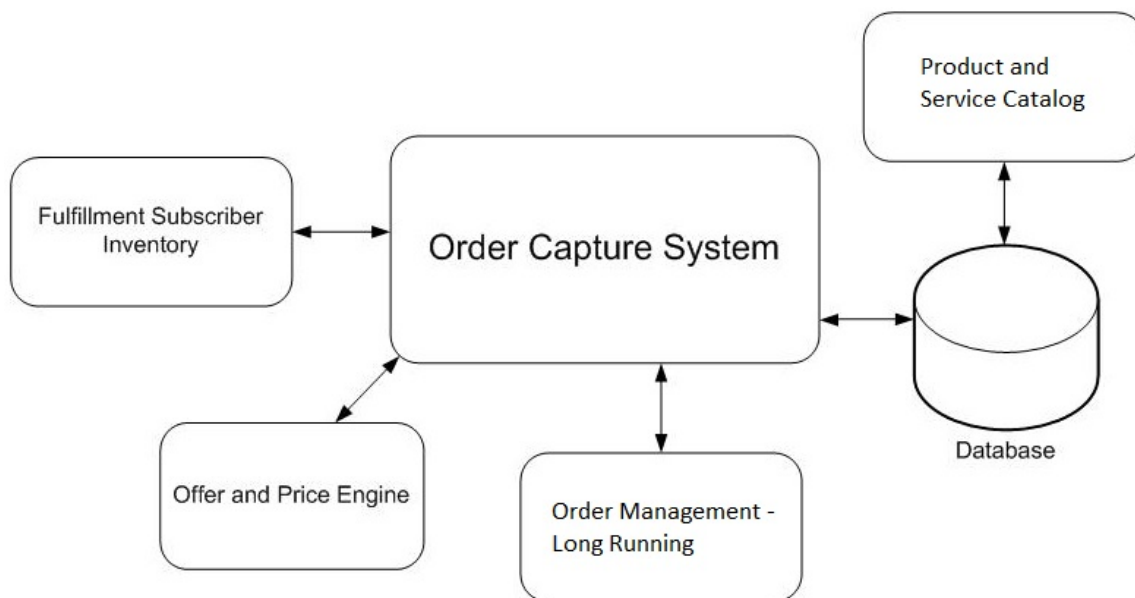
To construct an order, Order Capture System requests information from the following systems:

- TIBCO Fulfillment Catalog, which provides product definitions for subscribers
- A subscriber inventory, which provides subscriber details (such as names, addresses, IDs, and segments)
- TIBCO Fulfillment Subscriber Inventory, which provides information about the subscribers' provisioned products.
- Offer and price engine (OPE), which provides the eligible products for a specific subscriber, validates the order, and provides the price of what is provisioned.

Order Capture System then submits the orders to TIBCO Order Management.

The following figure illustrates the interconnection between Order Capture System and the Fulfillment Orchestration subsystems:

Order Capture System Architecture

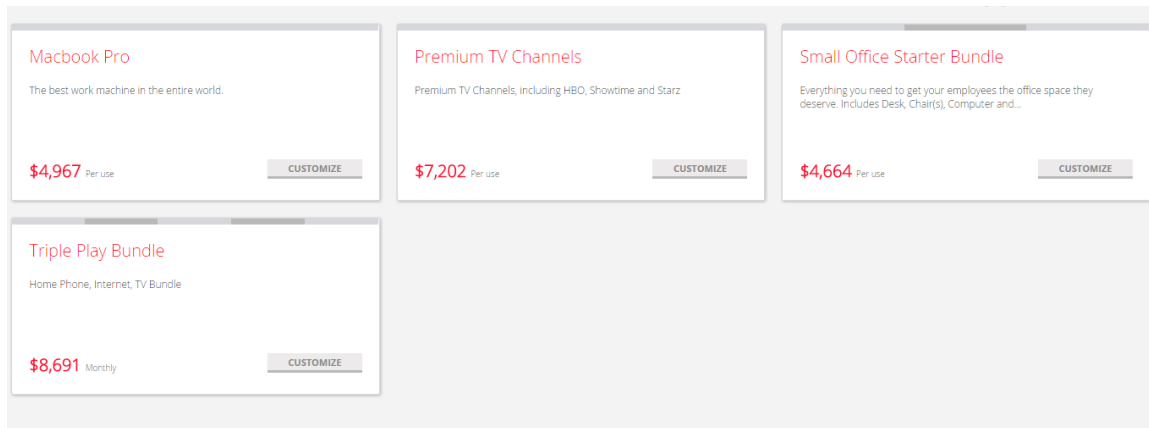


Order Capture System is an optional component that is a part of Fulfillment Orchestration and shipped within TIBCO Order Management.




Order Capture System User Interface Overview


This is an example of how the products are displayed after selecting a subscriber.



Each product or bundle must be customized before you can place it in the shopping cart.



When you click **CUSTOMIZE**, the product or bundle box expands, and displays how many products are within that bundle. To the top right of the expanded box, you can see the following three icons:


-  **Add to cart**
-  **Full screen**
-  **Close**

After adding a product or bundle to the cart, you can click the **Display shopping cart** icon  to view the items in the cart. On the Order details page, you can hover over the item and see the **Edit shopping cart**

icon  **Edit** and the **Remove from cart**  **Remove** icon. Using the edit icon, you can change the quantity of products or bundles and characteristic values. Using the remove icon, you can remove products or bundles from the cart.

When you click **CHECKOUT**, you can see the **Edit order** option. Using the **Edit order** option, you can add the order priority number, delivery date, or any notes to the product. Also on the Check Out window, you

can click the **BACK TO CART** button  or the **PLACE ORDER** button to submit the built order to TIBCO Order Management Long Running.



Searching for Subscribers

Log in to Order Capture System and search for a subscriber to complete tasks such as submitting a new order, amending an existing order, updating an order, or canceling an order.

Logging In

Log into Order Capture System using your specific user name and password.




A timeout mechanism automatically closes the session after 15 minutes and returns to the login page. After you are logged in, Order Capture System reloads the previous shopping cart, if there is one. If no previous shopping cart exists, you must select a subscriber.

Searching for a Subscriber

Search for a subscriber based on the subscriber's first name, last name, or ID to submit, amend, update, or cancel orders. For more information on the search syntax, see [Search Syntax](#).

Enter the subscriber's first name, last name, or ID in the customer search box, and press Enter or click the magnifying glass.

Customer Search


Search by name or ID 


Submitting an Order in Order Capture System

Select a subscriber, browse TIBCO Fulfillment Catalog, customize an order for the selected subscriber, and submit the order to TIBCO Order Management.

Procedure

1. Search for a subscriber. For instructions, see [Searching for Subscribers](#).
2. On the Customer search page, click **SELECT CUSTOMER**.
3. Browse the product catalog in one or more of the following ways:
 - **Categories:** select a product from the categories list.
 - **Search bar:** search for products by typing specific keywords.

 The search is case sensitive and supports complex searches. For more information, see [Search Syntax](#).
 - **Filters :** use the selected filters to correspond to a segmented list in TIBCO Fulfillment Catalog.

 By default, the selected filters correspond to the filters set in the subscriber's profile. You can clear them to see the impact on retrieved products.
 For example, a subscriber is moving states. Clearing the local area filter can display or hide certain products.
4. Click **CUSTOMIZE** for the product you want to order. If a bundle contains more than one product, multiple tabs are shown. Click each tab to customize each item.
5. After customizing, click **SAVE TO CART** or click the **Add to cart** icon. Repeat steps 3 and 4 to add more items to the cart.
6. Optional: Click the shopping cart icon to review your order. On the Shopping Cart page, click the product to view the details.
7. Click **CHECKOUT**. On the Checkout window, you can complete the following tasks:
 - **View order plan**, to open a window in the Order Management Server UI with the corresponding order
 - **Edit order**, to edit details such as priority, delivery date, and notes
 - **BACK TO CART**, to go back to the shopping cart to add, edit, or delete products
 - **PLACE ORDER**, to submit your finalized order
8. Click **PLACE ORDER**.

After you click **PLACE ORDER**, you see the Confirmation window. This window is displayed to indicate that Order Capture System submitted the order to TIBCO Order Management Long Running.

On the Confirmation window, you can start a new order for this customer or clear the customer selection:

- **NEW ORDER FOR THIS CUSTOMER:** using this feature, you can start a new order for the same subscriber.
- **CLEAR CUSTOMER SELECTION:** using this feature, you can submit an order for a different subscriber.



To make changes to a submitted order, see [Order Capture System](#).

Amending an Order in Order Capture System

Amend an order by modifying a submitted order.

Procedure

1. Search for a subscriber. For information on how to complete this task, see [Searching for Subscribers](#).
2. On the Customer search page, click the subscriber's name.
3. Click **View order history**.
On the Order history window, you can complete the following tasks:

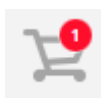
- View order details
- Modify an order
- Cancel an order



When the order is in execution, you can view the order details, modify, or cancel an order. When an order is complete or canceled, you can only view the order details.

4. View order details, modify, or add a product.
 - View order details
 1. Click **View order details** to see all the details associated with that specific order on a read only window.
 - Modify an order
 1. Click **Modify**.
 2. On the Shopping cart window, click the product you want to modify.
 3. Click **Edit**.
 4. Make necessary changes and click **FINISH EDITING**.
 - Add a product
 1. Click **Modify**.
 2. Click **Add products**.
 3. Search for a product or bundle and customize.
 4. Click **SAVE TO CART**.




5. Optional: Click the **Display Cart** icon



to compare orders. Click **COMPARE** to review the changes made.



The Compare orders window does not show price details for the original order to avoid displaying an incorrect price. Pricing models within Offer and Price Engine might have been changed and reloaded during the amendment.

6. Optional: Hover over the item, to see the **Edit**  **Edit** and **Remove**  **Remove** icons. Click the edit icon, to change quantity of products or bundles and the characteristic values. Click the remove icon, to remove products or bundles from the cart.
7. Click **CHECKOUT**.
8. Optional: Hover over the product or bundle, and click the **edit** icon  to edit any order details such as priority, delivery date, and notes, and click **Save**.
9. Click **PLACE ORDER**.

After you click **PLACE ORDER**, you see the Confirmation window. This window is displayed to indicate that Order Capture System submitted the order to TIBCO Order Management Long Running.

On the Confirmation window, start a new order for this customer or clear the customer selection:

- **NEW ORDER FOR THIS CUSTOMER:** using this feature, you can start a new order for the same subscriber.
- **CLEAR CUSTOMER SELECTION:** using this feature, you can submit an order for a different subscriber.

Canceling an Order in Order Capture System

Cancel an order for subscribers using the Order history page.

Procedure

1. Search for a subscriber. For information on how to complete this task, see [Searching for Subscribers](#).
2. On the Customer search page, click the subscriber's name.
3. Click **View order history**.
4. Click **Cancel** in the ACTIONS column.
5. Click **YES** in the pop-up window to confirm the cancellation.

Order Capture System Error Codes and Messages

Messages returned by the software consist of a code and an associated message.

The following table lists the error IDs and their descriptions.

Error Code	Error Message	Description
no server	The server is not responding. Please try again later.	The server is not responding to the GUI. "no server" indicates that an error code cannot be returned.
1	The server is not responding. Please try again later.	

Error Code	Error Message	Description
401	Your session has timed out. Please sign in again.	Every 15 minutes you are automatically logged out of the system. Log back in with your ID and password.
10002	The username or password you entered is invalid.	Enter a valid email or password.
10003	Your session has expired. Please sign in again.	The HTTP session is invalid. This might happen if you are still on a page when the session expired. Sign in to log back into the system.
10004	Your request is invalid. Please sign in again.	An HTTP request to the server is invalid according to server authentication. Sign in to log back into the system.
20000	There's a problem accessing the subscriber inventory. Please try again later.	This indicates failure to access the subscriber inventory. Reasons might be the network problem, configuration problem on the current application, configuration problem on the remote app, or implementation problem (bug). Retrying later might solve the problem depending on the nature of the problem.
20100	There's a problem retrieving the pricing info. Please try again later.	
20101	There's a problem retrieving the eligible products. Please try again later.	
20102	There's a problem submitting or modifying this order. Please try again later.	
20103	There's a problem retrieving the plan preview details. Please try again later.	

Error Code	Error Message	Description
20104	There's a problem rebuilding the product/bundle hierarchy. Are the catalogs used in Order Capture System and in the order you want to display the same?	
20200	There's a problem accessing the subscriber item inventory. Please try again later.	
20201	There's a problem retrieving the plan preview details. Please try again later.	

Search Syntax

The search input supports complex searches and is case sensitive. You can use this feature to search for subscribers or products. The search input supports complex searches and is case sensitive.

Complex Searches

When you search for products or bundles, you retrieve results with all letters in the query.

For example, if you search "co" you retrieve results having "co" in the names, such as "*computer*".

If you type "sm on", you retrieve results having both "sm" and "on" in the names, such as "*smart phone*".

If you type "double OR triple", you retrieve results having either "double" or "triple" in the names, such as "*double bundle*" and "*triple bundle*".

Case Sensitive Searches

When you search for products, the search is case sensitive.

For example, if you search 'experience', you retrieve objects having "Experience" or "experience". If you search for 'Experience', you retrieve products only having "Experience".

The Search can have many tokens, which acts as an "OR":

For example, if you search 'xoom droid', you retrieve objects having "xoom" or "droid" (and also any combination of Xoom, XOom, DRoID).

Tokens can be protected by double quotation marks to search for a phrase:

For example, if you search "'first smart phone'" and 'first smart phone', you do not retrieve the same amount of products.

Use a dash (-) in front of a token or phrase to exclude it from search.

For example, if you search '-droid', you do retrieve all objects that do not contain 'droid'.

Order Management Long Running User Interface

This section describes the TIBCO® Order Management Long Running User Interface.

The application UI provides the following features:


- a visual interface to view order details, order execution plans, plan for Fulfillment Provisioning, jeopardy rule configuration and activity logs
- facility to search orders fast
- a complete view of orders that were fulfilled or failed during the fulfillment process
- end-to-end tracking, storing and monitoring capability for orders in the order fulfillment system
- capability to perform actions on the orders being executed in the system
- add and submit order in TIBCO Order Management Long Running through Order Management Server UI
- show plan preview, which provides a view of the plan without adding it into TIBCO Order Management Long Running



The date is in the MM/DD/YYYY HH:MM:SS Z format, where Z is the time zone where the request is processed. For instance, UTC-7. You can configure the date from the Fulfillment Configurator.

You can perform the following actions on Order Management Long Running:

Order Management Actions	Description
Dashboard-specific actions	<p>Viewing Dashboard: View the Order Management Long Running Dashboard for summarized information about:</p> <p>Orders Panel</p> <ul style="list-style-type: none"> • Order Summary • Orders in Execution • Backlog Orders • Amended Orders <p>Jeopardy Panel</p> <ul style="list-style-type: none"> • Orders in Jeopardy • Jeopardy Live Alerts • Jeopardy Recorded Alerts <p>For details, refer to Dashboard .</p>

Order Management Actions	Description
Order-specific actions	<p>Order Management Long Running allows you to:</p> <ul style="list-style-type: none"> • View order Details • Search orders • Amend orders • Add orders • Show Plan preview for orders <p>For details, refer to Orders Page.</p>
Plan-specific actions	<p>Perform the following plan specific actions in the Order Management Long Running.</p> <ul style="list-style-type: none"> • View Plan Details • Search plan • View GANTT chart for the plan • Dependency View for the plan
Activity Log	<p>Shows the status and revision history of an object (order or a plan) and Fulfillment Provisioning (FP) logs based on the following criteria:</p> <ul style="list-style-type: none"> • Order Ref • Plan Id • Rule Name • FP Order Id • FP OrderLine Id • FP Resource Id • FP TechnicalOrder Id <div data-bbox="565 1352 1487 1451">  <p>All the Order Management Long Running related options are displayed only if Order Management Long Running configuration is enabled.</p> </div> <p>For details, refer to About Activity Log.</p>

Order Management Actions	Description
Rule Config	<p>The Rule Config panel allows you to add rule to receive Jeopardy notification on configured channel or invoke specific service if order is in jeopardy according to the configured condition.</p> <p>There are two notification types:</p> <ol style="list-style-type: none"> 1. Service 2. Notification <p>If you choose the 'service' as notification type, provide the service parameters in the respective tab.</p> <p>The notification channels are:</p> <ol style="list-style-type: none"> 1. JMS 2. File 3. Tibbr 4. SMTP
Catalog	<p>The Catalog panel allows you to view the catalog model associated with the Order Management Server user interface.</p> <p>You can:</p> <ol style="list-style-type: none"> 1. View a catalog model of all products. 2. Navigate from the order's product ID to view the specific catalog model w.r.t. product ID.

The Order Management Long Running also describes the functionalities of Fulfillment Provisioning related to:

- Activity log
- FP Service Order Hierarchy
- Display Service Order (attributes, parameters, and service logs)

Navigation

Access to the Dashboard is controlled through basic username and password authentication.

To access TIBCO Order Management System form a browser window, perform the following steps:



All latest versions of Google Chrome, Mozilla FireFox, and Internet Explorer are supported by OMSUI.

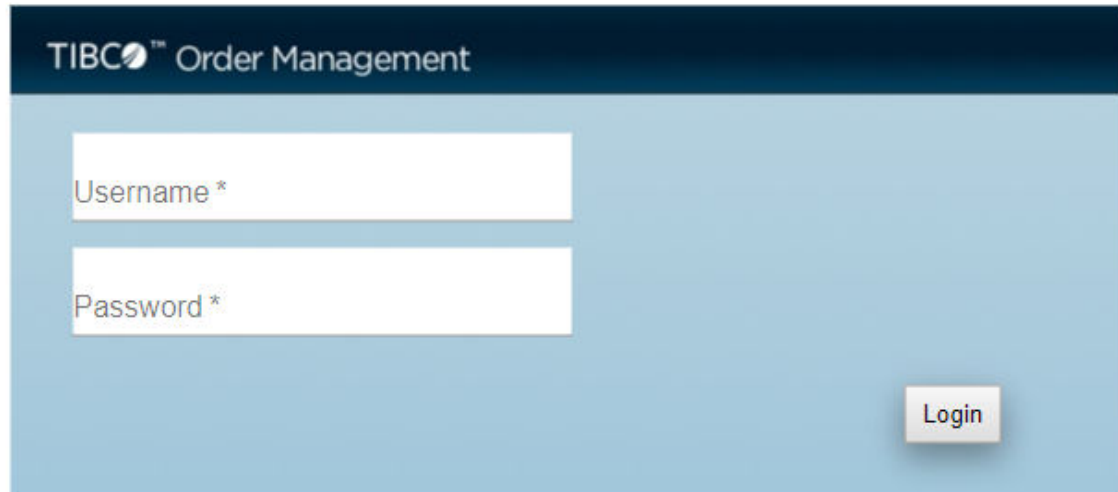
1. Go to the URL `http://<host>:<port number>/Login/Login.jsp` to access the Login page where:

- host is the computer where you installed the Order Management Long Running.
- port is the port number of the machine where Order Management Long Running installation listens to the requests. The default port number is 8080.



You can configure host and port from the Configurator.

Order Management Long Running Login



The login form for TIBCO Order Management features a dark blue header with the TIBCO logo and the text "TIBCO™ Order Management". Below the header, on a light blue background, are two white input fields. The first field is labeled "Username *" and the second is labeled "Password *". To the right of these fields is a grey "Login" button.

2. Enter the username and password to sign in. The default username and password is admin. The Order Management Long Running Dashboard is displayed. For details, see [Dashboard](#)

Order Management Long Running Dashboard

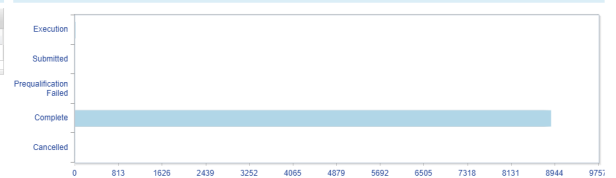
Dashboard

Order | Inflow Orders

Amended Orders

OrderRef Id	Customer Id	Status
No records found		

Orders Summary



Orders in Execution

OrderRef Id	Customer Id	Submitted Date
Order_17	AFFINITY_CUSTOMER	06/30/2017 12:30:28
Order_16	AFFINITY_CUSTOMER	06/29/2017 15:02:58
Order_7	TIBCO	06/27/2017 15:22:41
Order_6	TIBCO	06/27/2017 14:44:33
Order_5	TIBCO	06/27/2017 14:09:03
Order_4	TIBCO	06/27/2017 13:33:05

Backlog Orders

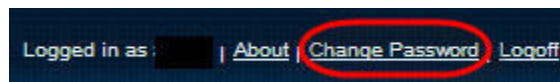
OrderRef Id	Customer Id	Status
No records found		

Changing Password

Order Management Long Running allows you to change your password in the dashboard. To change the password:

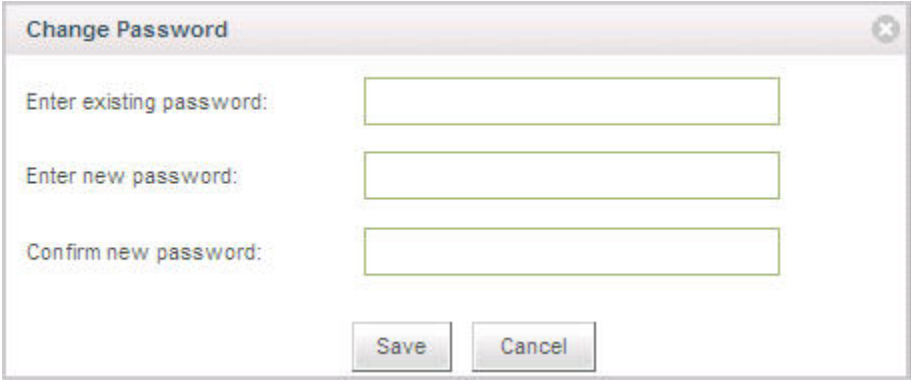
1. Login to the Order Management Long Running.
2. On the top right corner, click **Change Password**.

Change Password



3. In the **Change Password** dialog, enter your existing password, new password, and re-enter your new password to confirm the change.

Change Password Dialog



A dialog box titled "Change Password" with a close button in the top right corner. It contains three text input fields: "Enter existing password:", "Enter new password:", and "Confirm new password:". At the bottom, there are two buttons: "Save" and "Cancel".

- 4. Click **Save** to successfully change your password. Click **Cancel** to exit without saving the new password.

Order Management Server User Interface Logging Notifications

Order Management Server user interface is updated with bootstrap growls that are used for different status of messages like info, error, and warning, which align themselves based on an order.

A notification label with a notification count on Order Management Server user interface is introduced that shows the number of notifications.

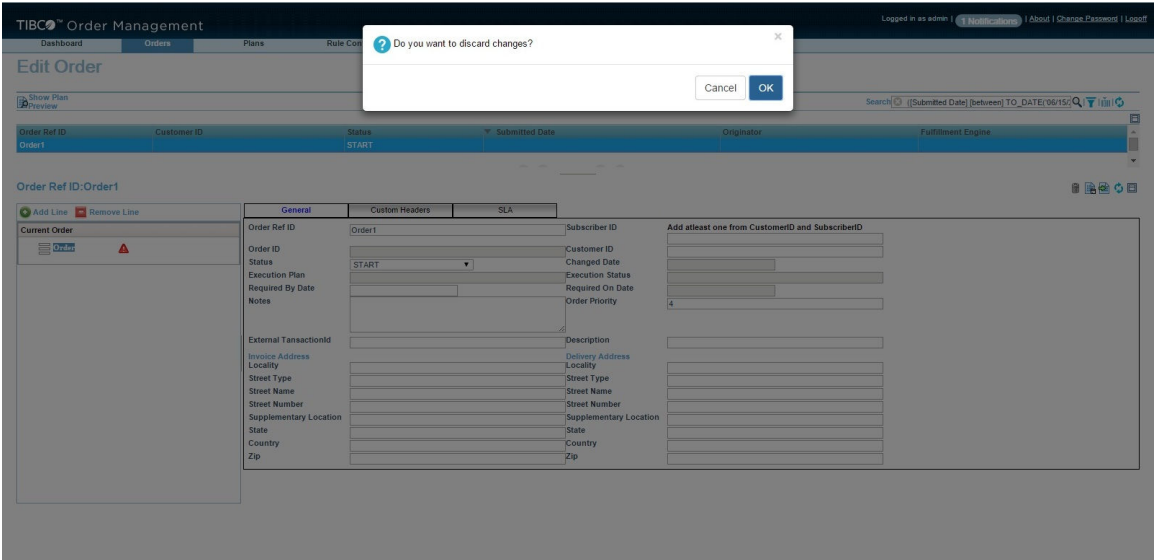
When clicked it shows the notification details for each level of logs.

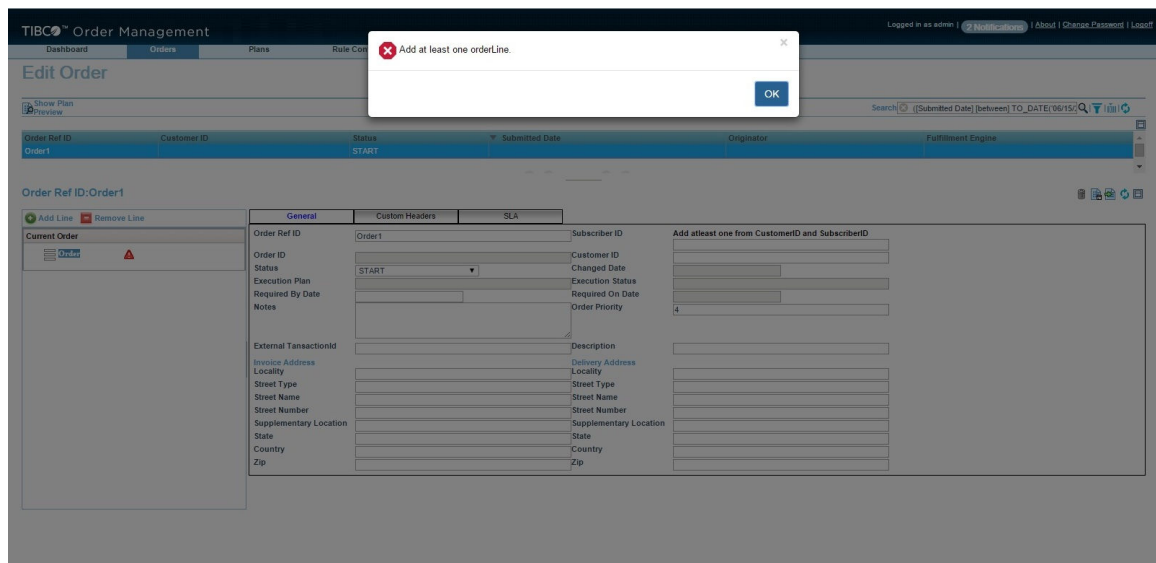
Clicking the notification label opens a notification pop up that maintains a different log details based on level.

Alert and Confirmation Box

Alert and confirmation box UI has been changed to facilitate the user to view on upper center of Order Management Server UI page irrespective of browser. Alerts can be of three types: Warning, Error and Success. Each have respective icons.

The following images are the user interface for Alert and Confirmation window:

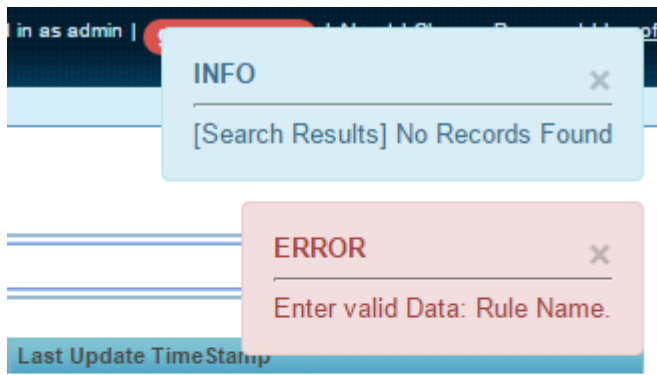




Growls for Information and Error Messages

Information and error messages has been removed from the top of the tab and each information and error is shown in growls with respect to their log level.

Growl shown have fade timeout and can be closed by user.



Notification Logger

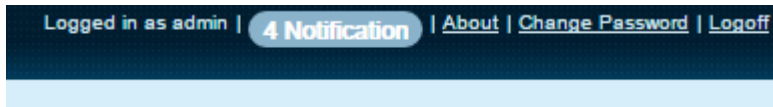
Notification Logger lets you view the count of logging notifications and their details. You can view previous and present client side logs on the Order Management Server UI.

The following are some of the major functionalities of Logger notification:

- Notification label implemented in Order Management Server UI next to logged in <Username> label.
- Initially **No Notification** label is shown, when there is no notification for the system is logged after login to Order Management Server UI.



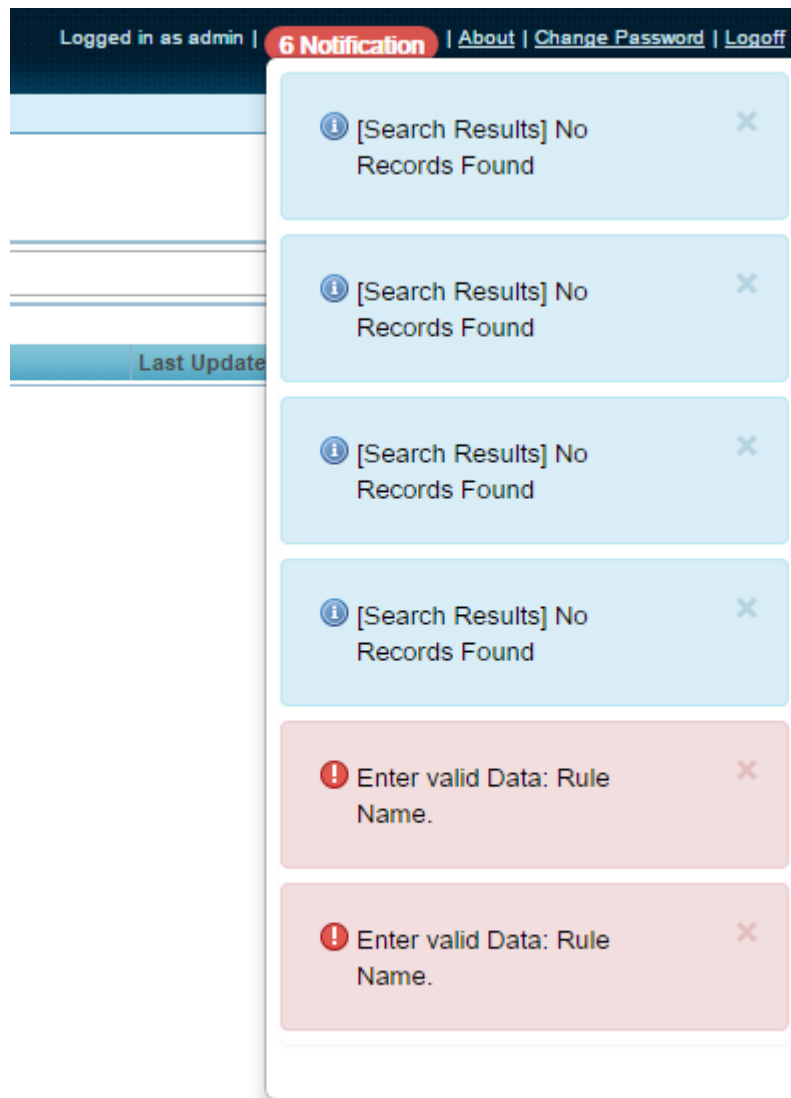
- There is a counter maintained for the number of notifications for each info, error, and warning messages logged in to the Order Management Server UI.
- You can see previous and present logs for the system.
- Notification label is shown in blue color until a critical log is logged into the system.



- Notification label is shown in red if there is at least one critical or error notification logged and it is present in notification detail window.



- There is a pop-over window for detailed description of different level of logs.
- User interface for log details shown depends upon the level of log. There are different user interfaces for each level error, information, and warning.
- User can delete or remove logs from the notification detail window by clicking the **Close** button.
- Closing or removing logs from notification detail view decreases notification count at run time.



- Level of logs to be shown and maintained in notification detail can be configured in `OMSUILog4j.xml` file with category: `com.tibco.aff`.

Order Management - Long Running Functionality

The Order Management UI consists of the following:

1. [Dashboard](#).
2. [Orders Page](#).
3. [Plans Page](#).
4. [Jeopardy Rule Configuration](#).
5. [GANTT Chart](#).
6. [About Activity Log](#).

Dashboard

An Order Management Long Running Dashboard is a graphical user interface that organizes and presents rich and enhanced information in a format that is easy to read and interpret. The Dashboard is the default view when you access the Order Management Long Running.

Features of the dashboard include:

- **An intuitive graphical display that is easy to navigate** - A rich Graphical User Interface (GUI) with user/role security to manage/view orders.
- **A logical structure that makes information easily accessible** - Ability to view all orders through graphical Dashboard summary.
- **Data displays that can be customized and categorized** - Ability to drill-down into order details by setting display preferences.
- **Regular and frequent updates of dashboard information for accuracy and relevance** - Ability to auto-refresh to display updated details for an order cancellation, amendment, suspension, and resumption.
- **Information from multiple sources can be viewed simultaneously** - Ability to manage, search and filter lists of existing orders.

The Dashboard allows effective order management with a comprehensive operations view. The information displayed is a combination of text and graphical views, as:

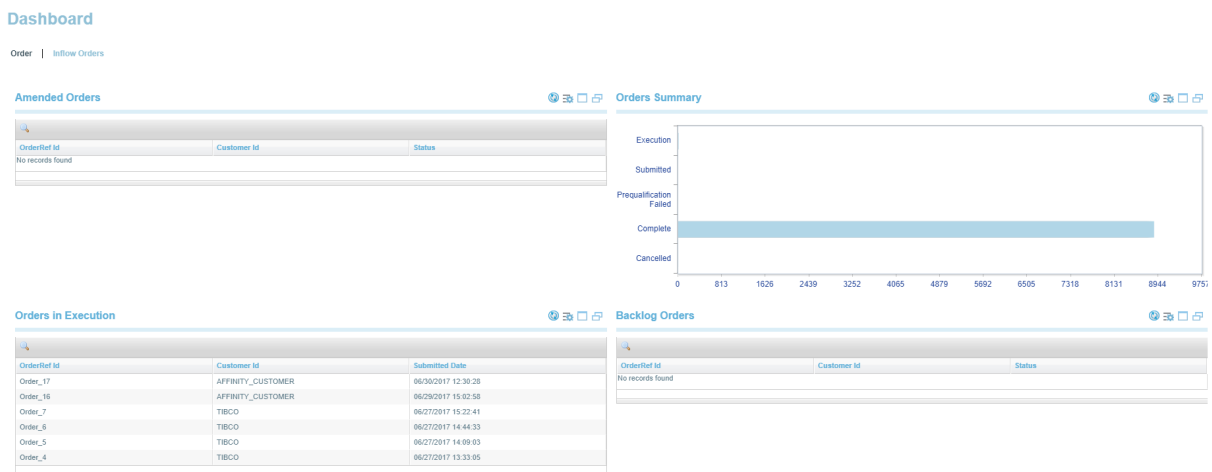
- Current number of orders being processed.
- Current number of orders completed in the last 24 hrs.
- Current number of orders in the Execution state.
- Current number of orders errored out in the last 24 hrs.
- Current number of orders amended in the last 24 hrs.
- Current number of suspended orders.
- Current number of orders in Jeopardy.
- Current number of Jeopardy live alerts.

Dashboard Components

The Dashboard displays the following panels:

- Orders Summary
- Amended Orders
- Backlog Orders
- Orders in Execution

Order Management Long Running Dashboard





All the above sections except Backlog Orders by default show the data for the Month. You can change the data window anytime by editing the display preferences.



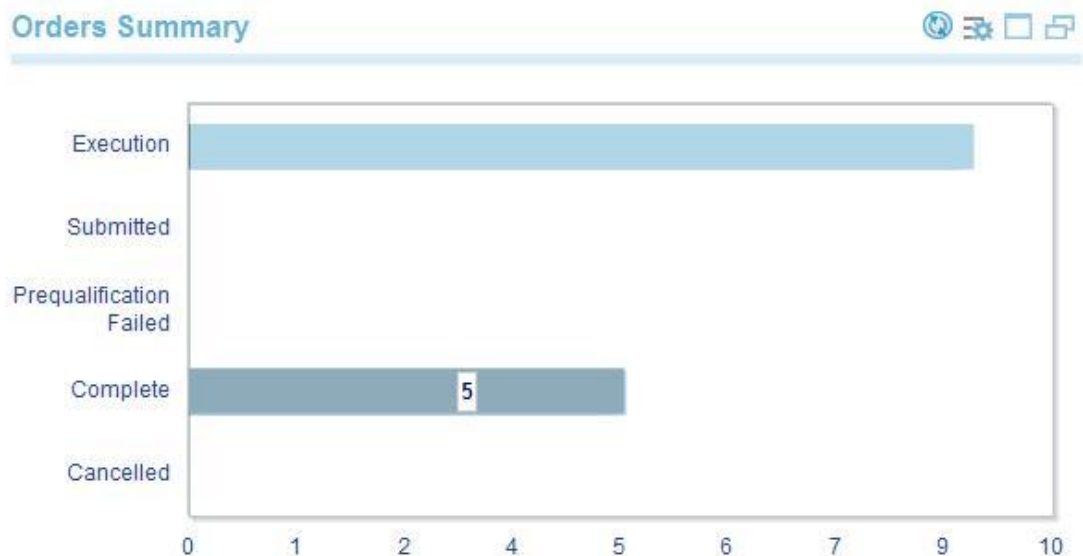
Every section updates itself periodically with the default interval of 300 seconds (300000 milliseconds). However, you can change the default interval by editing the preferences.

Order Summary

The Orders Summary section displays the summary of orders in terms of the order status and corresponding count of orders.

Summary is displayed in the form of a horizontal bar chart (order status versus number of orders). To see the details of a particular order status with the list of orders on the Orders page, click on the respective horizontal bar.

Order Summary Section




Under the Orders Summary section, you can perform the following operations:

- **Refresh Interval:** This is the Interval after which the chart is refreshed. To set the refresh interval, refer to [Auto-refreshing the Interval](#).
- **Set the Time Period to View the Chart:** You can set the time period to view the order chart. To do this, refer to [Viewing Order Summary Data Based on the Definite Time Period](#).

Auto-refreshing the Interval

Auto-refreshing the chart saves you the time and effort to view the fresh information on the Dashboard. The interval is in milliseconds.

To auto-refresh the chart after a definite time period, perform the following steps:

1. On the top-right corner, click the **Edit Preferences** icon .
2. In the **Auto refresh interval** field, enter the time (in milliseconds) that you wish to set to auto-refresh the Order Summary chart information.


3. Click the **Save** button to save the setting and exit the dialog. You are now set to see the refreshed Dashboard after the interval you have specified. Click the **Cancel** button to exit the dialog without saving the setting.

Viewing Order Summary Data Based on the Definite Time Period

The Dashboard allows you to customize the information format that is displayed through the chart. Here are the different time period parameters based on which you can see the chart data.

Time Period	Result Data Details
Day	The chart shows the data for the current day.
Week	The chart shows the data for the current week.
Month	The chart shows the data for the current month.
Year	The chart shows the data for the current year.

To view the chart for the specified time period, perform the following steps:

1. On the top-right corner on the Order Summary section, click the **Edit Preferences** icon .
2. From the Data Window list, select the time period from the available options - Day, Week, Month, or Year for which you wish to see the data.




The default selected option is Month.

3. Click the **Save** button to save the information and exit the dialog. The data is displayed based on the time period option you selected. Click the **Cancel** button to exit the dialog without specifying the time period - the data is displayed according to the default selected time period Month.

Backlog Orders

This section shows the list of *backlog orders*. The term *backlog orders* refers to those orders that have been not yet completed and not covered in the Order Summary section.

You can select, which columns you wish to see by editing the display preferences. *The steps to edit the preferences are similar to those of **Order Summary** or **Amended Orders** section.*

Like the Amended Orders section, by default you can see the top six records of the backlog orders. To see more orders, click the **Maximize** icon .

If you want to view details for any particular order, click on the respective row and see the details of the selected order on the Orders page.

Amended Orders

Amended Orders section provides you with the consolidated view of the list of amended orders. By default, records of the top six amended orders are displayed. To see more orders, click the **Maximize** icon



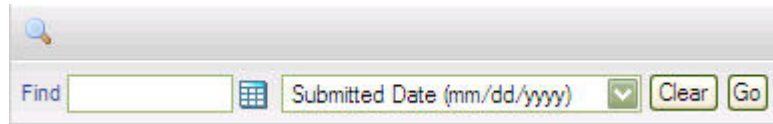
You can perform the following operations under the Amended Orders section:



- **View the Order Details:** If you wish to view the details for any particular order, click on the respective order link in the row. The **Orders** page with the selected order detail is displayed.

- **Customize the Amended Orders Display Format:** The Dashboard allows you to customize the information display under the Amended Orders section according to your convenience and ease of use. To know how, refer to [Setting Display Preferences for Amended Orders](#).
- **Filter the Amended Orders Data:** You can display the data in the Amended Orders section with the Submitted Date filter. To do this:

1. On the Amended Orders section, click the **Filter** icon . The Filter window is displayed.


Filtering Data




2. In the Find field, enter the (order) submission date in the *mm/dd/yyyy* format or click the **Calendar**  icon to select the date from the calendar.
3. Click the **Go** button to find orders submitted on the specified date. Click the **Clear** button to reset the order submission date.
4. Click the **Filter**  icon again to exit the **Filter** window (optional).

Setting Display Preferences for Amended Orders

You can customize the Amended Orders section display based on the following parameters:

- **Column Selection:** Choose from a list of available columns that you wish to see.
1. On the top-right corner on the Amended Orders section, click the  icon. The Edit Preferences dialog is displayed.
 2. Select the column you wish to see to display data in the Amended Orders grid. You can select from the following available options:

Column Header	Description
Show Order Id column	Displays the Order Ref ID column.  This column is always visible.
Show Customer Id column	Displays the Customer ID column.
Show SubscriberId column	Displays the Subscriber ID column.
Show Status column	Displays the Status column.
Show Submitted Date column	Displays the Submitted Date column.


- **Time period:** Set the time period to view the amended orders.
1. In the Edit Preferences dialog, select the time period for which you wish to see the amended orders. For example, if you wish to see the data on a weekly basis, select Week from the list. The default selected option is Month.

2. Click the **Save** button to save the information and exit the dialog. The data is displayed based on the time period option you selected. Click the **Cancel** button to exit the dialog without specifying the time period - the data is displayed according to the default selected time period Month.
- **Refresh Interval:** Set the auto-refresh interval in milliseconds to display the information for all the amended orders in a tabular format.
 1. In the Auto refresh interval field, enter the time (in milliseconds) that you wish to set to auto-refresh the Amended Orders information.
 2. Click the **Save** button to save the setting and exit the dialog. You are now set to see the refreshed Dashboard for Amended Orders after the interval you have specified. Click the **Cancel** button to exit the dialog without saving the setting.

Orders in Execution

This section shows the list of all the orders, which are in the execution state.

Like other sections,

- You can select, which columns you wish to see by editing the display preferences. *The steps to edit the preferences are similar to those of **Order Summary** or **Amended Orders** section.*
- By default you can see the top six records of the orders. To see more orders, click the **Maximize** icon .
- If you want to view details for any particular order, click on the respective row and see the details of the selected order on the **Orders** page.

Inflow Orders

Inflow Orders is a new tab introduced on Order Management Server Dashboard. You can use the Inflow Orders tab to generate a graph for the orders that were submitted for a specified period.

You can modify the period by clicking the **Edit Preferences** icon and clicking the **Data window** drop-down box. The choices are as follows:

- Day
- Week
- Month
- Year

The default selection for period is Month.

The graph displayed is a bar graph. Each bar in the graph contains a numerical value, which indicates the number of orders for that timeline. You can either click the bar or the numerical value to view the list of orders for that particular timeline.

Jeopardy Dashboard

This section explains the capabilities of Orders in Jeopardy and the Jeopardy Alerts panels and how to integrate this with the existing Dashboard components. Current Dashboard user interface has been enhanced and improved for these additional jeopardy panels.



By default, Jeopardy is deployed in collocated mode with omsServer. Jeopardy Management System is enabled by default in the omsServer service. To disable it, you have to disable this for both omsui and omsServer. To do this, set `com.tibco.fom.jeoms.deployMode=JEOMS_disabled` in `profiles.properties` in the config folder for omsui and omsServer.

The following are the Jeopardy panels:

- Orders In Jeopardy
- Jeopardy Live Alerts
- Jeopardy Recorded Alerts

Orders In Jeopardy

The Order in Jeopardy panel lists all the order in a jeopardy state in a

To access the Orders in Jeopardy panel, click the Jeopardy tab.

The following are the details of the Jeopardy tab:

OrderRef Id	Order reference id of the order in jeopardy. Clickable and takes to order window.
Plan Id	Plan Id of the order in jeopardy. Clickable and takes to Plan Screen on the Order Management Server UI.
Risk Region	<p>The Risk Region comprises below values:</p> <ul style="list-style-type: none"> • NORMAL - the plan item section completed less than the typical duration • HAZARD - the plan item section completed greater than the typical duration but less than the maximum duration • CRITICAL - the plan item section completed greater than the maximum duration • OUT OF SCOPE – the plan exceeded last acceptable duration limit

The Orders in Jeopardy Panel

Orders in Jeopardy



OrderRef Id	Plan Id	Risk Region
p:orderRef	1	HAZARD
p:orderRef	2	HAZARD
p:orderRef	3	CRITICAL
p:orderRef	4	OUT_OF_SCOPE
p:orderRef	3	CRITICAL
p:orderRef	5	CRITICAL

The Orders in Jeopardy panel allows you to select columns you wish to see by editing the display preferences. The steps to edit the preferences are similar to those of the Order Summary or Amended Orders section.

To view details for any particular order in jeopardy, click the respective value in the Order Ref Id column. The details of the selected order are displayed on the Orders page. Similarly, you can view the order details and jeopardy plan details on the Plans page when you click the value in the Plan Id column.

Jeopardy Live Alerts

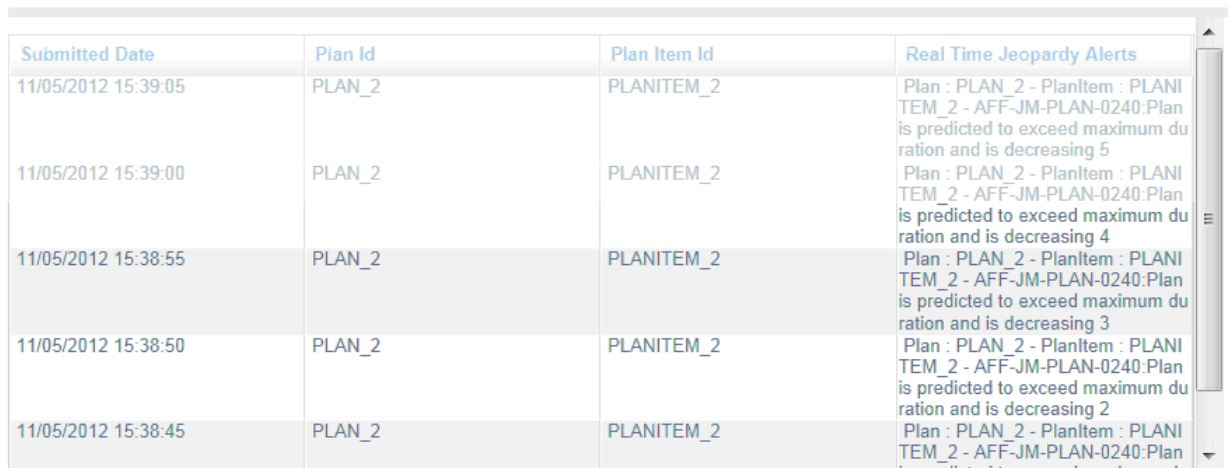
The Jeopardy Live Alerts panel displays the live alerts from the Jeopardy Manager.

The details of the Jeopardy Live Alerts panel are as follows:

Submitted Date	Timestamp details when jeopardy alert message received
Plan Id	If present in the incoming message from Jeopardy Manager, Plan Id of the order is display under Plan Id column.
Plan Item Id	If present in the incoming message from Jeopardy Manager, Plan Item Id of the order is display under Plan Item Id column.
Real Time Jeopardy Alerts	Actual text message contents of the alert

Jeopardy Live Alerts

Jeopardy Live Alerts



Submitted Date	Plan Id	Plan Item Id	Real Time Jeopardy Alerts
11/05/2012 15:39:05	PLAN_2	PLANITEM_2	Plan : PLAN_2 - PlanItem : PLANITEM_2 - AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 5
11/05/2012 15:39:00	PLAN_2	PLANITEM_2	Plan : PLAN_2 - PlanItem : PLANITEM_2 - AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 4
11/05/2012 15:38:55	PLAN_2	PLANITEM_2	Plan : PLAN_2 - PlanItem : PLANITEM_2 - AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 3
11/05/2012 15:38:50	PLAN_2	PLANITEM_2	Plan : PLAN_2 - PlanItem : PLANITEM_2 - AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 2
11/05/2012 15:38:45	PLAN_2	PLANITEM_2	Plan : PLAN_2 - PlanItem : PLANITEM_2 - AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 1

Select, which columns you wish to see by editing the display preferences. To edit the preferences, perform the steps mentioned in the Order Summary or Amended Orders section.



You cannot refresh Interval.

If you click on Refresh button, all existing live alert messages are cleared out. Afterwards these cleared messages are available under Jeopardy Recorded Alerts. The incoming messages are always displayed at the top and push the old messages down.

If you want to view details for any particular plan, click the respective row value. The details of the selected plan are displayed on the Plans page.

Jeopardy Recorded Alerts

The Jeopardy Recorded Alerts panel displays the recorded alerts from the Jeopardy Manager.

The Jeopardy Recorded Alerts panel displays the following information:

Submitted Date	Timestamp details when jeopardy alert message recorded in Order Management Server
Plan Id	If present in the incoming message from Jeopardy Manager, Plan Id of the order is display under Plan Id column.
Plan Item Id	If present in the incoming message from Jeopardy Manager, Plan Item Id of the order is display under Plan Item Id column.

Historical Jeopardy Alerts

Actual text message contents of the alert received in Order Management Server.

Jeopardy Recorded Alerts Panel

Jeopardy Recorded Alerts



The screenshot shows a web interface titled 'Jeopardy Recorded Alerts'. It features a table with four columns: 'Submitted Date', 'Plan Id', 'Plan Item Id', and 'Historical Jeopardy Alerts'. The table contains five rows of data, all with the same values for the first three columns: '11/05/2012 15:39:05', 'PLAN_2', and 'PLANITEM_2'. The 'Historical Jeopardy Alerts' column contains five different messages, each starting with 'AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing' followed by a number from 5 down to 1. To the right of the table is a vertical toolbar with icons for search, settings, and other actions.

Submitted Date	Plan Id	Plan Item Id	Historical Jeopardy Alerts
11/05/2012 15:39:05	PLAN_2	PLANITEM_2	AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 5
11/05/2012 15:39:00	PLAN_2	PLANITEM_2	AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 4
11/05/2012 15:38:55	PLAN_2	PLANITEM_2	AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 3
11/05/2012 15:38:50	PLAN_2	PLANITEM_2	AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 2
11/05/2012 15:38:45	PLAN_2	PLANITEM_2	AFF-JM-PLAN-0240:Plan is predicted to exceed maximum duration and is decreasing 1

Select the columns you wish to see by editing the display preferences. This panel shows the alerts for the past 1 hour (default selected in Edit Preferences), 12 hours or 24 hours. To accommodate newly generated alerts under this panel, refresh or setup the refresh interval in the Edit Preferences icon.

To view details of any particular plan, click the respective row value. The details of the selected plan are displayed on the Plans page.

Orders Page

On the Order page you can view details about the order, including status and revision history of the orders, add orders, and show plan preview. You can also edit the order and resubmit the order for further fulfillment process.

You can also view the details for initial request and amendments and also take action on the orders. The following actions are allowed on the Orders page:

- **Suspend Order:** This is available when an order is in the execution state and not yet completed. This state is also available when an order fails.
- **Cancel Order:** This is available before the order goes to the completed state.
- **Amend Order:** This is available in the Order Management Server UI when the order is in the suspended state.
- **Withdraw Order:** This is available when the order is in the execution or suspended state. Once the order is withdrawn, it is not listed in the Order or Plans pages. However, the audit log for the withdrawn order can be viewed on the Activity Log page.
- **Resume Order:** This is available when the order in the execution state.
- **Show Execution Plan:** You can view the execution plan for the respective order by clicking the Show Execution Plan option. This option displays the Plans page and the respective plan's details .
- **Show Activity Log:** You can view the activity log for the respective order by clicking the Show Activity Log option. Show Activity Log shows flow of change in statuses for each of the entities: order, order-line, plan, and plan-items.
- **Show Catalog:** Click on Product Id in orderline tab to view the related catalog.
- **Add Order:** You can submit new order functionality through the order component.

- **Plan Preview:** You can generate a plan preview by providing submit order XML without submitting order.

You must have ADMIN privileges (USER_ADMIN role) to do the above mentioned operations.



You have to suspend an order before amending an order.

Viewing Order Priority

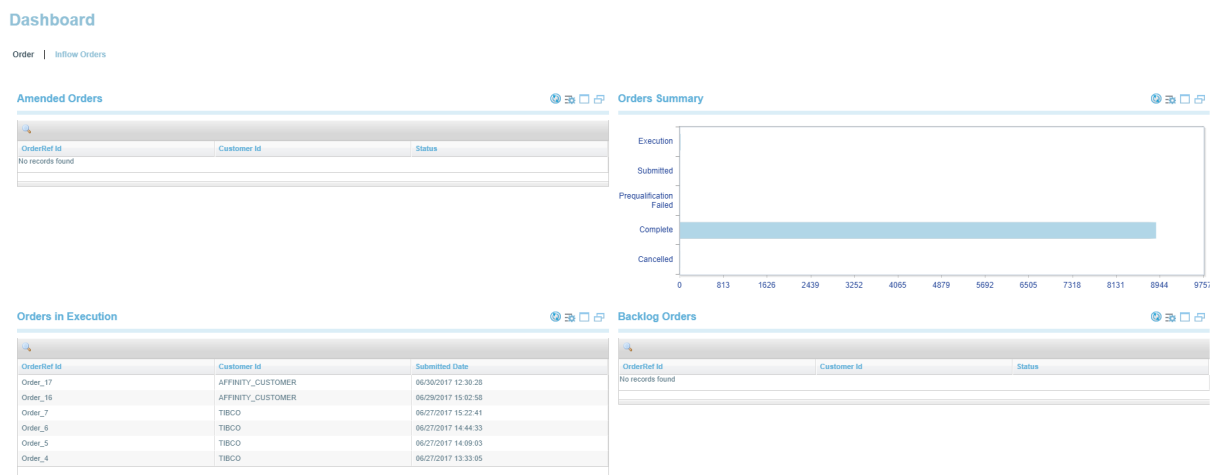
To track the order, perform the following steps:

1. Go to `http://<host>:<port number>/omsui/Login/Login.jsp` to access the Login page where:
 - host is the computer where you installed the Order Management.
 - port is the port number of the machine where Order Management installation listens to the requests. The default port number is 8080.

Order Management System Login

2. Enter the username and password to sign in.
The Order Management Server Dashboard is displayed.

Order Management System Dashboard



3. Go to the Orders page.
The order priority is displayed along with the other order details.

Order Priority


Browse Orders

Searching for an Order

You can search for the order from the Order Management Long Running GUI. The Orders page allows you to key in the order reference and filters for the search results. The Orders page shows paginated view of search results. You can click on the order link to view details about the order and perform an action on order.

Do the following to search the order:

- Go to the Orders page.


Click  to refine your search using the search criteria specified in the following table:

Search Criterion	Description
Order Ref ID	The unique identifier of the order, supplied by the order originator.
Customer ID	The reference that enables the Order Management Server to retrieve the current customer profile and to identify the customer to other systems interested in the order.
Status	<p>The current status of the order.</p> <p>The different statuses are:</p> <ul style="list-style-type: none"> • START • FEASIBILITY • OPD • EXECUTION • SUSPENDED • COMPLETE • CANCELLED • WITHDRAWN • PREQUALIFICATION FAILED
Submitted Date	The date of order submission.

Search Criterion	Description
Originator	Order Management, where the orders are created.
Fulfillment Engine	Advanced Fulfillment Orchestration or IPC (iProcess Conductor). Advanced Fulfillment Orchestration for fulfillment orchestrator.
Order ID	Order Id is the internal identifier of the Order generated by Order Management Server.
OPD Source	The value can be Automated Order Plan Development or Manual Order Plan Development.

You can use a combination of search criteria to search for a specific order object. For example, you can select a status of *COMPLETE* from the Status column and type the username in the Originator column to find all orders with a status of *COMPLETE* that were created by a particular user.

The search is case-sensitive.

- Click  to display the list of orders that match your search criteria. Select the order to display its details.

Viewing Order Information

The Orders page allows you to view the consolidated view of the order details and perform action(s) on an order.

To view the consolidated order information, complete the following steps:

Go to the Orders page and click a row in the list of orders to view an order (highlighted in red in the figure below). The order details are displayed in the Order Grid View (highlighted in green below).



The screenshot shows the 'Browse Orders' page with a table of orders. The row for Order Ref ID: OS_111 is highlighted in red. Below the table, the 'Order Grid View' for Order Ref ID: OS_111 is displayed, showing details in two panels: the Tree View and the Details view. The Tree View shows the order status as 'COMPLETE' and the Details view shows the order information, including the Subscriber ID, Customer ID, and Order ID.

Order Ref ID	Customer ID	Status	Submitted Date	Fulfillment Engine	Subscriber ID
OS_111	Pooja	COMPLETE	11/07/2019 01:12:56 UTC-8	AFO	

General	Custom Headers	SLA
Order Ref ID	OS_111	
Order ID	361	
Status	COMPLETE	
Execution Plan	362	
Required By Date		
Notes		
Description	ProductDependsOn	
Invoice Address		
Locality	PUNE	
Street Type	PUNE	
Street Name	PUNE	
Street Number	PUNE	
Supplementary Location	PUNE	
State	MPO	
Country	AU	
Zip	1234	






Subscriber ID	Customer ID	Changed Date	Execution Status	Required On Date	Order Priority
Pooja		11/07/2019 01:16:09 UTC-8	COMPLETE		5

The Order Grid view displays the details for the order in two panels: the Tree View and the Details view.





Tree View

The Tree View displays the status of the order and each order line. Click an order or order line to display the details in the Details View.

If an order has been amended, the Tree View displays the information of the original order and its order lines and the amended order and its order lines, with the current status.

Current Order	
 Order	COMPLETE
 1:PO_ACCESS:PROVIDE	
 2:PO_OSI:CEASE	
Original Request	
Amendments	

The order line status and notifications are listed below:

Status	Icon Used
START	
PENDING	
COMPLETE	
CANCELLED	

Order Details View

Order Details View gives different information at order level and order line level.

Click 'Order' in the Order Tree View to display details at the order header level.

The order line status and notifications are listed below:

General	Custom Headers	SLA
Order Ref ID	AMD_TC01_CK_12	Subscriber ID
Order ID	01902199-6302-4709-8071-81c28d0e479	Customer ID
Status	COMPLETE	Changed Date
Execution Plan	063a0a5-0075-427d-868b-81f71c6b3f1	Execution Status
Required By Date		Required On Date
Notes		Order Priority
Invoice Address		Delivery Address
Locality	PUNE	Locality
Street Type	PUNE	Street Type
Street Name	PUNE	Street Name
Street Number	PUNE	Street Number
Supplementary Location	PUNE	Supplementary Location
State	PUNE	State
Country	IN	Country
Zip	1234	Zip

Click an individual order line in the Tree View to display order line level details.


The order line status and notifications are listed below:

General	Custom Headers	SLA	Characteristics
Line No		1	Subscriber ID
Product ID		PO_ACCESS	Product Version
Inventory ID		1	LinkID
Action		PROVIDE	Status
Required By Date			Status Changed
Quantity		1	UOM
Required On Date			Notes
Delivery Address			
Locality		PUNE	
Street Type		PUNE	
Street Name		PUNE	
Street Number		PUNE	
Supplementary Location		PUNE	
State		PUNE	
Country		IN	
Zip		1234	

Suspending an Order


When you suspend an order, the TIBCO Order Management - Long Running suspends each of the processes that are attached to the execution plan. This means that depending on their current statuses, each of the processes associated with the process components and the plan tasks is suspended.

To suspend an order, complete the following steps:

1. Go to the Orders page.
2. Select an order in the order row, which you wish to suspend. You can view the details of the order in the pane below. You can view the current order details, original request, and amendments in the below pane along with the status and revision history of the order.
3. On the top right of the Orders page, click the  **Suspend Execution** icon.


Resuming an Order

To resume an order, complete the following steps:

1. Go to the Orders page.
2. Select an order in the order row, which you wish to resume. You can view the details of the order in the pane below. You can view the current order details, original request, and amendments in the below pane along with the status and revision history of the order.
3. On the top right of the Orders page, click the  **Resume Execution** icon.

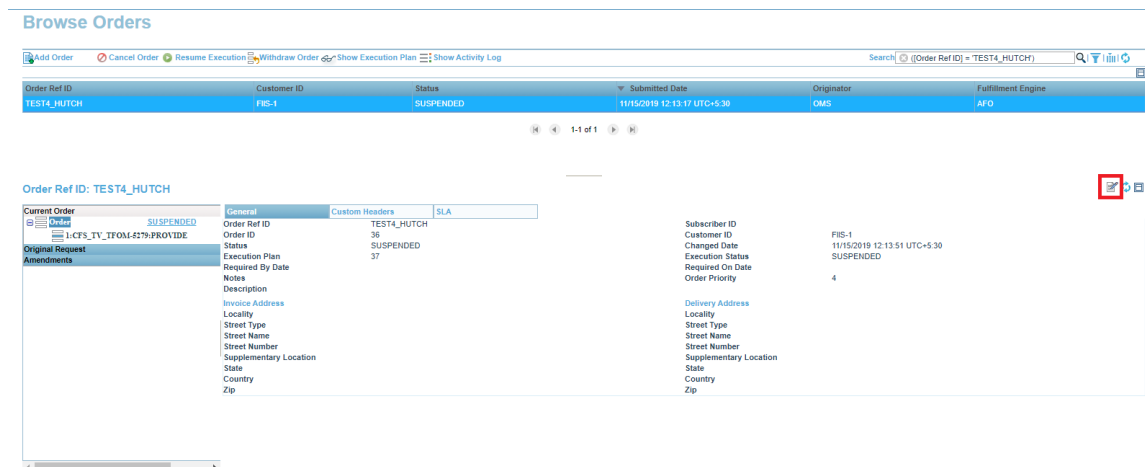
Canceling an Order

To cancel an order, complete the following steps:

1. Go to the Orders page.
2. Select an order in the order row, which you wish to cancel. You can view the details of the order in the pane below. You can view the current order details, original request, and amendments in the below pane along with the status and revision history of the order.
3. On the top right of the Orders page, click the  **Cancel Order** icon.

Amending an Order

You can amend an order when an order is in SUSPENDED state. When an order is in SUSPENDED state you are given an option to edit the order.



The screenshot displays the 'Browse Orders' interface. At the top, there are navigation buttons: Add Order, Cancel Order, Resume Execution, Withdraw Order, Show Execution Plan, and Show Activity Log. A search bar contains the text '(Order Ref ID) = TEST4_HUTCH'. Below this is a table with columns: Order Ref ID, Customer ID, Status, Submitted Date, Originator, and Fulfillment Engine. The table shows one row for 'TEST4_HUTCH' with status 'SUSPENDED' and originator 'OMS'. Below the table, there is a pane for 'Order Ref ID: TEST4_HUTCH'. This pane has tabs for 'Current Order', 'Original Request', and 'Amendments'. The 'Current Order' tab is active, showing details for a 'SUSPENDED' order. The details are organized into sections: General (Order Ref ID, Order ID, Status, Execution Plan, Required By Date, Notes, Description, Invoice Address, Locality, Street Type, Street Name, Street Number, Supplementary Location, State, Country, Zip), Custom Headers (Order ID, Status, Execution Plan, Required By Date, Notes, Description, Invoice Address, Locality, Street Type, Street Name, Street Number, Supplementary Location, State, Country, Zip), and SLA (Subscriber ID, Customer ID, Changed Date, Execution Status, Required On Date, Order Priority). The 'Original Request' and 'Amendments' tabs are also visible but not active.

After selecting the edit option, you can perform the following actions before sending the amendment for the order:

- Add a new order line by clicking the Add Line option.
- Remove existing order line by selecting an existing order line and then clicking the Remove Line option.

- Add new custom header of Delete existing custom header.
- At order line level add new or delete existing Characteristics.
- Update the existing order/order line fields.

After completing the changes in the SUSPENDED order, click the Post icon to post the amend order request.

The submitting the amend order request, the SUSPENDED order immediately comes into EXECUTION state.

Performing Bulk Action on Orders

Through bulk action functionality, you can **SUSPEND**, **RESUME**, or **WITHDRAW** a group of orders.

Procedure

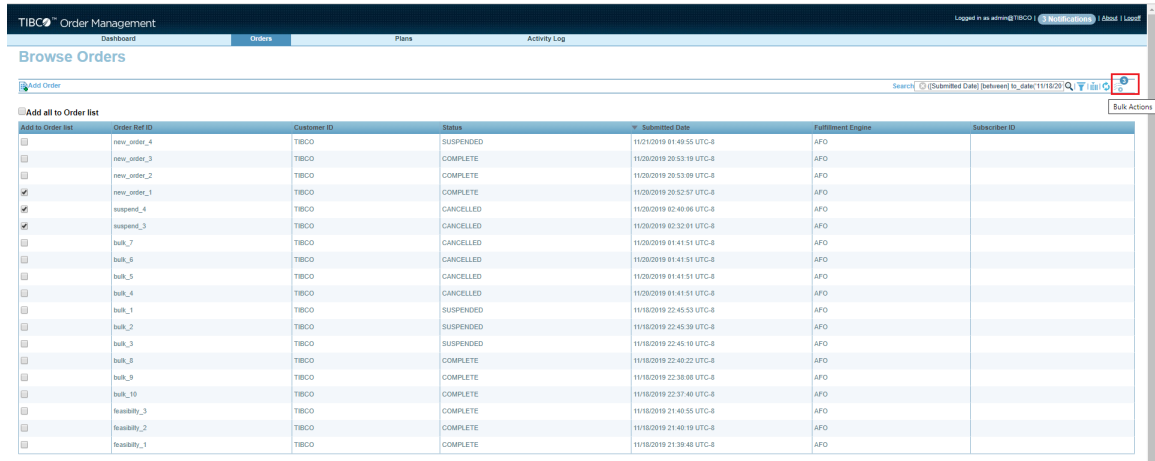
1. On **Orders** tab in Order Management Server UI, in the **Add to Order list** column, select the check box for the orders that you want to add to bulk action list.
In the upper-right corner, a pop-up notification shows a confirmation for the action (orders added to bulk action list).

Select Orders_Bulk Order Action

2. To add all the orders to bulk action list, select the **Add all to Order list** check box.

3. Click **Bulk Actions** icon in the top right corner. The icon also shows the number of orders you have selected.
- The **Order List** window opens.

Bulk Actions icon



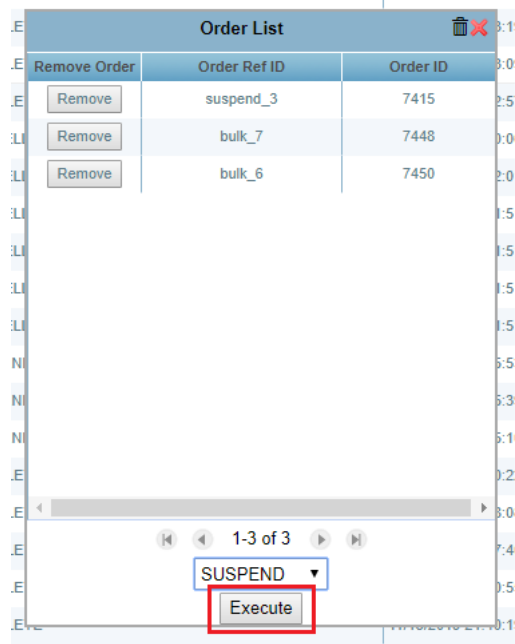
Order Ref ID	Customer ID	Status	Submitted Date	Fulfillment Engine	Subscriber ID
new_order_4	TIBCO	SUSPENDED	11/21/2019 01:49:55 UTC-8	AFO	
new_order_3	TIBCO	COMPLETE	11/20/2019 20:53:19 UTC-8	AFO	
new_order_2	TIBCO	COMPLETE	11/20/2019 20:53:09 UTC-8	AFO	
new_order_1	TIBCO	COMPLETE	11/20/2019 20:52:57 UTC-8	AFO	
suspend_4	TIBCO	CANCELLED	11/20/2019 02:40:06 UTC-8	AFO	
suspend_3	TIBCO	CANCELLED	11/20/2019 02:32:01 UTC-8	AFO	
bulk_7	TIBCO	CANCELLED	11/20/2019 01:41:51 UTC-8	AFO	
bulk_8	TIBCO	CANCELLED	11/20/2019 01:41:51 UTC-8	AFO	
bulk_5	TIBCO	CANCELLED	11/20/2019 01:41:51 UTC-8	AFO	
bulk_4	TIBCO	CANCELLED	11/20/2019 01:41:51 UTC-8	AFO	
bulk_1	TIBCO	SUSPENDED	11/18/2019 22:45:53 UTC-8	AFO	
bulk_2	TIBCO	SUSPENDED	11/18/2019 22:45:39 UTC-8	AFO	
bulk_3	TIBCO	SUSPENDED	11/18/2019 22:45:10 UTC-8	AFO	
bulk_6	TIBCO	COMPLETE	11/18/2019 22:40:22 UTC-8	AFO	
bulk_9	TIBCO	COMPLETE	11/18/2019 22:38:06 UTC-8	AFO	
bulk_10	TIBCO	COMPLETE	11/18/2019 22:37:40 UTC-8	AFO	
readability_3	TIBCO	COMPLETE	11/18/2019 21:40:55 UTC-8	AFO	
readability_2	TIBCO	COMPLETE	11/18/2019 21:40:19 UTC-8	AFO	
readability_1	TIBCO	COMPLETE	11/18/2019 21:39:48 UTC-8	AFO	



You can click **Delete** icon to clear the entire order list or can click **Remove** button against the orders that you want to remove.

4. Select an action **SUSPEND**, **RESUME**, or **WITHDRAW** from the drop-down list and then click **Execute** button.

Execute button_Bulk Order Action



After the request is submitted, the list is cleared and closed.



Only those orders that are eligible for the chosen action are executed and rest orders are ignored. See [Orders Page](#) for more information on eligibility of orders for an action. If no order is eligible, the list is kept open for the user to make the necessary changes.


Import Order

```
<ord:SubmitOrderRequest ExternalBusinessTransactionID="12345" xmlns:ord="http://www.tibco.com/aff/orderservice"
xmlns:ord1="http://www.tibco.com/aff/order" xmlns:com="http://www.tibco.com/aff/commontypes">
  <ord:orderRequest>
    <ord1:orderRef>ORDER REF 0004</ord1:orderRef>
    <ord1:header>
      <!--Optional-->
      <ord1:description>ORDER REF 0004</ord1:description>
      <!--Optional-->
      <ord1:customerID>TIBCO</ord1:customerID>
      <!--Optional-->
      <ord1:subscriberID>ABC</ord1:subscriberID>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional-->
      <!--<ord1:requiredBy>2011-05-16T13:20:00+05:30</ord1:requiredByDate-->
      <!--Optional-->
      <ord1:invoiceAddress>
        <com:line1>line1</com:line1>
        <!--Optional-->
        <com:line2>line2</com:line2>
        <!--Optional-->
        <com:line3>line3</com:line3>
        <com:locality>locality</com:locality>
        <!--Optional-->
        <com:region>region</com:region>
        <com:country>US</com:country>
        <com:postCode>11111</com:postCode>
        <!--Optional-->
        <com:supplementaryLocation>supplementaryLocation</com:supplementaryLocation>
      </ord1:invoiceAddress>
      <!--Optional-->
      <ord1:deliveryAddress>
        <com:line1>line1</com:line1>
        <!--Optional-->
        <com:line2>line2</com:line2>
        <!--Optional-->

```

Save Reset Cancel

3. Submit the file `SubmitOrder.xml` in the text area and click the **Save** button in the **Import Order** dialog. The values are displayed in respective attributes.
4. Click **Plan Preview** or **Save Order** to add an order.

5. Click **Show Plan Preview** button  **Show Plan Preview**.
The plan preview for order is generated without submitting the order in Order Management Long Running.
6. Click **Save Order**.
The order is saved in Order Management Long Running.



You can also choose to discard submitting order by clicking the **Discard** button.

Plans Page

The Plans page provides details for the plan generated by Automated Order Plan Development. Clicking the Plans tab gives you a tabular list for all the plans in your database repository.

You can even filter plans. For more information, see [Searching for a Plan](#).



This tabular view of plans is supported by configurable pagination. Plans per page can be configured through the Order Management Server configurator. Refer to *TIBCO Order Management Long Running Administration* for details.

You can click an individual plan in the table and see the details for the plan. Each of these views is displayed in the bottom panel.

1. [Grid View](#)
2. [Accessing Dependency View of Plan Items and their Status](#)
3. [GANTT Chart](#)
4. [Activity Log](#)

Searching for a Plan

The Plans page allows you to key in the order reference and filters for the search results and the page shows a paginated view of the search results. To view Plan details, the plan link.

Do the following to search the plan:

- Go to the Plans page.

Click  to refine your search using the search criteria specified in the following table:

Search Criterion	Description
Order Ref ID	The unique identifier of the order, supplied by the order originator.
Plan ID	The unique identifier of the plan.
Description	Description of the plan.
Status	<p>The current status of the Plan.</p> <p>The different statuses are:</p> <ul style="list-style-type: none"> • START • FEASIBILITY • OPD • EXECUTION • SUSPENDED • COMPLETE • CANCELLED • WITHDRAWN

Search Criterion	Description
Plan-Item Status	<p>The current status of the Plan-Item.</p> <p>The different statuses are:</p> <ul style="list-style-type: none"> • START • FEASIBILITY • OPD • EXECUTION • SUSPENDED • COMPLETE • CANCELLED • WITHDRAWN • ERROR • ERROR_HANDLER
Originator	Order Management Long Running where the plans are created.
Created On	The date of plan creation.
Changed On	Date of plan change.
Order ID	Order Id is the internal identifier of the Order generated by Order Management Server.

You can use a combination of search criteria to search for a specific plan object. For example, you can select a status of *COMPLETE* from the Status column and type the username in the Originator column to find all plans with a status of *COMPLETE* that were created by a particular user.

The search is case-sensitive.

- Click  to display the list of plans that match your search criteria. Select the plan to display its details.

Grid View

The Grid View is the default view of the Plans page. The Grid View displays:

- Details of the plan including custom headers.
- Details of Plan items, including custom headers, order line, process information and section level information.
- Milestone IDs along with their dependency information.

Plan View

TIBCO® Order Management

DashboardOrdersPlansRule ConfigActivity Log

Browse Plans

Search [x] [Created on] [between] to_date[11/01/2019 11:01:19] [Q] [Y] [Info] [x]

Order Ref ID	Plan ID	Description	Status	Created on	Changed on
9102_01	381	ProductDependsOn	EXECUTION	11/08/2019 01:40:19 UTC-8	11/08/2019 01:40:19 UTC-8
9091_01	375	AMD_TC01_CH_17	COMPLETE	11/08/2019 01:23:06 UTC-8	11/08/2019 01:24:37 UTC-8
OS_121	372	ProductDependsOn	CANCELLED	11/08/2019 01:08:13 UTC-8	11/08/2019 01:08:26 UTC-8
Order11	370	ProductDependsOn	COMPLETE	11/08/2019 00:55:55 UTC-8	11/08/2019 00:56:31 UTC-8
	366	Plan For Order11	COMPLETE	11/07/2019 03:19:18 UTC-8	11/07/2019 03:19:39 UTC-8
	364	Plan For Product	COMPLETE	11/07/2019 03:42:38 UTC-8	11/07/2019 03:42:38 UTC-8

Plan ID: 375(Plan View)

Hide Non-Execution Plan Item

Plan

Custom Headers

Plan ID

375

Description

AMD_TC01_CH_17

Originator

AFO

Created On

11/08/2019 01:23:06 UTC-8

Order ID

374

Order Ref ID

AMD_G01_83

Status

COMPLETE

Changed On

11/08/2019 01:24:37 UTC-8

Is Amendment?

true

Start Date

11/08/2019 01:23:06 UTC-8

Risk Region

NORMAL

Jeopardy Message

None

When you enable Order Management Server UI Grid View and Gantt View Pagination through configuration, you can see pagination in Grid View.

In Grid View, you can see Previous  and Next  icons for traversing within the plan-items.

Grid View

Browse Plans						
Order Ref ID	Plan ID	Description	Status	Originator	Created on	Changed on
AF-6411_02	295	Plan For AF-6411_02	EXECUTION	AFO	01/04/2016 14:35:34 UTC+5:30	01/04/2016 14:35:35 UTC+5:30
AF-6411_01	292	Plan For AF-6411_01	EXECUTION	AFO	01/04/2016 13:46:05 UTC+5:30	01/04/2016 13:46:45 UTC+5:30
Plan ID: 292(Plan View)						
<div> <div>Plan</div> <div>Custom Headers</div> <div> <div>Plan ID</div> <div>292</div> <div>Description</div> <div>Plan For AF-6411_01</div> <div>Originator</div> <div>AFO</div> <div>Created On</div> <div>01/04/2016 13:46:05 UTC+5:30</div> <div>Order ID</div> <div>291</div> <div>Order Ref ID</div> <div>AF-6411_01</div> <div>Status</div> <div>EXECUTION</div> <div>Changed On</div> <div>01/04/2016 13:48:45 UTC+5:30</div> <div>Is Amendment?</div> <div>true</div> <div>Start Date</div> <div>01/04/2016 13:46:06 UTC+5:30</div> <div>Risk Region</div> <div>NORMAL</div> <div>Jeopardy Message</div> <div>None</div> </div> </div>						

You can use a PlanItem display template for Grid View on the Order Management Server UI. The following are template variables you can use:

- PlanItemID
- PlanFragmentID
- ProductID
- Action
- Description

These template variables are case sensitive and have to be specified within curly braces ({}). {PlanItemID} must be included as a mandatory variable in the template. Different template variables can be combined with any type of delimiter along with PlanItemID. For example, {PlanFragmentID}|{PlanItemID}.

GANTT Chart View

You can view the orchestration of an execution plan by using the Gantt chart.

For more information, see [GANTT Chart](#).

Accessing Dependency View of Plan Items and their Status

The **Grid View** is the default view of the **Plans** page. You can switch to **Dependency View** to see the point dependencies between the plan items. The dependency arrows point toward the plan items on which it (plan item) is dependent.

Procedure

1. On the **Plans** tab in Order Management Server UI, from the plan list, select a plan and then click **Dependency View** icon.

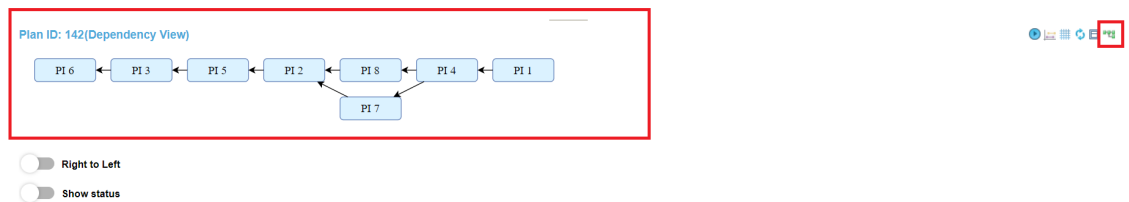
The plan items are shown in their default plan item execution flow direction, which is left to right.



The arrow direction shows the dependency between plan items and must not be mistaken for the plan item execution flow direction.

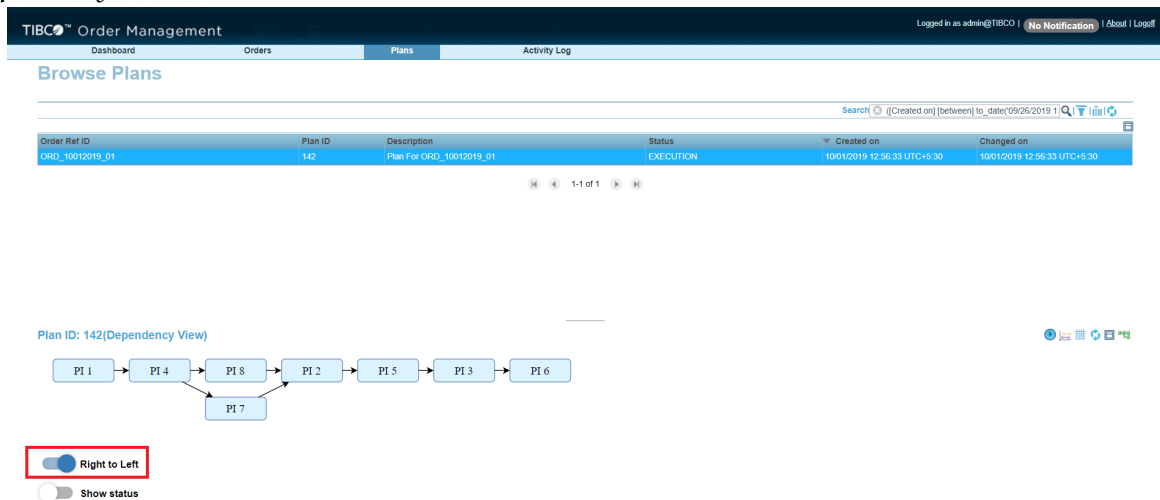
Dependency View

Order Ref ID	Plan ID	Description	Status	Created on	Changed on
ORD_10012019_01	142	Plan For ORD_10012019_01	EXECUTION	10/01/2019 12:56:33 UTC+5:30	10/01/2019 12:56:33 UTC+5:30



2. To change the plan item execution flow direction from right to left, click **Right to Left** direction switch.

Dependency View_direction switch



3. To view the current status, click **Show status** switch.

Dependency View_show status switch

The screenshot shows the TIBCO Order Management interface. The 'Plans' tab is selected, and the 'Dependency View' for Plan ID 142 is displayed. The plan items are arranged in a sequence: PI 6, PI 3, PI 5, PI 2, PI 8, PI 4, PI 1, and PI 7. The 'Show status' switch is enabled, and a legend is shown below the plan items. The legend includes the following status categories and their corresponding colors:

- Pending (Orange)
- Execution (Green)
- Complete (Blue)
- Suspended (Yellow)
- Cancelled (Red)
- Error (Red)
- Error Handler (Grey)

When the **Show status** switch is enabled, a legend shows the status associated with each plan items. You can identify the plan items by their plan item id written in the boxes representing the plan items.

Dependency View_legend

The screenshot shows the TIBCO Order Management interface. The 'Plans' tab is selected, and the 'Dependency View' for Plan ID 142 is displayed. The plan items are arranged in a sequence: PI 1, PI 4, PI 8, PI 2, PI 5, PI 3, PI 6, and PI 7. The 'Show status' switch is enabled, and a legend is shown below the plan items. The legend includes the following status categories and their corresponding colors:

- Pending (Orange)
- Execution (Green)
- Complete (Blue)
- Suspended (Yellow)
- Cancelled (Red)
- Error (Red)
- Error Handler (Grey)

- Click **refresh** icon to fetch the latest state of plan items, which is applied to the dependency view immediately.



You can drag the plan items to rearrange them according to the space available and the number of plan items in the plan. However, cannot change the dependency.

Jeopardy Rule Configuration

This section explains how you can configure jeopardy rules for jeopardy management to send notification, or invoke web services to perform consequential actions. For details on the Jeopardy Management System, see *TIBCO® Order Management Long Running Concepts and Architecture*.

The Jeopardy management is a process of monitoring execution of plan to ensure they are completed according to a SLA associated with the plan items and plan. Plans are based on a time schedule. When execution is predicted to violate the schedule, the system raises an event and the user can configure rules to automatically send notification or perform consequential action by invoking web service.

The Jeopardy Rule Configuration allows you to configure rules for Jeopardy Management.

Refer to the *TIBCO® Order Management Long Running Concepts and Architecture* document for more details on the Jeopardy Management System.

The Jeopardy Management System raises the following two types of jeopardy events:

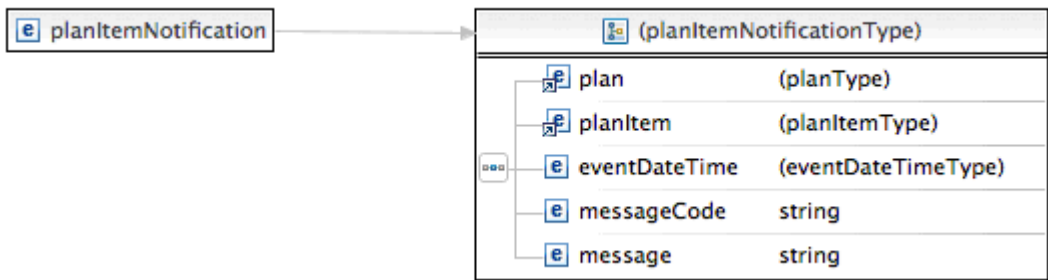
- Plan Item Jeopardy
- Plan Jeopardy

Plan Item Jeopardy

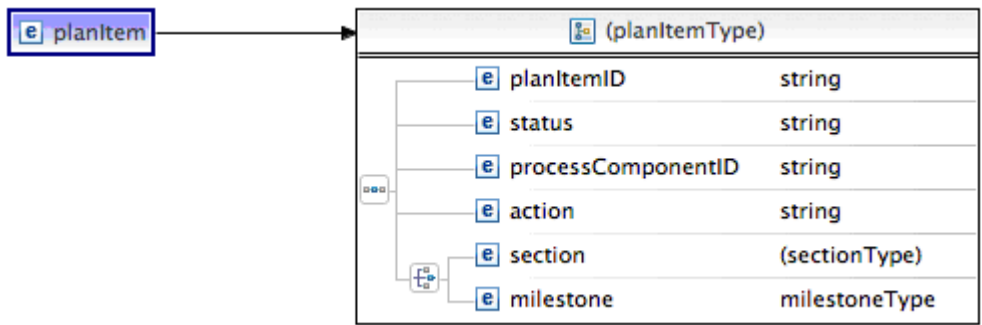
The plan item jeopardy occurs when plan items exceed or predicted to exceed following thresholds specified in a plan fragment model of a process component:

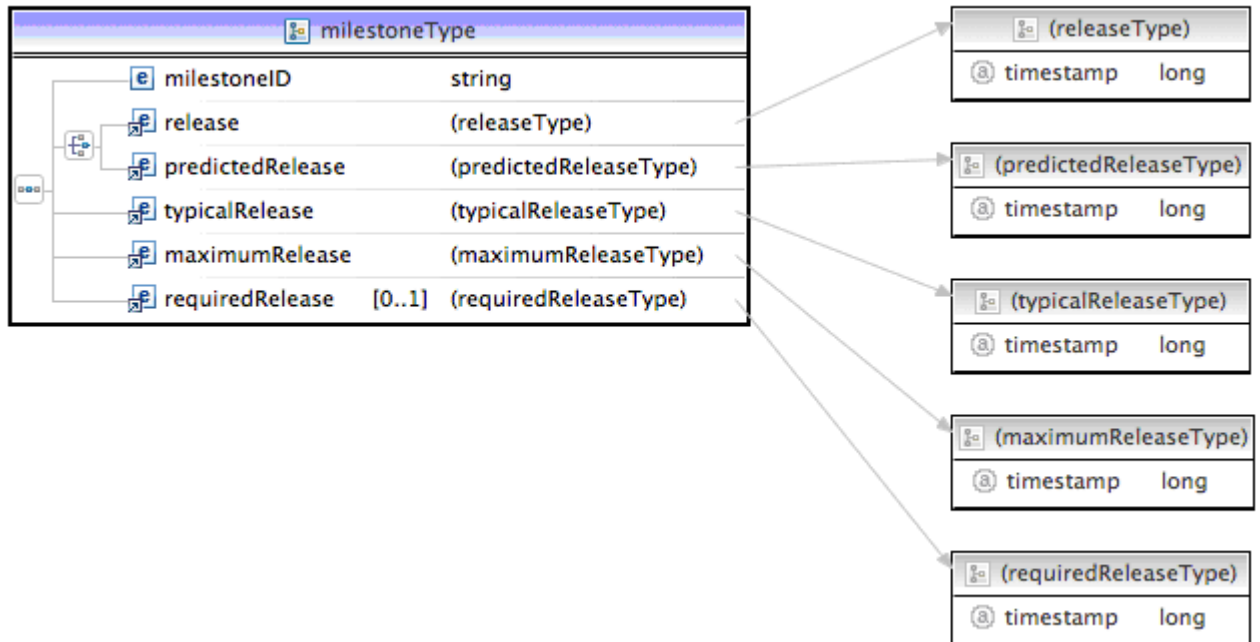
- Typical duration
- Maximum duration

The visual representation of the jeopardy event payload for the PlanItem Jeopardy event XML object is as follows:



The following is the visual representation of the plan item:





The following table lists all the jeopardy events that Jeopardy Management System raises for the Plan Item jeopardy.

Plan Item Jeopardy Conditions	Description
AFF-JM-PLANITEM-0100	Plan item has exceeded typical duration
AFF-JM-PLANITEM-0110	Plan item has exceeded maximum duration
AFF-JM-PLANITEM-0120	Plan item has exceeded required start
AFF-JM-PLANITEM-0200	Plan item start is predicted to exceed required start and is increasing
AFF-JM-PLANITEM-0210	Plan item start is predicted to exceed required start and is decreasing
AFF-JM-PLANITEM-0220	Plan item is no longer predicted to exceed required start

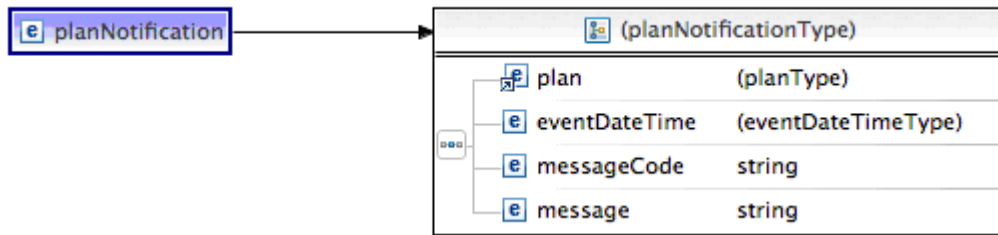
Plan Jeopardy

The plan jeopardy occurs when the execution plan exceeds or is predicted to exceed the following thresholds:

- Typical Duration
- Maximum Duration

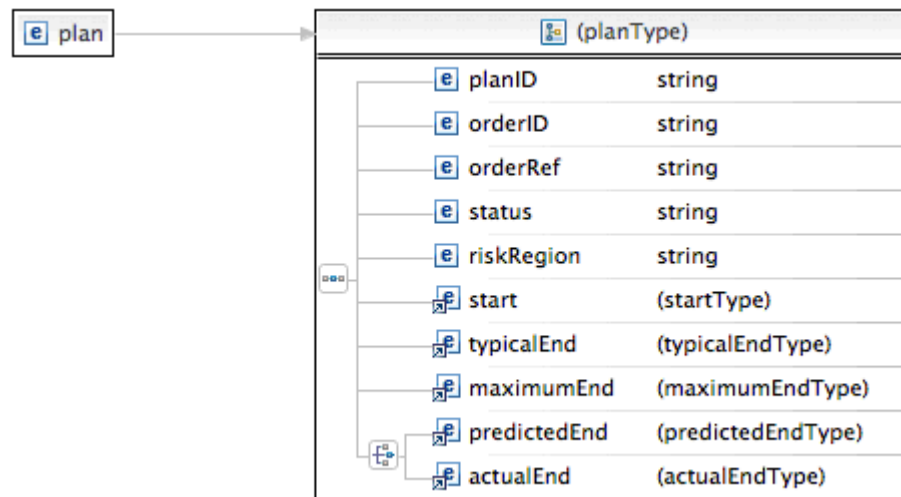
Jeopardy event payload for the Plan Jeopardy is an XML object and shown as follows:

Plan Notification Event



The plan element is visually represented as follows:

Plan Element



Following table lists all the jeopardy events that Jeopardy Management System can raise for the Plan Item jeopardy:

Plan Jeopardy Conditions	Description
AFF-JM-PLAN-0100	Plan has exceeded typical duration
AFF-JM-PLAN-0110	Plan has exceeded maximum duration
AFF-JM-PLAN-0120	Plan has exceeded out of scope threshold
AFF-JM-PLAN-0200	Plan is predicted to exceed typical duration and is increasing
AFF-JM-PLAN-0210	Plan is predicted to exceed typical duration and is decreasing
AFF-JM-PLAN-0220	Plan is no longer predicted to exceed typical duration
AFF-JM-PLAN-0230	Plan is predicted to exceed maximum duration and is increasing
AFF-JM-PLAN-0240	Plan is predicted to exceed maximum duration and is decreasing
AFF-JM-PLAN-0250	Plan is no longer predicted to exceed maximum duration

Jeopardy event rules can be configured in the Order Management UI to either send notification or invoke a web service.

You can use Rule Configuration to do the following:

- Add rule
- Update rule
- Delete rule
- Suspend rule
- Activate rule

The following table lists the rule header information:

General Details	Condition	Action
Name, description, status, Event Group, Event Type, etc.	Criterion to be monitored for Jeopardy Management	Action to be taken if condition matches

Adding and Configuring Rule

To add a rule, perform the following steps:

1. On the Order Management UI, click **Rule Config**
2. Click **Add**.

Adding Rule



Only active rules are used to monitor jeopardy condition.

3. Edit the following rule attributes:

Name	Description	Run on Error /Run on Failure
rule identifier	description of the rule	You can set them to <code>true</code> if you want the system to execute in spite of a failure or error in an action. A failure is the same thing as an error, except it represents communication failure to the configured end point.

Rule Attributes

Rule Name: rule1

Name	rule1	Description	rule1
Status	active	Status Changed	
Created On		Run on Error	<input type="checkbox"/>
Run on Failure	<input type="checkbox"/>	Event Group	Jeopardy
Event Type	Pan_jenardy		



Optionally, you might add condition to the rule to filter specific events with attribute values in event payload.

Adding Condition to Rule

1. On the Order Management UI, click **Condition**. The Condition Editor tab is displayed.

Adding Condition to Rule



2. Click **Edit Attribute** in Condition Editor tab to add or edit conditions. Condition Builder window is displayed as a pop, which can be used to configure conditions for this rule.

Edit Attribute



3. A criteria has the Left Operand, Operator and Right Operand, as displayed:

Left Operand	Operator	Right Operand
The left side of a quantity upon which a criteria operation is performed	Performs calculations on the operands in a query or an expression	The right side of a quantity upon which a criteria operation is performed

Select Left Operand, Operator and Right Operand from the Condition Builder to create a condition. Left and Right Operand can be selected from a predefined list of attributes or utility methods. Similarly, Operator can also be selected from a predefined list of attributes depending on the selection of Left Operand.

Selecting Left Operand

To select Left Operand click the **Edit** icon besides Left Operand in the Condition section. A predefined list of Attributes/utility methods is displayed on right side in Attributes section to select as Left Operand. An attribute or utility method from this available list can be selected.

Using attributes as Left Operand

Select an attribute from the available list and click the **Map** icon to assign this value as left operand. Please refer Plan Jeopardy and Plan Item jeopardy section for list of attributes which can be selected as Left Operand.

Condition Builder

Condition Builder

Condition Expression

Input Mapping

Condition

- Left Operand
- Operator
- Right Operand

Attributes (Select the Node)

- event
 - Plan Notification
 - orderId
 - orderRef
 - planID
 - status
 - riskRegion
 - actualEnd
 - maximumEnd
 - predictedEnd
 - typicalEnd
 - Message Code
 - Message Desc
- utility

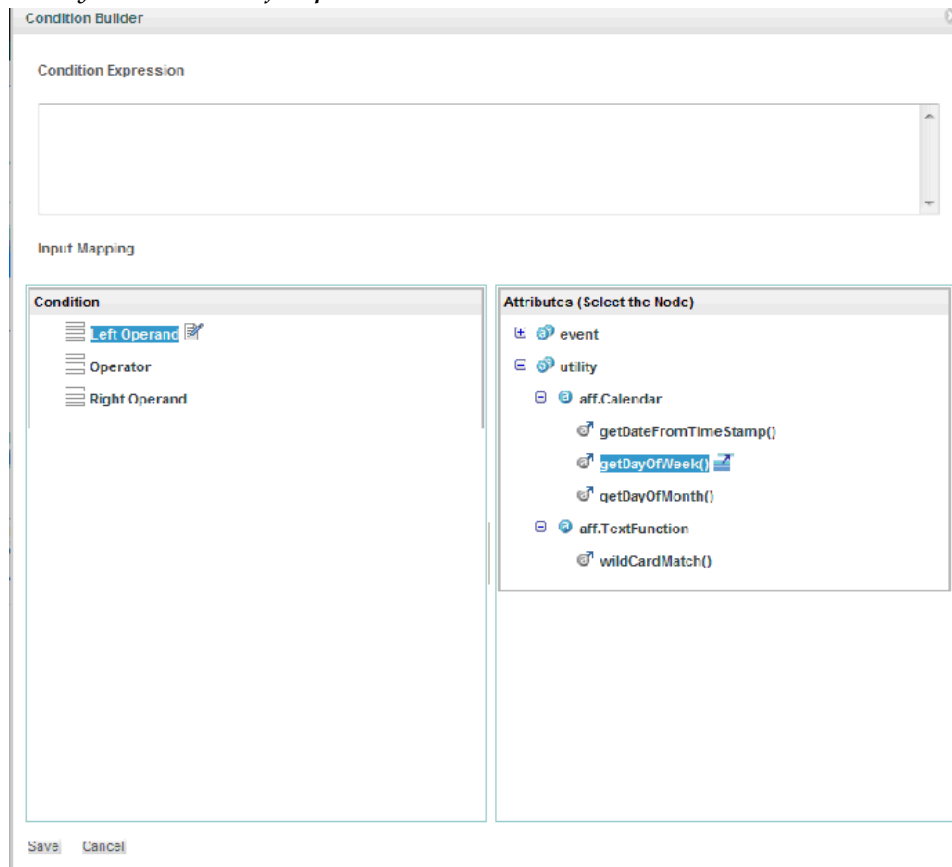
Save Cancel

Using Utility methods as Left Operand

List of utility methods is displayed in Attributes section under utility node. Select a utility method from this list and click **Map** icon to assign the return value of this method as Left Operand.

Following table shows the utility methods listed under utility node in Attributes section:

Utility methods as Left Operand



Values to the input parameters of the method can be assigned using the **Edit** icon appearing besides method parameter. When you click **Edit**, a list of attributes appears on right side and value can be assigned by selecting any attribute. Use Data Input to assign any input value for an input parameter of the method.

Utility Methods

Condition Builder

Condition Expression

Input Mapping

Condition

- Left Operand: Calendar.getDayOfWeek()
 - timezone: America/Chicago
 - country: US
 - language: en
 - timestamp: 0
- Operator
- Right Operand

Name	Description
America/Chicago	CST - Central Daylight Time
America/Los_Angeles	PST - Pacific Daylight Time
America/New_York	EST - Eastern Daylight Time
Asia/Kolkata	IST - India Standard Time
Australia/Melbourne	EST - Eastern Standard Time (Victoria)

1-5 of 5

Save Cancel

The following table shows supported utility methods with description:

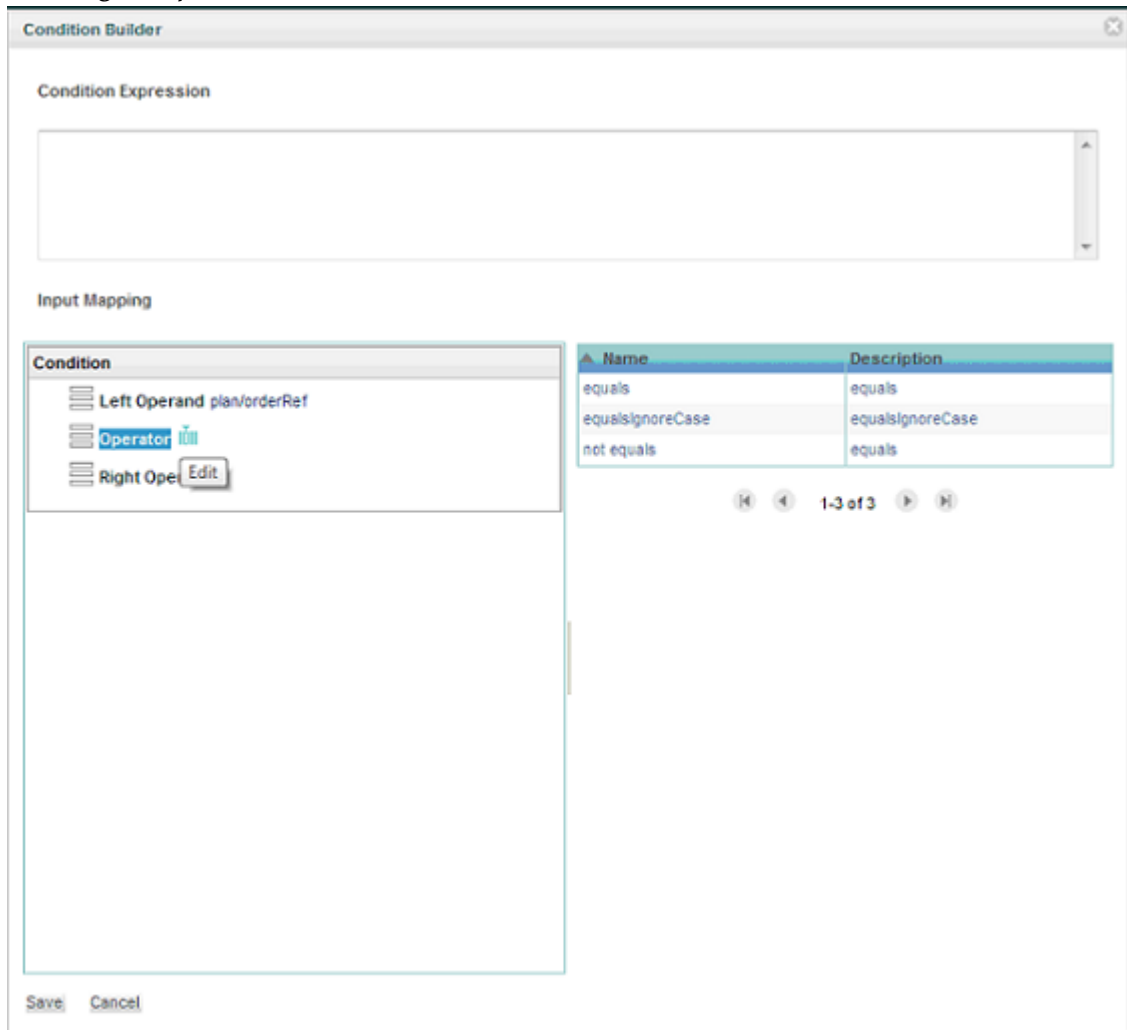
Method Signature	Description
<pre>public Boolean wildCardMatch(java.lang.String text, java.lang.String pattern)</pre>	Test if the pattern matches with given text. Parameters: text - java.lang.String to be searched in an effort to find pattern pattern - character or java.lang.String of one or more characters to be searched for Returns: Returns value true if the given text matches the pattern and false otherwise
<pre>public java.util.Date getDateFromTimeStamp(java.lang.Long inputTimeStamp)</pre>	Indicating java.util.Date for specified timestamp in milliseconds. Parameters: inputTimeStamp - milliseconds value as java.lang.Long Returns: java.util.Date object and initializes it to represent the specified number of milliseconds.
<pre>public java.lang.Integer getDayOfWeek(java.lang.String timezone, java.lang.String country, java.lang.String language, java.lang.Long timestamp)</pre>	Indicating the day of week for specified timestamp in milliseconds Parameters: timezone - the timezone (Example: 'America/Chicago') country - the country (Example: 'US') language - the language (Example: 'en') timestamp - the timestamp as milliseconds Returns: Day of the week as an Integer value for the timestamp value using the time zone, country and language specified

Method Signature	Description
<pre>public java.lang.Integer getDayOfMonth(java.lang.String timezone, java.lang.String country, java.lang.String language, java.lang.Long timestamp)</pre>	<p>Indicating the day of month for specified timestamp in milliseconds Parameters: timezone -the timezone (Example: 'America/Chicago') country - the country (Example: 'US') language - the language (Example: 'en') timestamp - the timestamp as milliseconds Returns: Day of the month as an Integer value for the timestamp value using the timezone, country and language specified.</p>

Select an Operator

To select an Operator, click Edit icon besides Operator in condition section a list of operators with description is displayed on right side. Select any operator to assign Operator value.

Selecting an Operator



Select Right Operand

To select Right Operand click **Edit** button besides Right Operand in Condition section. A list of attributes appears on the right side in the Attributes section to select as Right Operand. Select any attribute from the available list or input a value by selecting **Data Input** from Attributes section and click **Map** icon to assign this value as the Right Operand.

Selecting Right Operand

The screenshot shows the 'Condition Builder' dialog box. At the top, there is a 'Condition Expression' text area. Below it is the 'Input Mapping' section, which is divided into two panes. The left pane, titled 'Condition', contains a list of items: 'Left Operand plan/orderRef', 'Operator equals', and 'Right Operand' (which is highlighted in blue). The right pane, titled 'Attributes (Select the Node)', contains a 'Data Input' section with a text field containing 'sample_order_ref' and a 'Map' button below it. At the bottom of the dialog are 'Save' and 'Cancel' buttons.

4. Click **Generate Groovy Method** to verify the criteria and generate a corresponding expression

Groovy Method

The screenshot shows a 'Condition Builder' window. It has a 'Condition Expression' section at the top with a large text area. Below it is an 'Input Mapping' section. Inside 'Input Mapping', there is a 'Condition' box containing three items: 'Left Operand plan/orderRef', 'Operator equals', and 'Right Operand sample_order_ref'. A 'Generate Groovy Method.' button is located to the right of the 'Right Operand' field. At the bottom of the window are 'Save' and 'Cancel' buttons.

An expression is generated using the selected Left Operand, Operator, Right Operand and it is displayed in the Condition Expression section.

Expression

Condition Builder

Script Generated successfully

Condition Expression

```
(eventObject.getPlan().getOrderRef().equals(new String("sample_order_ref")))
```

Input Mapping

Condition

- Left Operand plan/orderRef
- Operator equals
- Right Operand sample_order_ref

Save Cancel

5. Click **Save** to add condition to Condition Editor.

6. To add more condition click **Add** icon as shown below. To add a nested condition click **Add Nested** icon. **Match All** is used if all the conditions have to be evaluated true for this rule to execute. **Match Any** is used if any of the specified conditions have to be evaluated true for this rule to execute.

Condition Editor

Condition Editor Expression

Match All

Match All

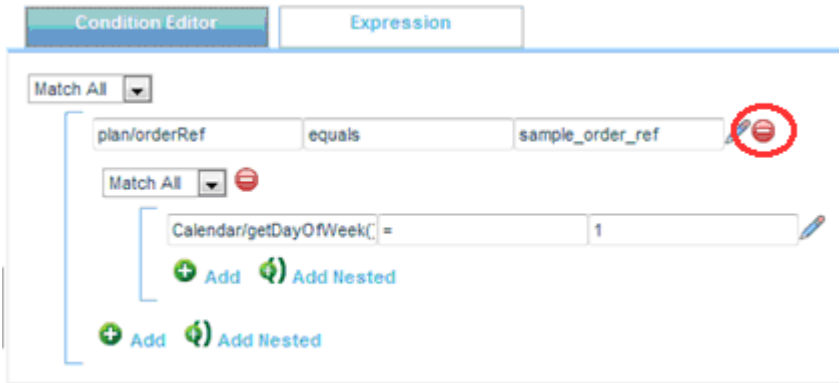
Match Any

derRef equals sample_order_ref

Add Add Nested

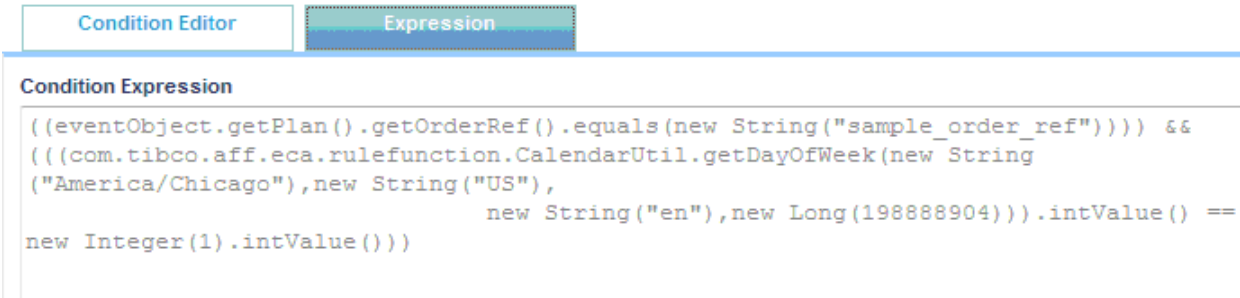
Conditions in the Condition Editor are shown as follows: You can delete a Condition or nested condition from the Condition Editor using the Delete icon.

Deleting Condition



Groovy script generated from a defined condition is shown in the Expression tab as follows.

Groovy script generated from a defined condition



Rule Actions

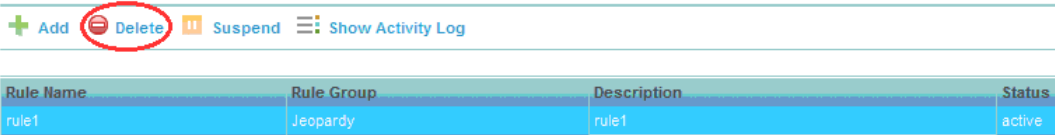
The rule actions are as follows:

Delete Rule

- 1. Select an existing rule to delete.
- 2. Click **Delete**.

Delete Rule

Rule Configuration



Suspend Rule

- 1. Select an existing rule to suspend.
- 2. Click **Suspend**.

Suspend Rule

Rule Configuration

+ Add - Delete **Suspend** Show Activity Log

Rule Name	Rule Group	Description	Status
rule1	Jeopardy	rule1	active

Activate Rule

1. Select an existing rule to activate.
2. Click **Activate**.

Activate Rule

Rule Configuration

+ Add - Delete Show Activity Log **Activate**

Rule Name	Rule Group	Description	Status
rule1	Jeopardy	rule1	inactive

Configure Actions

The Jeopardy Management System (JeOMS) supports the following two types of consequential actions for jeopardy events:

1. Alert or Notification - Supports sending notification to the following end points:
 - a. SMTP - notification to specific email accounts.
 - b. JMS - JMS (Java Message Service) messages to JMS queue or topic.
 - c. Tibbr - notification message to Tibbr subject.
 - d. File - log notification messages.
2. Service Invocation - Supports invoking web services, which uses WSDL 1.1 specification.

Jeopardy Management System also allows you to specify dampening criteria to action. Give the dampening (*or flow control*) criteria to action on how often to send notifications or invoke service for the same jeopardy event.

The frequency for sending alerts depends on the expected behavior of the process component. There are several frequency settings:

- Every time the jeopardy rule condition evaluates to `true`. This sends the most alerts. It's the default setting if dampening criteria is not specified.
- Every X times the condition is `true`. This sets a number of times that the condition has to occur before an action is executed. This offsets any occasional spikes in executing plan item in process component or in the fulfillment system. To add this criteria set following values:

True Count	Evaluation Count	Notification Count
X	Y	1

- Every X times the condition is true in Y evaluations. This sets a number of times that the condition has to occur in a given number of monitoring evaluations cycles before an action is executed.

True Count	Evaluation Count	Notification Count
X	Y	1

- Every X times the condition is true in Y time period.

True Count	Evaluation Count	Notification Count	Time Value	Time Unit
X	Y	1	Y	Period ((SECOND/MINUTE/HOUR/DAY/WEEK/MONTH))


- No more than X notification in Y time period

True Count	Evaluation Count	Notification Count	Time Value	Time Unit
X	Y	1	Y	Period ((SECOND/MINUTE/HOUR/DAY/WEEK/MONTH))

Webservice Configuration

Add Action

1. Click Add Action.
2. Edit the following action attributes:

Name	Description	Notification type	Dampening Criteria (Optional)
action identifier	action description	Select <i>service</i>	<p>specify any dampening criteria to the action</p> <p>Select Service Parameter tab</p> <p>Enter the WSDL location</p> <div>  <p>If WSDL location is valid, Service and Method are populated with respective values. Select a service name and Method name to be invoked</p> </div>

3. Enter the parameter information.
4. Click the **Template** tab. It shows the web service request parameters in an XML form. This request parameter can be configured using Template Builder. Click the icon in Template tab to use Template Builder and a popup would appear with a predefined list of parameters.

The web service request parameters in an XML form

The screenshot shows the 'Template Builder' window with two main sections: 'Body' and 'Input Mapping'.

Body: Contains an XML snippet for a CancelOrderRequest. The XML is as follows:

```
<ord:CancelOrderRequest xmlns:ord="http://www.tibco.com/aff/orderservice" xmlns:ord1="http://www.tibco.com/aff/order"
xmlns:com="http://www.tibco.com/aff/commontypes" xmlns:fault="http://www.tibco.com/aff/orderservice/result"
ExternalBusinessTransactionID="?">
  <ord:orderID>${plan/orderID}</ord:orderID>
  <ord:orderRef></ord:orderRef>
```

Input Mapping: A list of attributes is shown under the 'event' category. The 'orderRef' attribute is highlighted with a blue selection box.

- event
 - Plan Notification
 - orderID
 - orderRef
 - planID
 - status
 - riskRegion
 - actualEnd
 - maximumEnd
 - predictedEnd
 - typicalEnd
 - Message Code
 - Message Desc

At the bottom of the window are 'Save' and 'Cancel' buttons.

Configure input parameters using the list of attributes and this would be replaced with actual values when invoking the web service.

5. Click **Save**. Webservice request parameters would be saved and display in the Template tab.

Delete Action

- Select an existing action.
- Click **Remove Action**.

Notification Action Configuration

You can receive messages only on the given queues to check and read the notification. The receiver count is 0 as there is no receiver. Also, protocols such as tibbr, jms, smtp, and file options are available to get the notification messages only.



To receive the notifications, open the `profiles.properties` file in the config folder for omsui and omsServer and set the value as `com.tibco.fom.jeoms.deployMode=JEOMS_colocated`.

The notification action configuration is as follows:

Add Action

1. Click **Add Action**.
2. Edit the following action attributes:

Action Attributes

Rule Name: rule1

[Add Action](#)
[Remove Action](#)
[Test Action](#)

[Rule](#)
[Condition](#)
[Action](#)

[Dampening Criteria](#)

Name: action1
 Notification Type: notification
 Dampening Criteria: 0
 True Count: 0
 Notification Count: 0
 Unit: MINUTE

Description:
 Evaluation Count: 0
 Time Value: 0

Name	Description	Notification type	Dampening Criteria (Optional)
action identifier	action description	Select <i>notification</i>	specify any dampening criteria to the action

3. Select Notification Parameter tab
4. Select the protocol
5. Enter end point URI. The following table shows supported protocols with syntax and example

End Point Type	URI Syntax	Example
JMS Queue	jms:queue:<queue name>	jms:queue:jeopardy.notification
JMS Topic	jms:topic:<topic name>	jms:topic:jeopardy.notification
Tibbr	tibbr://<hostname:port>	tibbr://xyz.tibbr.com
SMTP	smtp://<hostname:port>	smtp://smtp.xyz.com:25
SMTPS	smtps://<hostname:port>	smtps://smtp.xyz.com:465
File	file://<filepath>	file:///opt/notification

6. Select message type for the notification. The following table shows protocol and supported out message type

Protocol	OutBound Message Type
JMS	plainStringOutMessage
	jsonMessage
	javabean
	xmlOutMessage
Tibbr	plainStringOutMessage

Protocol	OutBound Message Type
SMTP	plainStringOutMessage
	xmlOutMessage
SMTPS	plainStringOutMessage
	xmlOutMessage
File	plainStringOutMessage
	jsonMessage
	javabean
	xmlOutMessage

The following table shows the supported out message type:

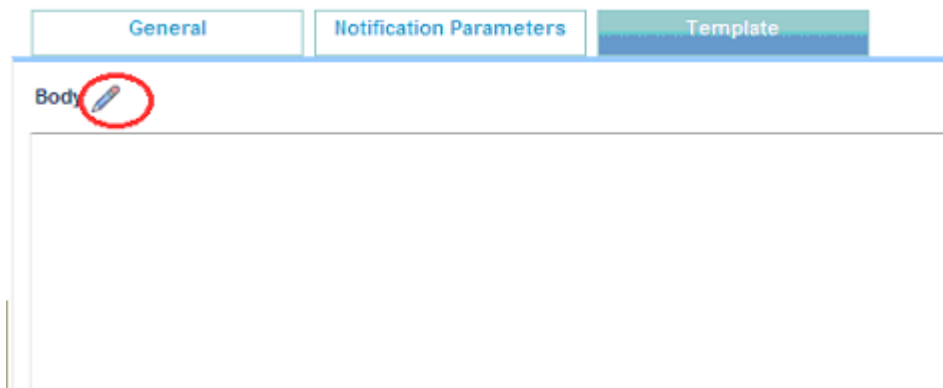
OutBound Message Type	Description
plainStringOutMessage	The notification message in plain string
jsonMessage	The notification message in the form of a json object
javabean	The notification message in the form of a javabean object
xmlOutMessage	The notification message in the form of a xml string

7. Edit the parameters information for the selected protocol. The following table shows the protocol and the respective parameters configuration required:

Protocol	Parameters	Description
JMS	No parameter required	
Tibbr	clientKey	Client Key used to post messages
	Password	Password of the user
	Subject	Subject used to post messages
	username	User name of the user
SMTP	CC	CC to be used while sending e-mail
	debugMode	Specify debug mode
	From	Sender of the email
	Password	Password of the user
	Subject	Subject used in the email
	To	Receiver of the email
	username	User name of the user
SMTPS	CC	CC to be used while sending e-mail
	debugMode	Specify debug mode
	From	Sender of the email
	Password	Password of the user
	Subject	Subject used in the email
	To	Receiver of the email
	username	User name of the user
File	filename	File name used for sending notification.

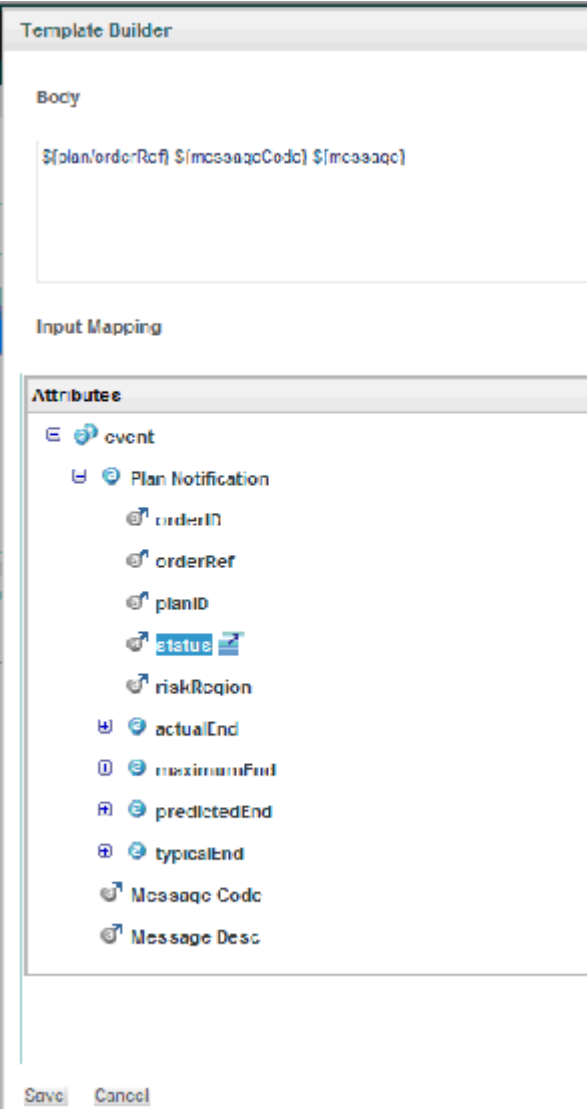
8. Edit Template to configure `plainStringOutMessage` content using Template Builder. Click the icon mentioned in the following image in the Template tab to open the Template Builder window:

Template



9. The following is the Template Builder window, which has all the notifications parameters listed to configure a template for notification messages.

Template Builder Notifications



- 10. Click **Save** to save the template content in action.
 - 11. Click Test Action to verify the action created is valid. If there is no error, the Test Connection Successful message is displayed.
- If there is any error respective error message is displayed. The following table shows error messages, which are displayed:

End Point Type	Error Message	Description
JMS	Invalid URL	URI is null or not provided
	Protocol not supported	URI pattern is different than the syntax mentioned in point 5 Ex. abc, abc:topic:notification
	Invalid protocol scheme	URI does not start with 'jms'
	Invalid JMS destination type	URI does not have destination type as queue or topic
	Unable to resolve the destination :<destination name>	Not able to find specified queue or topic name
Tibbr	Invalid URL	URI is null or not provided
	Protocol not supported	URI pattern is different than the syntax mentioned in point 5 Ex. abc://xyz.tibbr.com
	Invalid protocol scheme	URI does not start with 'tibbr'
	Valid Tibbr subject must be specified	Subject is not specified
	Unknown host exception	Tibbr host is not known
	Unable to parse host response	Not able to read the response from tibbr
	Invalid user credential	No user found for details provided
	User Name or Password you entered is incorrect	Cannot login with the user details provided
	Tibbr Subject not found	Subject specified is not available
	clientKey must be specified	clientKey is missing

End Point Type	Error Message	Description
SMTP	Protocol not supported	URI pattern is different than the syntax mentioned in point 5 Ex. abc://smtp.xyz.com:25
	Invalid URL	URI is null or not provided
	Invalid protocol scheme	URI cannot be reached using smtp
	Port must be a valid number	Port mentioned in URI is not valid
	User credential is missing	User name or password is missing
	Invalid email id	Email id is not valid
	Unable to connect to the end point	Not able to connect the URI using the user id and password
SMTPS	Protocol not supported	If URI pattern is different from the syntax mentioned in point 5 Ex. abc://smtp.xyz.com:25
	Invalid URL	If URI is null or not provided
	Invalid protocol scheme	URI cannot be reached using smtp
	Port must be a valid number	Port mentioned in URI is not a number
	User credential is missing	User name or password is missing
	Invalid email id	Email id is not valid
	Unable to connect to the end point	Not able to connect the URI using the user id and password
File	Protocol not supported	If URI pattern is different from the syntax mentioned in point 5 Ex. abc:///opt/notification
	Invalid URL	If URI is null or not provided
	Invalid protocol scheme	If URI does not start with file:// Ex. file:/jeopardy
	File end point cannot be null	If URI is null
	File path not found	If URI mentioned does not exist
	Invalid file name	If a file cannot be created with specified name
	file path must be a valid directory	If URI mentioned is not a valid directory
	No write permission to create file in the directory	If file cannot be created/written in the file path mentioned as URI
	Minimum 1GB free space is required for jeopardy logging	Less than 1 GB space is available to store notifications

Hot Deployment

Rules configured using Order Management Server UI are hot deployable. Rules configured through Order Management Server UI are published to all instances of Jeopardy Management System in cluster.

GANTT Chart

You can view the orchestration of an execution plan by using the Gantt chart.

The GANTT chart displays the following objects as a graphical representation on the Order Management UI:

- Plan Summary Task
- Plan Item Summary Task
- Milestones
- Sections
- Typical duration constraint for the section
- Maximum Duration constraint for the section
- Forecast the Plan in execution (Prediction)
- Show Critical path
- Time dependencies and point dependencies
- Different regions in which a plan can be – Typical Duration, Risk Threshold, Maximum Duration, Contingency Buffer, Out-Of-Specification region
- Show Jeopardy conditions
- You can view the GANTT chart in different time scale levels (*zoom levels*), starting from weeks till the milliseconds level

When you enable the Order Management Server UI Grid View and GANTT View Pagination through configuration, and set a page size, the next page is available when you scroll down the GANTT chart. For example, when you scroll down the current page limit, the next page is automatically loaded in the GANTT chart.

Jeopardy Management System



When you opt for deploying Jeopardy Management System again, you must publish the plan-fragment models again. Plans which are in execution are not part of the Jeopardy cycle but the UI can enrich the plan and show the GANTT chart.

When Jeopardy Management System is deployed, it enriches the plan and this enriched plan is used by the GANTT chart to plot some of the information. You can choose not to deploy Jeopardy Management System and still view the GANTT chart.

When Jeopardy Management System is not deployed, the GANTT chart does not show the following:

1. Real time predictions for the GANTT chart of the plan. Predicted GANTT chart for the plan is based on the typical duration set in plan-fragment model.
2. Background color of the GANTT chart for plan in different states.
Grey is used as a background color for the plan when Jeopardy Management System is not deployed.
3. Jeopardy Messages/Notifications on the tool tip of GANTT chart items.


Viewing a GANTT Chart

The GANTT chart displays the execution plan as a graphical representation.

Procedure

1. Navigate to the Plans page.
2. Select any plan.



3. Click the GANTT chart icon  to view a diagram of the selected plan.

For details on the GANTT chart, see [GANTT Chart Details](#).

GANTT Chart Components

The GANTT chart is made up of several key components.

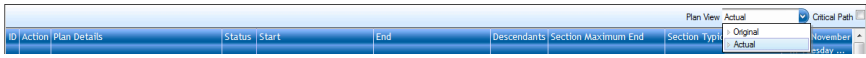

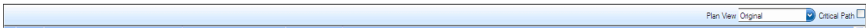
The following are the components of the GANTT chart:

- Grid Header
- Grid
- Top and Bottom Toolbars
- GANTT Header
- GANTT diagram

Top Tool Bar

The top tool bar changes according to the plan status.

The following table shows the GANTT charts depending upon the plan status:

Plan Status	GANTT Chart
COMPLETE or CANCELLED	
PENDING and EXECUTION	
SUSPENDED	

The top tool bar has the following two display options:

- **Plan View:** This combo-box has different values based on the execution plan status.
 - Plan Status COMPLETE or CANCELLED: Shows the GANTT with Actual & Original view.
 - Plan Status EXECUTING or PENDING: Shows the GANTT with Predicted & Original view.
 - Plan Status SUSPENDED: Shows the GANTT with Original view.

Different execution plan views shown in GANTT chart

Actual view: Shows the plan, plan items, sections & milestone with actual timestamp value.

Predicted view: Plan item with status as EXECUTING or PENDING are shown with predicted timestamp value and plan items with the COMPLETE status is shown with actual timestamp value.

Original view: Shows plan, plan items, sections, and milestone with typical timestamp value.

- **Critical Path:** This check-box shows the critical path with the specified plan view.

Bottom Tool Bar

The bottom tool bar has the following options:

- **Zoom Options:** Allows you to select the zoom level. The GANTT diagram changes according to the selected zoom level. You can select the following zoom options:
 - Week
 - Days

- Hours
- Minutes
- Seconds
- Milliseconds

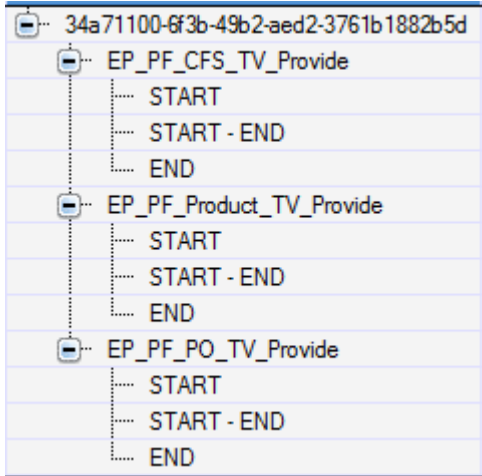
The Zoom options are available for the user according to the difference between plan start date and plan end date. For example, if the difference between plan start date and plan end date is more than 1 hour but less than 24 hours, different zoom levels available are - Hours, Minutes, Seconds, Milliseconds.



- **Auto-fit:** Allows you to fit the Gantt chart to the best possible zoom level. The GANTT diagram changes accordingly.
- **Help:** Shows the color coding for the GANTT chart.

Grid Header

The Grid Header component shows the ID, Action, Plan Details, Status, START, END, Decedents, Duration Constraint and Description for each grid column




The following table shows the constituents of a Grid Header:



ID	Shows numeric values starting from 1...n.										
Plan Details	Shows the following Plan information: <table border="1"> <thead> <tr> <th>Type</th><th>Convention</th></tr> </thead> <tbody> <tr> <td>Plan</td><td>Plan ID according to the generated plan.</td></tr> <tr> <td>Plan Item</td><td>Plan Item description from the plan.</td></tr> <tr> <td>Milestone</td><td>Milestone ID. For Example, <i>START</i> or <i>END</i>.</td></tr> <tr> <td>Sections</td><td>Merged Milestone IDs with "-" as delimiter. For example <i>START-END</i>.</td></tr> </tbody> </table> 	Type	Convention	Plan	Plan ID according to the generated plan.	Plan Item	Plan Item description from the plan.	Milestone	Milestone ID. For Example, <i>START</i> or <i>END</i> .	Sections	Merged Milestone IDs with "-" as delimiter. For example <i>START-END</i> .
Type	Convention										
Plan	Plan ID according to the generated plan.										
Plan Item	Plan Item description from the plan.										
Milestone	Milestone ID. For Example, <i>START</i> or <i>END</i> .										
Sections	Merged Milestone IDs with "-" as delimiter. For example <i>START-END</i> .										
Status	All the status icons used in this column are similar to icons used in the Plan tab grid level.										

START	<p>The start time for each Plan Details.</p> <p> The value is in the MM/dd/yyyy HH:mm:ss z format (z stands for time zone offset). You can configure this value through the Configurator.</p>
END	<p>The end time for Plan, Plan Item & Section.</p> <p> The value is in the MM/dd/yyyy HH:mm:ss z format (z stands for time zone offset). You can configure this value through the Configurator.</p>
Descendants	The ID value for the next dependent milestone, if any.
Section Maximum End	The Date format value showing maximum end value at the section level.
Section Typical End	The Date format value showing typical end value at the section level.
Action	This column contains icons. Clicking icons details for that particular row are shown in a popup window. For example, if you click the icon in action column at plan level, popup appears with plan details. If you click the icon in action column at plan item level so popup is shown with plan item details and likewise.

GANTT Chart Diagram

The following table describes the different GANTT chart components:

Component	Description
Plan Summary Task Bar	Indicates the entire plan, from start till end.
Plan Item Summary Task Bar	Indicates the plan item covering all the milestones and sections of a plan item.
Milestone	Represented by a grey diamond.
Section	Indicates the START and END milestones part.
Duration	<p>The two forecast types are shown at the section level:</p> <ul style="list-style-type: none"> Typical Duration  . Maximum Duration  .
Critical Path	Indicates the path with maximum time duration. It is represented by the section bar with red border  . When you click the critical path option check-box on the top tool bar, only the critical path is shown with red border sections.
Tool tip	Shows the details of the respective GANTT item when you place the cursor over any GANTT chart items, for example, plan, plan item, section, milestone, typical or maximum duration icons.

Component	Description
Time and Point Dependencies	<p>Time dependency: Represented by the  icon at the section level, if a section has started on the specified time dependency. If section start time has missed the specified time dependency, it is represented by the .</p> <p>Point dependency: Represented by connecting arrow. This is a unidirectional arrow which connects two milestones.</p>

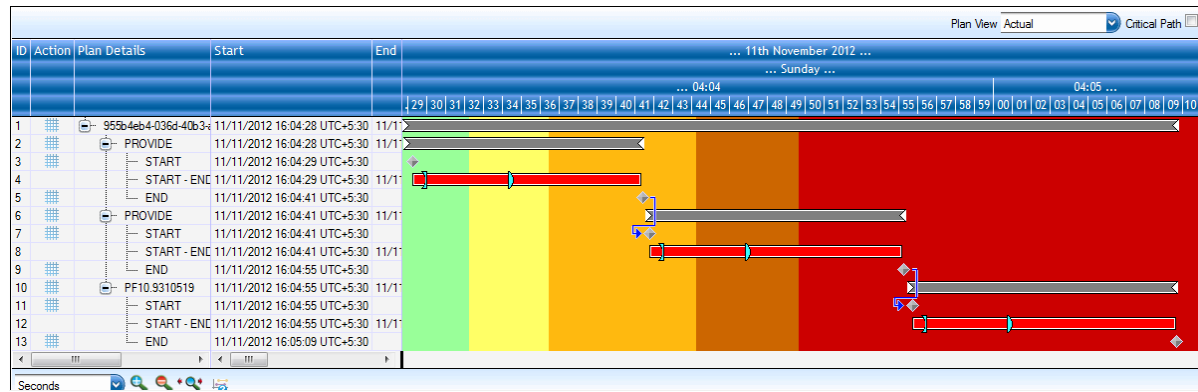
GANTT Chart Details

The different color schemes used in the GANTT chart to visually represent the status of the execution plan and plan item sections are as follows:

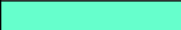




GANTT Chart Background Colors

The background color describes the status of the plan. The following figure displays the background color codes representing the execution plan in different states:

GANTT Chart Background Colors



Each color in the GANTT chart has its own significance. The following table describes the GANTT chart background colors:

Color	Description
	From the start of plan till the end of typical duration.
	From the end of typical duration till the start of risk region threshold.
	From the end of risk region till the start of maximum duration.
	From the end of maximum duration till the start of out-of-specification threshold.
	Out-of-specification region.

GANTT Chart Section Color

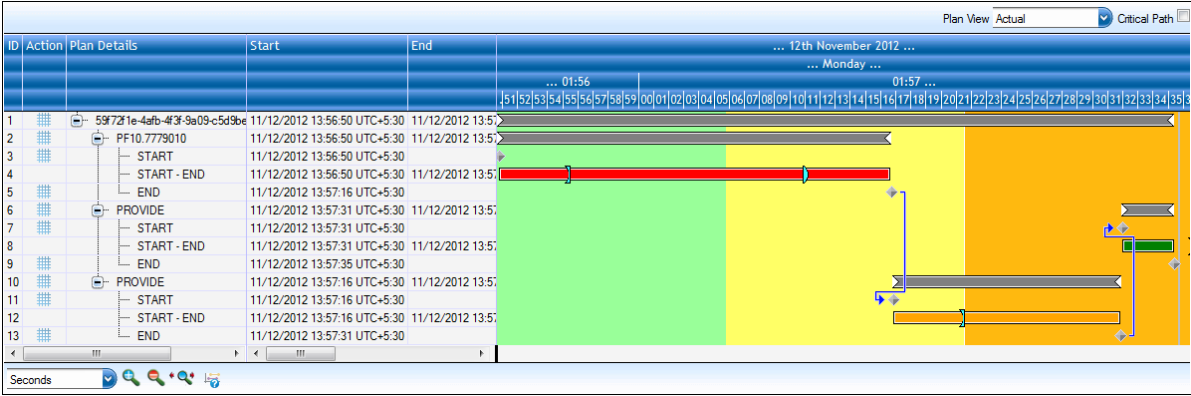
The Section level color code describes the status of the section. The following figure displays the section color codes representing the plan item section in different states:



If the section has not started its execution or the section is in the execution state, it is represented by grey color .

The following figure displays all the available section colors:

Section Colors



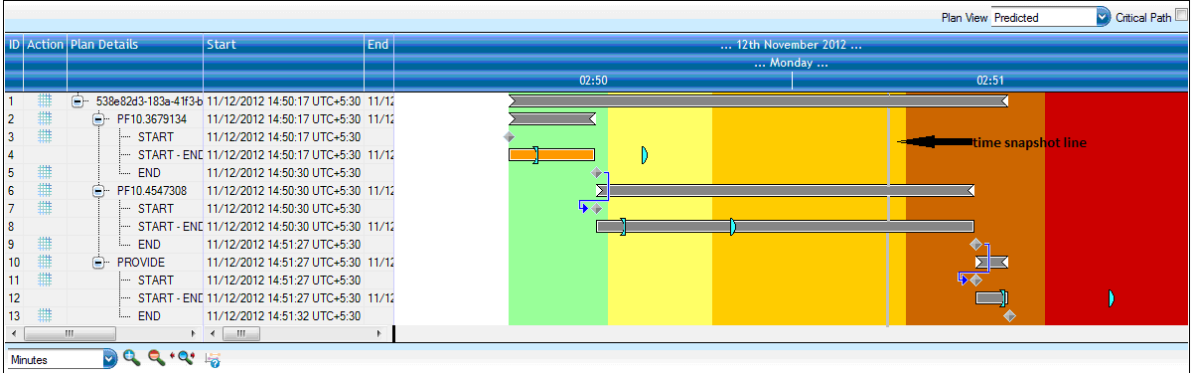
The following table describes the GANTT chart section level colors:

Color	Description
Green	Section completed within typical duration.
Yellow	Section completed after typical duration but before maximum duration
Orange	Section completed after maximum duration.
Red	Section in-progress (executing) or not yet started.

GANTT Chart Time Snapshot Line

The Time snapshot line represents the time when Gantt chart is loaded on the UI.

Time Snapshot Line



Tooltips

Each of the items on Gantt diagram has a tooltip, which has further information about that particular Gantt item.

Here is a description of tooltips for each of the Gantt items and the information that is displayed when user take mouse pointer over a Gantt item:

- Plan Summary Task Bar tooltip

Plan ID

466da968-a40b-44b1-bb31-3ae06ee5bb62

Start Time

05/03/2013 16:47:50 UTC+5:30

End Time

05/03/2013 16:50:29 UTC+5:30

- *Plan Item Summary Task Bar tooltip*

Plan Item ID	0.4309616438866778
Description	PROVIDE
Start Time	05/03/2013 16:49:48 UTC+5:30
End Time	05/03/2013 16:50:29 UTC+5:30
PlanFragment ID	PF1
PlanFragment Name	AFF_EP_PROVIDE
PlanFragment Version	1.0

- *Milestone tooltip*

Milestone ID	START
Start Time	05/03/2013 16:49:48 UTC+5:30

- *Section tooltip*

Start Time	05/03/2013 16:49:48 UTC+5:30
End Time	05/03/2013 16:50:29 UTC+5:30

- *Typical End Icons tooltip*

Forecasted Typical End	05/03/2013 16:50:18 UTC+5:30
------------------------	------------------------------

Maximum End Icons tooltip

Forecasted Maximum End	05/03/2013 16:50:48 UTC+5:30
------------------------	------------------------------

Activity Log

This section explains how to view the status of objects in the Order Management using the *Activity Log* page.


About Activity Log

You can view a detailed log of activities for an Order, Plan, Rule and FP Orders to see the history at run-time and perform a search on the following objects in the Order Management:

- Order Ref
- Plan Id
- Rule Name
- FP Order Id
- FP OrderLine Id
- FP Resource Id
- FP TechnicalOrder Id

An activity log displays messages for the following actions:

- status changes

Messages are grouped chronologically by the object type. You can press the **Refresh** button  at any time to update the current view.

Viewing the Activity Log

You must search an object to view the activity log associated with a particular object. To do this, refer to [Searching for an Order in the Activity Log](#).

Searching for an Order in the Activity Log

To view an activity log for a selected object, you must first search for it from the Activity Log page. Once you have found the object whose log you wish to view, select it to display it in the Activity Log page. To do this, perform the following steps:

1. Access the Order Management Long Running.
2. Click the **Activity Log** tab.
3. In the Search field, type the reference ID of an order to search.



The text is case-sensitive.

Searching Order

4. Click the **Search** button . Either of the following is the search result:
 - a. The details of an order that matches your search criteria is displayed.
 - b. If no match is found, no list is displayed.

Interpreting the Log Messages

It is important to interpret and understand the meaning of the log messages. To explain the log message, given below is an example illustrating a search performed on an order.

Log Message

Date	Ref	Type	Message
03/22/2011 10:11:51	OMSORDREF_15	ORDER	Updated ORDER OMSORDREF_15 from status EXECUTION to COMPLETE
03/22/2011 10:08:55	OMSORDREF_15	ORDER	Order OMSORDREF_15 created with status START
03/22/2011 10:09:56	OMSORDREF_15	ORDER	Order Amendment for Order OMSORDREF_15 created with status START
03/22/2011 10:09:56	OMSORDREF_15	ORDER	Updated ORDER OMSORDREF_15 from status EXECUTION to SUSPENDED
03/22/2011 10:09:57	OMSORDREF_15	ORDER	Updated ORDER OMSORDREF_15 from status SUSPENDED to EXECUTION
03/22/2011 10:08:56	OMSORDREF_15	ORDER	Updated ORDER OMSORDREF_15 from status START to SUBMITTED

The following information is displayed about the searched order:

Column Heading	Description
Date	The date and time the object was updated.
Ref	The reference ID of the object.
Type	The object type, for example, order, plan, order line, or plan item.

Column Heading	Description
Message	The action that was performed on the object. Refer to Understanding the Types of Log Messages to know about the different types of log messages and their respective meanings.

Understanding the Types of Log Messages

The log messages describe the action that was performed on the object. That action can be any one of the following:

Log Message	Description
Updated <i>object</i> from status <i>status</i> to <i>status</i>	<p>updated an order where:</p> <ul style="list-style-type: none"> <i>object</i> is either order, plan, order line, or plan item the first <i>status</i> is the object's previous status; the second <i>status</i> is the status the object has been changed to.
Amended <i>order id</i> of status <i>status</i>	<p>an order has been amended where:</p> <ul style="list-style-type: none"> <i>order id</i> is the unique ID of the order. <i>status</i> is the current status of the order.
Order <i>order id</i> created with status <i>status</i>	<p>an order has been created where:</p> <ul style="list-style-type: none"> <i>order id</i> is the unique ID of the order. <i>status</i> is the current status of the order.

Fulfillment Provisioning Service Order Hierarchy

The FP Service order is displayed in plan tab of Order Management. It is integrated with the Plan item tree hierarchy displayed at left side of the Plan tab. The Fulfillment Provisioning Service Order (SO) integration with the plan tree hierarchy display:

- Milestone for the related plans. The milestone contains the START, END, and intermediate milestones
- The hierarchy with the Fulfillment Provisioning service order ID, which contains the child node as the OrderLine, Rollback flow, and nominal flow
- The Rollback and Nominal node contain technical order IDs
- All the technical order line IDs, which can be rolled back are contained under the Rollback flow



If there is no roll back associated with service order then all the technical order IDs fall under the service order node.

The Fulfillment Provisioning flow accepts and parses an incoming provisioning message and generates a corresponding service order. The product orders are also created and attached to the service order.

TIBCO Order Management - Long Running performs a number of important tasks on the product order, according to its currently defined rules and other configuration. These tasks include:

- Decompose each product order into one or more technical product orders
- Validate, add, exclude, and sort these technical product orders and attach product order flows to them

- Enrich the data with specific provisioning parameters

Fulfillment Provisioning Attributes and Parameters

The Fulfillment Provisioning parameters and attributes are displayed on the right side panel of the Plan page. These are displayed from the Fulfillment Provisioning server based on the following node type selected from the Plan panel:

- Service Order
- Order Line
- Technical Order Line
- Resource Order Line



The attributes and parameters have name-value pairs based on the node selected in the Plan tree panel. The corresponding process flow is displayed based on the selected node. If the Plan item does not contain the FP service order, then only Milestone is added to the hierarchy view.

If a Plan item contains the Fulfillment Provisioning service order then the service order is displayed as:

Fulfillment Provisioning Service Order View

Browse Plans

Plan ID	Originator	Description	Plan For
90245a4b-20b4-4bb7-b7f5-a4fe64d9d906	Orderline	Plan For: Voice_Update_act10_11_500_1_1346061017093_http_P8	
Order ID	306cc62-a7d8-48e0-0757-066ad020x80	Created On	2010/02/12 14:52:33 UTC+5:30
Status	PPFT/UPDN	Order Ref ID	Voice_Update_act10_11_500_1_1346061017093_http_P8
Is Amendment?	Initial	Changed On	18/02/2012 14:53:34 UTC+5:30
Risk Region	NORMAL	Start Date	2010/02/12 14:53:34 UTC+5:30
		Jeopardy Message	None

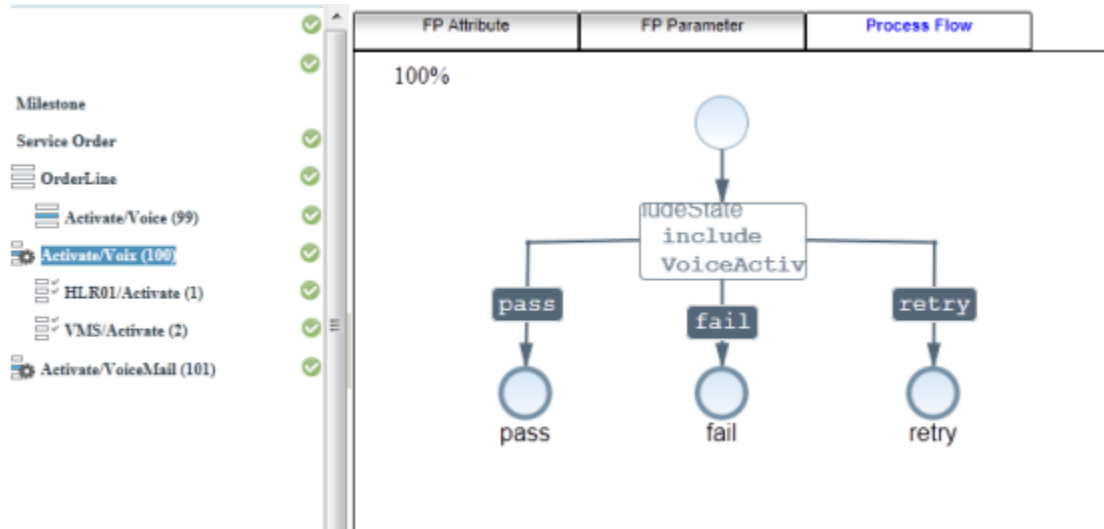
Fulfillment Provisioning Process Flow

The Fulfillment Provisioning parameters and attributes are displayed on the right side panel of the Plan page. These are displayed from the Fulfillment Provisioning server based on the following node type selected from the Plan panel:

- Service Order
- Order Line
- Technical Order Line

The Process Flow displays a graphical representation of process flow for the selected node.

Fulfillment Provisioning Process Flow



Searching for Fulfillment Provisioning Components

When you select Order Id, Order Line Id, Technical Order Id or Resource Id, a filter icon is displayed. Click the icon to display the corresponding search fields.

Filter Icon



For Order Line Id, Technical Order Id and Resource Id, the search text box is disabled and you have to select minimum and mandatory fields from the filter. Follow the steps below:

1. Click the **Activity Log** tab.
2. In the **Search By** box, select Fulfillment Provisioning components
3. In the **Search** field, type the Fulfillment Provisioning order Id or select the filter.
4. Click **Enter** or the search icon to retrieve records.

Filter Icon

Date	Severity	Source	EventId	Originator	FP Message
1378795350486770	info	prov	2	completed	Entrée dans la heap complétée de l'OrderQueue.
1378795350486837	info	pop	1	GSM/Activate	sortie du ProductOrderProcessor, tentative 1, status: OK
1378795350486920	info	pop	3	Voice/Activate	Le Product Order Execution est fini, tentative 1, status: OK
1378795350486966	info	wip	2	VMS/Activate..._confip04	Execution de Cartridge WO ENDED - tentative : 1 - instance : VMS - etat : OK - temps de traitement : 7610 microsecondes
1378795350487128	info	wip	2	HLR/Activate..._confip04	Execution de Cartridge WO ENDED - tentative : 1 - instance : HLR01 - etat : OK - temps de traitement : 6424 microsecondes
1378795350487394	info	carbase	57	VMS 0	OUTGOING: For WO «Activate»: return status is «WOS_Completed», in dataset: Found param:«WVOO_OBJECT», with value list: «VMS/Activate obj» Found param:«WVOO_INVSET», with value list: «VMS/Activate» Found param:«BASESODATA.startTime», with value list: «137879535048734858» Found param:«RNE_COMMAND», with value list: «Create 9998867777 XYZ English» Found param:«RNE_RESPONSE», with value list: «SUCCESS EXECUTED.»
1378795350487573	info	carplugin	33	VMS 0	Poursuite de l'exécution.
1378795350487660	info	est	28	VMS 0	Early token trouvé «SUCCESS».
1378795350487698	info	est	21	VMS 0	Buffer reçu «SUCCESS EXECUTED...».
13787953504878171	info	est	23	VMS 0	Envoi du buffer «Create 9998867777 XYZ English...».
1378795350487812	info	carbase	57	HLR01 0	OUTGOING: For WO «Activate»: return status is «WOS_Completed», in dataset: Found param:«WVOO_OBJECT», with value list: «HLR/Activate obj» Found param:«WVOO_INVSET», with value list: «HLR/Activate» Found param:«MSISDN», with value list: «33610814402» Found param:«RNE», with value list: «12345678901» Found param:«Rdn», with value list: «0» Found param:«cdn», with value list: «0» Found param:«BASESODATA.startTime», with value list: «1378795350486950000» Found param:«RNE_COMMAND», with value list: «ADDRESS MSISDN+33610814402 SOCLUP+0 SOCLUP+0» Found param:«RNE_RESPONSE», with value list: «SUCCESS EXECUTED.»
1378795350487879	info	carplugin	33	HLR01 0	Poursuite de l'exécution.
1378795350487951	info	est	28	HLR01 0	Early token trouvé «SUCCESS».
1378795350487947	info	est	21	HLR01 0	Buffer reçu «SUCCESS EXECUTED...».
1378795350487979	info	carbase	48	VMS 0	NOVALIDATION: For WO «Activate»: In dataset: Found param:«WVOO_OBJECT», with value list: «VMS/Activate obj» Found param:«WVOO_INVSET», with value list: «VMS/Activate» Found param:«BASESODATA.startTime», with value list: «13787953504871754000»
1378795350487943	info	est	23	HLR01 0	Envoi du buffer «ADDRESS MSISDN+33610814402 SOCLUP+0 SOCLUP+0...».
1378795350487918	info	carplugin	33	HLR01 0	Poursuite de l'exécution.
1378795350487983	info	est	28	HLR01 0	Early token trouvé «SUCCESS».
1378795350487983	info	est	21	HLR01 0	Buffer reçu «SUCCESS EXECUTED...».
1378795350487952	info	wip	0	VMS/Activate..._confip04	Execution de Cartridge WO STARTED - tentative : 1 - instance : VMS action (executée).

Third Party Access to Order Management - Long Running Functionality User Interface

Order Management Server UI components are exposed to third party applications using OAuth2 authentication. Where the client needs to get the initial access token for Order Management Server UI and request for particular Order Management Server UI component by providing the user name and password.

Implementing OMSUIClient.jar

- Below dependencies can be added to in pom 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 -->
```

- Fetch access token through OMSUIClient for Order Management Server UI:
 - Call `getAccessToken (accessTokenUri, clientID, userName, password, clientSecret)` method of `AccessTokenService` of `omsuiClient`.
 - Provide the following parameters:
 - Access token URL: `http://[hostname]:[PortName]/omsui/oauth/token`
 - Client ID: [Provided by OMSUI] (for example, my-trusted-client-with-secret)
 - User name: [user name for Order Management Server UI authorization]
 - Password: [password for Order Management Server UI authorization]
 - clientSecret: [some secret key provided by OMSUI]
- OMSUIClient fetches access tokens for OAuth2 authorization, you can then add those token in the target URI to access OMSUI.

Single URI to Access Order Management Server User Interface Component

You can directly access Order Management Server User Interface by providing target components for each order, dashboard, plan, ruleconfig, catalog and activitylog. The target parameter in URI redirects to a specific component of Order Management Server User Interface, for example, `http://localhost:8080/omsui/OTS/main?target=order` redirects to the Order's tab of OMSUI.

The following are the two different scenarios:

- Using Order Management Server User Interface Client:** URI is redirected to a specific component based on the target and search parameters.

- **Directly Accessing Order Management Server User Interface URL with target specified:** Initially you are redirected to the login page and once authentication completes, you are redirected to a specific component based on the target and search parameter.

Target Parameters for Order Management Server User Interface

The following are the target parameters for Order Management Server User Interface:

- Order
- Plan
- Ruleconfig
- ActivityLog

Additional Parameters

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

The following are search parameters for order, plan, and activitylog:

Order:

- orderRef
- orderID
- status

Plan:

- orderRef
- PlanID

Activitylog

- Type (orderRef/plan)

Data Access Interfaces

Overview

TIBCO Order Management - Long Running exposes five JMS based interfaces for process components to access or update the user-defined fields data during order fulfillment.

The following are the interfaces:

1. **GetOrder** - to get the order requests of all the versions (including amendments) of an order.
2. **GetPlan** - to get the User Defined Fields data of the execution plan associated with an order. This can also include the User Defined Fields data of all comprising plan items.
3. **GetPlanItems** - to get the User Defined Fields data of the specific plan items passed in the request.
4. **SetPlan** - to update or replace the User Defined Fields data of the execution plan associated with an order. This can also include the User Defined Fields data of one or more comprising plan items.
5. **SetPlanItem** - to update/replace the User Defined Fields data of a specific plan item passed in the request.

The following sub-sections cover these JMS interfaces in details.

The schema for these interfaces is `$OM_HOME/schemas/schema/tds/sharedResources/schemas/services/TransientDataStoreService.xsd`.

The TIBCO Enterprise Message Service messages including replies for all Transient Data Store operations are persistent. These messages can be made non-persistent by using the provided configurations. For more information, see "Changing Transient Data Store Operation Messages to Non-Persistent" in the *TIBCO Order Management - Long Running Administration Guide*.

Get Order

Overview

The **GetOrder** interface can be used by the process components to retrieve a specific order from the Order Management system. It is a synchronous request/reply interface on a JMS queue that returns a reply either to the specified `JMSReplyTo` destination on the request message or to the default reply queue if not specified.

If an exception occurs during Get Order operation, then it is logged into the `omsServer` log. The details of the exception are returned in the response.

Get Order Request

The request specification is as follows:

Queue or Topic	Queue
Destination	<code>tibco.aff.tds.order.read.request</code>

The following properties must be passed in the request message header:

Element	Type	Cardinality	Description
---------	------	-------------	-------------

nm	String	Required	Interface identifier name; MUST be set as <i>GetOrderRequestEvent</i> .
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
orderId	String	Required	Internal unique identifier for the order to retrieve.
requestReply	Boolean	Required	If set to true: The response is sent on the temporary queue by default or on the destination set as JMSReplyTo property in the request message. If set to false: The response is sent on <code>tibco.aff.tds.order.reply</code> queue.
allValues	Boolean	Required	If true, retrieve all versions of the order in case of amendments.

There is no body (payload) associated with the request message.

Get Order Response

The response specification is as follows:

Queue or Topic	Queue
Destination	temp queue or JMSReplyTo destination or <code>tibco.aff.tds.order.reply</code> queue

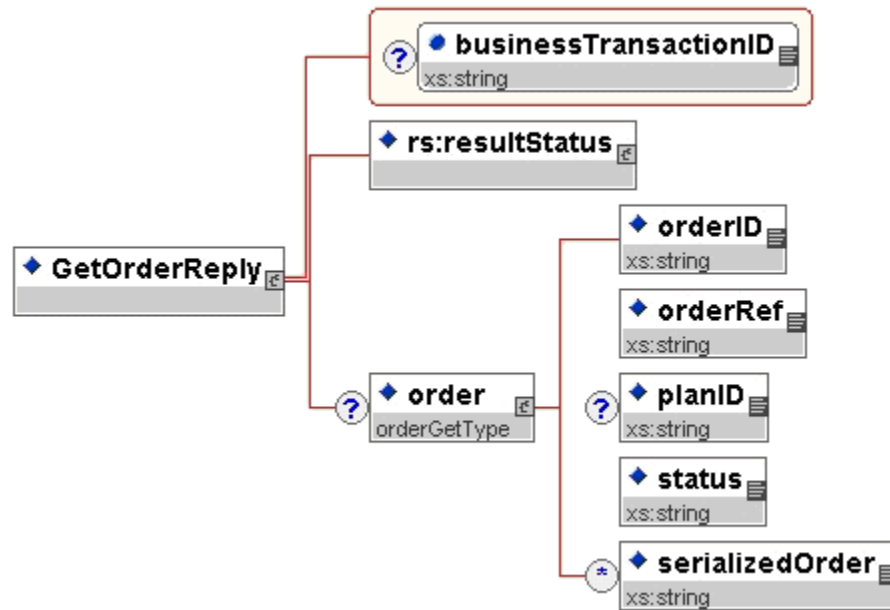
The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
success	Boolean	Required	Flag indicating if the call was successful.
messageCode	String	Required	Result message code.
message	String	Required	Result message.

orderId	String	Required	Internal unique identifier for the order specified in the request.
found	Boolean	Required	Flag indicating if the order was found.

The response message contains the XML payload as per the following schema:

Get Order Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
resultStatus	Type	Required	Result status type. See Appendix A for the specification of this type.
order	Type	Optional	Order type. If the order is not found this is omitted.
order/orderID	String	Required	Internal unique identifier for the order.
order/orderRef	String	Required	External unique identifier for the order.
order/planID	String	Optional	The internal unique identifier for the plan for the order if it exists.
order/status	String	Required	The Status of the order from a cache data store perspective (active or complete)

order/ serializedOrder	String	0-M	Xml string representation of the order, multiple if all versions are requested, otherwise just the active one
---------------------------	--------	-----	---

Get Order Messages and Message Codes

The error codes and their respective error messages for the Get Order interface are as follows:

Get Order Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed GetOrderRequestEvent	Order is found and data is mapped in the response successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input orderID in GetOrderRequestEvent	orderID parameter in the request header is null or empty.
FOM-DATA-ACCESS-ORDER-NOT-FOUND-9999	Order not found for orderID <orderID value>	Order is not found against the orderID specified in the request header.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing GetOrderRequestEvent	Any other exception occurred when processing the request.
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing GetOrderRequestEvent	Resource Failure (DB connection) issues are encountered when processing the request.

Get Plan

Overview

The GetPlan interface can be used by the process components to retrieve the plan corresponding to an order from the TIBCO Order Management - Long Running system. It is a synchronous request/reply interface on a JMS queue that returns a reply either to the specified JMSReplyTo destination on the request message or to the default reply queue, if not specified.

If an exception occurs during the GetPlan operation, it is logged into the omsServer log. The details of the exception are returned in the response.

Get Plan Request

The request specification is as follows:

Queue or Topic	Queue
Destination	<code>tibco.aff.tds.plan.read.request</code>

The following properties must be passed in the request message header:

Request Endpoint: /v1/plan

Element	Type	Cardinality	Description
nm	String	Required	The interface identifier name; must be set as <code>GetPlanRequestEvent</code> .
businessTransactionID	String	Optional	The unique identifier for tracing purposes across function calls.
planID	String	Required	The internal unique identifier for the plan to retrieve.
correlationID	String	Optional	The unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
requestReply	Boolean	Required	<p>If set to <code>true</code>:</p> <p>The response is sent on the temporary queue by default or on the destination set as <code>JMSReplyTo</code> property in the request message.</p> <p>If set to <code>false</code>:</p> <p>The response is sent on <code>tibco.aff.tds.plan.reply</code> queue.</p>
idsOnly	Boolean	Required	If <code>true</code> , only returns the IDs of elements rather than all plan data. If <code>false</code> , returns all plan data.
includeItems	Boolean	Required	If <code>true</code> returns all plan items with the plan. If <code>false</code> , only the plan details are returned.
originator	String	Optional	The component, which has originated this call. For example, the name of the process component <code>PC_BUNDLE_PROVIDE</code> .

MATCHING_PI_UDF_NAME_VAL	String	Optional	<p>The value of this field is in the format: user-defined field name=user-defined field value. When this value is set, the output is a plan with the plan items, which have the matching user-defined field name and value as specified in the header value together with all the User Defined Fields for the corresponding matched plan items.</p> <p>If the format for this user-defined field is not in the form user-defined field name=user-defined field value, an error is returned.</p>
ALL_PI_SINGLE_UDF_NAME	String	Optional	<p>The value of this is a user-defined field name. The output is a plan with all the plan items and the plan items have only the user-defined field specified in the header if present.</p> <p>If both MATCHING_PI_UDF_NAME_VAL and ALL_PI_SINGLE_UDF_NAME are present in the request, an error is returned.</p>
ALL_PI_SINGLE_UDF_NAME_IGNORE_EMPTY_PI	Boolean	Optional	<p>This is intended to be used in conjunction with ALL_PI_SINGLE_UDF_NAME. The default value of false indicates that all plan items are present in the response even if they do not have a user-defined field matching the user-defined field specified in ALL_PI_SINGLE_UDF_NAME. A true value indicates that plan items with no matching User Defined Fields are removed from the response.</p>

Nobody (payload) is associated with the request message.

Get Plan Response

The response contains a user-defined field, namely 'PLAN_DESCRIPTION' under the Plan element, which gives the plan description to the process components. This value comes from the order description provided during the order submission process. If the order description is not provided, a default plan description 'Plan For' <order ref> is returned in the **PLAN_DESCRIPTION UDF** field.

The response specification is as follows:

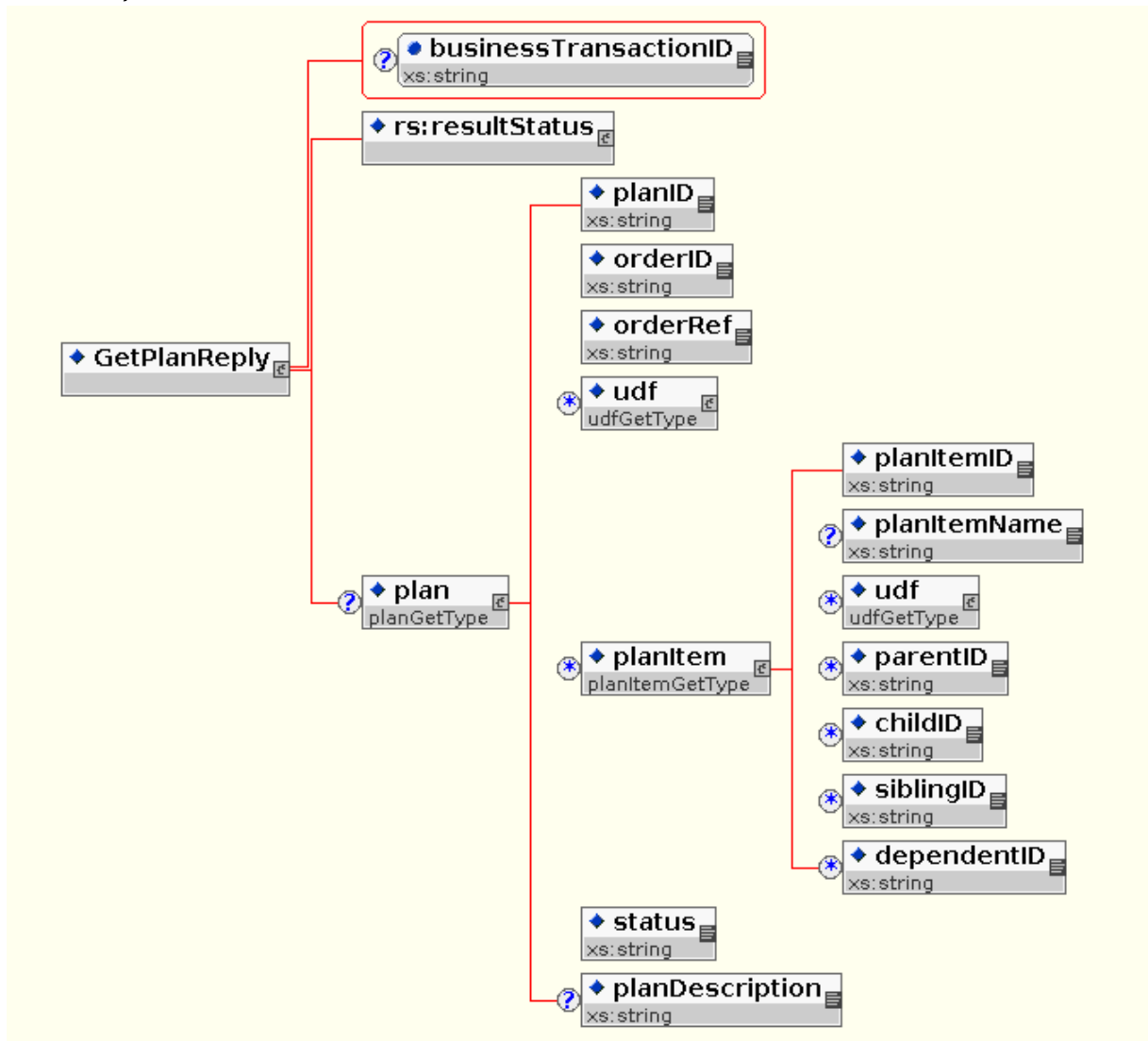
Queue or Topic	Queue
----------------	-------

Destination	temp queue or JMSReplyTo destination or <code>tibco.aff.tds.plan.reply</code> queue
-------------	---

The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	The unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	The unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
success	Boolean	Required	The flag indicating if the call was successful.
messageCode	String	Required	The result message code.
message	String	Required	The result message.
planID	String	Required	The internal unique identifier for the plan specified in the request.
found	Boolean	Required	The flag indicating if the plan was found.

The response message contains the XML payload according to the following schema:

Get Plan Response

The following table lists the details of the response elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	The unique identifier for tracing purposes across function calls.
resultStatus	Type	Required	The result status type. See Appendix A for the specification of this type.
plan	Type	Optional	The plan type. If the plan is not found, this is omitted.
plan/planID	String	Required	The internal unique identifier for the plan.
plan/orderID	String	Required	The internal unique identifier for the order for the plan.

plan/orderRef	String	Required	External unique identifier for the order for the plan.
plan/udf	Type	0-M	user-defined field type.
plan/udf/type	String	Optional	Type of the user defined field.
plan/udf/flavour	String	Optional	<p>Flavor of the user-defined field. The valid values are one of the following three:</p> <ul style="list-style-type: none"> • config - For user-defined field corresponding to a characteristic in the product model or a system user-defined field generated by Automated Order Plan Development. • input - For user-defined field passed in the order. • output - For user-defined field set by the process component.
plan/udf/name	String	Required	Field name.
plan/udf/value	String	Optional	Field value.
plan/udf/originalValue	String	Optional	Original field value at the time of plan creation.
plan/udf/lastModified	DateTime	Optional	Timestamp when the user-defined field was last modified.
plan/planItem	Type	0-M	Plan item type.
plan/planItem/ planItemID	String	Required	Internal unique identifier for the plan item.
plan/planItem/ planItemName	String	Optional	Name of the process component.
plan/planItem/udf	Type	0-M	user-defined field type.
plan/planItem/udf/type	String	Optional	Type of the user defined field.
plan/planItem/udf/ flavour	String	Optional	<p>Flavor of the user-defined field. The valid values are one of the following three:</p> <ul style="list-style-type: none"> • config - For user-defined field corresponding to a characteristic in the product model or a system user-defined field generated by Automated Order Plan Development. • input - For user-defined field passed in the order. • output - For user-defined field set by the process component.

plan/planItem/udf/name	String	Required	Field name.
plan/planItem/udf/value	String	Optional	Field value.
plan/planItem/udf/originalValue	String	Optional	Original field value at the time of plan creation.
plan/planItem/udf/lastModified	DateTime	Optional	Timestamp when the user-defined field was last modified.
plan/planItem/parentID	String	0-M	IDs of the plan items, which depend on the current plan item.
plan/planItem/childID	String	0-M	IDs of the plan items on, which the current plan item depends.
plan/planItem/siblingID	String	0-M	IDs of the plan items corresponding to SIBLING_PRODUCT_* of the product fulfilled by current plan item.
plan/planItem/dependentID	String	0-M	IDs of the plan items corresponding to DEPENDENT_PRODUCT_* of the product fulfilled by current plan item.
plan/status	String	Required	Status of the plan from a data perspective: active or complete.
plan/planDescription	String	Optional	Plan description.

Get Plan Messages and Message Codes

The error codes and their respective error messages for the Get Plan interface are as follows:

Get Plan Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed GetPlanRequestEvent	Plan is found and data is mapped in the response successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planID in GetPlanRequestEvent	planID parameter in the request header is null or empty.
FOM-DATA-ACCESS-PLAN-NOT-FOUND-9999	Plan not found for planID <planID value>	Plan is not found against the planID specified in the request header.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing GetPlanRequestEvent	Any other exception occurred when processing the request.

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing <code>GetPlanRequestEvent</code>	Resource Failure (DB connection) issues are encountered when processing the request.

Get Plan Items

Overview

The `GetPlanItems` interface can be used by the process components to retrieve the data of one or many plan items in the plan corresponding to an order from the TIBCO Order Management - Long Running system. It is a synchronous request/reply interface on a JMS queue that returns a reply either to the specified `JMSReplyTo` destination on the request message or to the default reply queue if not specified.

If an exception occurs during `GetPlanItems` operation, then it is logged into the `omsServer` log. The details of the exception are returned in the response.

Get Plan Items Request

The request specification is as follows:

Queue or Topic	Queue
Destination	<code>tibco.aff.tds.plan.read.request</code>

Request Endpoint: `/v1/planitems/get`

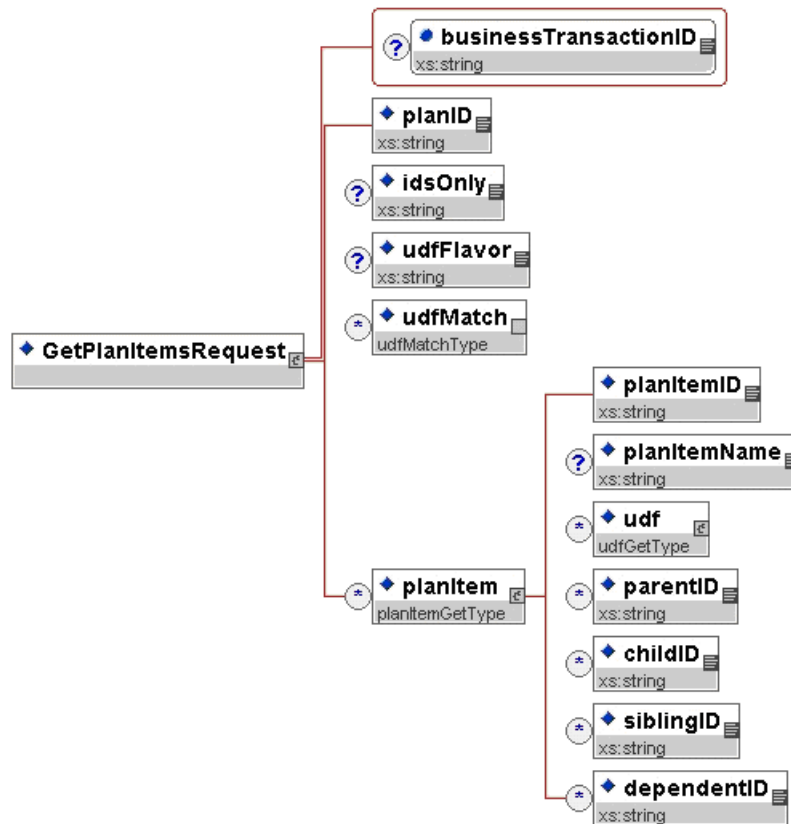
The following properties must be passed in the request message header:

Element	Type	Cardinality	Description
<code>_nm_</code>	String	Required	The interface identifier name; must be set as <code>GetPlanItemsRequestEvent</code> .
<code>businessTransactionID</code>	String	Optional	The unique identifier for tracing purposes across function calls.
<code>planID</code>	String	Required	The internal unique identifier for the plan to retrieve.
<code>correlationID</code>	String	Optional	The unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.

requestReply	Boolean	Required	<p>If set to <code>true</code>:</p> <p>The response is sent on the temporary queue by default or on the destination set as <code>JMSReplyTo</code> property in the request message.</p> <p>If set to <code>false</code>:</p> <p>The response is sent on <code>tibco.aff.tds.plan.reply</code> queue.</p>
idsOnly	Boolean	Required	If <code>true</code> , returns the IDs of elements rather than all plan data. If <code>false</code> , returns all plan data.
originator	String	Optional	The component, which has originated this call. For example, the name of the process component <code>PC_BUNDLE_PROVIDE</code> .
INCLUDE_RELATED_PLANITEMS	Boolean	Optional	<p>The value of this header is <code>true</code> or <code>false</code>, and the absence of this header denotes a <code>false</code> value. When this header is set in the request, the output ID all the plan items, which belong to the same order line as the input plan item.</p> <p>When this header is set in the request, only a single plan item can be present in the <code>getPlanItem</code> request. If multiple plan items are present in the request, an error is returned.</p>

The request message body must contain the XML payload as per the following schema:

Get Plan Items Request



The following table lists the details of the getPlanItemsRequest elements.

Element	Type	Cardinality	Description
businessTransacti onID	String	Optional	Unique identifier for tracing purposes across function calls.
planID	String	Required	Internal unique identifier for the plan to retrieve.
idsOnly	String	Optional	If true only returns the IDs of elements rather than all plan data. Otherwise, if false returns all plan data.
udfFlavour	String	Optional	Currently not implemented.
udfMatch	String	Optional	Currently not implemented.
planItem	Type	0-M	Plan item type.
planItem/ planItemID	String	Required	Internal unique identifier for the plan item to retrieve.
planItem/ planItemName	String	Optional	Not Applicable.
planItem/udf	Type	0-M	Not Applicable.

parentID	String	0-M	Not Applicable.
childID	String	0-M	Not Applicable.
siblingID	String	0-M	Not Applicable.
dependentID	String	0-M	Not Applicable.

Get Plan Items Response

The response specification is as follows:

Queue or Topic	Queue
Destination	temp queue or JMSReplyTo destination or <code>tibco.aff.tds.plan.reply</code> queue

The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
Success	Boolean	Required	Flag indicating if the call was successful.
messageCode	String	Required	Result message code.
Message	String	Required	Result message.
planID	String	Required	Internal unique identifier for the plan specified in the request.
Found	Boolean	Required	Flag indicating if the plan was found.

The response message contains the XML payload as per the following schema:

planItem/udf/flavour	String	Optional	Flavor of the user-defined field. The valid values are one of the following three: <ul style="list-style-type: none"> config - For user-defined field corresponding to a characteristic in the product model or a system user-defined field generated by Automated Order Plan Development. input - For user-defined field passed in the order. output - For user-defined field set by the process component.
planItem/udf/name	String	Required	Field name.
planItem/udf/value	String	Optional	Field value.
planItem/udf/originalValue	String	Optional	Original field value at the time of plan creation.
planItem/udf/lastModified	DateTime	Optional	Timestamp when the user-defined field was last modified.
planItem/parentID	String	0-M	IDs of the plan items, which depends on the current plan item.
planItem/childID	String	0-M	IDs of the plan items on which the current plan item depends.
planItem/siblingID	String	0-M	IDs of the plan items corresponding to SIBLING_PRODUCT_* of the product fulfilled by current plan item.
planItem/dependentID	String	0-M	IDs of the plan items corresponding to DEPENDENT_PRODUCT_* of the product fulfilled by current plan item.

Get Plan Items Messages and Message Codes

The error codes and their respective error messages for the Get Plan Items interface are as follows:

Get Plan Items Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed GetPlanItemsRequestEvent	PlanItem or PlanItems are found and data is mapped in the response successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planID in GetPlanItemsRequestEvent	planID parameter in the request header or XML payload is null or empty.

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planItemID in GetPlanItemsRequestEvent	planItemID parameter in the request XML payload is null or empty.
FOM-DATA-ACCESS-INVALID-REQUEST-9999	Invalid payload in GetPlanItemsRequestEvent	XML payload in the request is not valid.
FOM-DATA-ACCESS-INCONSISTENT-INPUT-9999	Inconsistent input values in request header and payload	Value of planID parameter in the request header and XML payload is not consistent.
FOM-DATA-ACCESS-PLANITEM-NOT-FOUND-9999	PlanItem not found for planID <planID value> and planItemID <planItemID value>	PlanItem is not found against the planID and planItemID specified in the request header.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing GetPlanItemsRequestEvent	Any other exception occurred when processing the request.
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing GetPlanItemsRequestEvent	Resource Failure (DB connection) issues are encountered when processing the request.

Set Plan

Overview

The SetPlan interface can be used by the process components to add a new user-defined field or update the value of an existing user-defined field of plan or any of the containing plan items in the TIBCO Order Management - Long Running system. However, it is suggested to use this interface for plan level User Defined Fields only since there is a separate interface for plan items as explained further in this guide.

It is a synchronous request/reply interface on a JMS queue that returns a reply either to the specified JMSReplyTo destination on the request message or to the default reply queue if not specified.

If an exception occurs during SetPlan operation, then it is logged into the omsServer log. The details of the exception are returned in the response.

Set Plan Request

The request specification is as follows:

Queue or Topic	Queue
Destination	tibco.aff.tds.plan.request

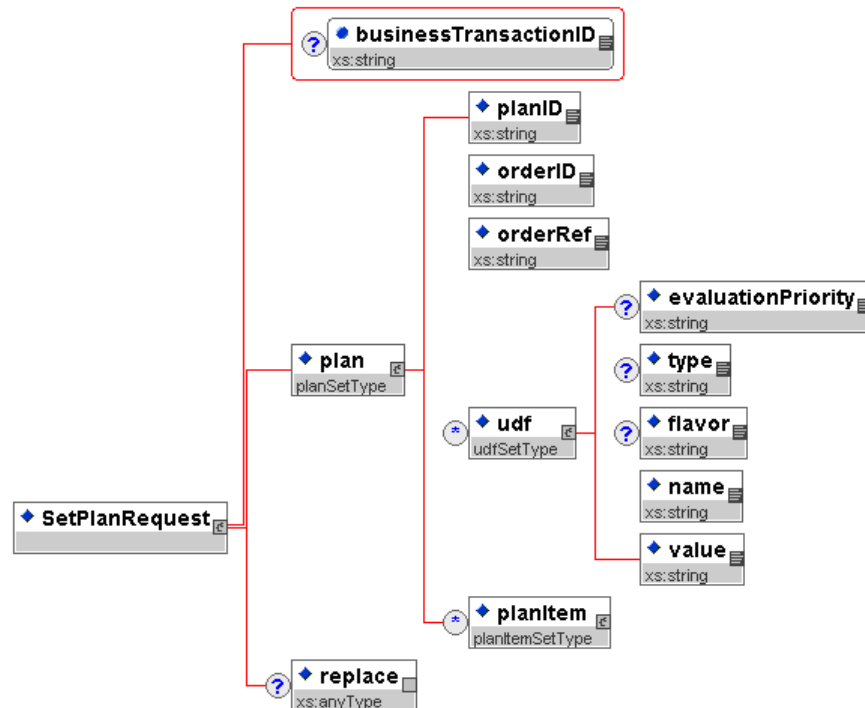
Request Endpoint: /v1/plan

The following properties must be passed in the request message header:

Element	Type	Cardinality	Description
nm	String	Required	Interface identifier name; MUST be set as <i>SetPlanRequestEvent</i> .
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
planID	String	Required	Internal unique identifier for the plan to retrieve.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
requestReply	Boolean	Required	<p>If set to true:</p> <p>The response is sent on the temporary queue by default or on the destination set as JMSReplyTo property in the request message.</p> <p>If set to false:</p> <p>The response is sent on <code>tibco.af.lds.plan.reply</code> queue.</p>
replace	Boolean	Required	<p>If set to true:</p> <p>All the existing User Defined Fields is replaced by the User Defined Fields that are present in the request.</p> <p>If set to false:</p> <p>The User Defined Fields passed in the request is merged with the existing User Defined Fields.</p> <p>In any of the above case, the uniqueness of a user-defined field is maintained based on the 'name' and 'flavor' combination in the user-defined field. A user-defined field having exactly same 'name' and 'flavor' is not duplicated, if the flag <code>EnableUniqueUDFNames</code> is set to true in Order Management Server configurations. In case of multiple User Defined Fields with exactly same name and flavor in the request, the value from the last encountered user-defined field is considered.</p>
originator	String	Optional	The component, which has originated this call. E.g. the name of the process component <code>PC_BUNDLE_PROVIDE</code> .

The request message body must contain the XML payload as per the following schema:

Set Plan Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
Plan	Type	Required	Plan type.
plan/planID	String	Required	Internal unique identifier for the plan to update.
plan/orderID	String	Required	Internal unique identifier for the order related to the plan to update.
plan/orderRef	String	Required	External unique identifier for the order related to the plan to update.
plan/udf	Type	0-M	
plan/udf/type	String	Optional	Type of the user defined field.
plan/udf/flavour	String	Optional	Flavor of the user-defined field. Must be set as <i>output</i> .
plan /udf/name	String	Required	Field name.
plan/udf/value	String	Required	Field value.

plan/planItem	Type	0-M	Plan item type.
plan/planItem/ planItemID	String	Required	Internal unique identifier for the plan item to update.
plan/planItem/ planItemName	String	Optional	Process component name.
plan/planItem/udf	Type	0-M	user-defined field type.
plan/planItem/udf/ type	String	Optional	Type of the user defined field.
plan/planItem/udf/ flavour	String	Optional	Flavor of the user-defined field. Must be set as <i>output</i> .
plan/planItem/udf/ name	String	Required	Field name.
plan/planItem/udf/ value	String	Required	Field value.
replace	Any	Optional	If true it completely replaces the plan item, otherwise merges the user-defined field data.

Set Plan Response

The response specification is as follows:

Queue or Topic	Queue
Destination	temp queue or JMSReplyTo destination or <code>tibco.aff.tds.plan.reply</code> queue

The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
success	Boolean	Required	Flag indicating if the call was successful.
messageCode	String	Required	Result message code.

message	String	Required	Result message.
planID	String	Required	Internal unique identifier for the plan specified in the request.

There is nobody (payload) associated with the response message.

Set Plan Messages and Message Codes

The error codes and their respective error messages for the Set Plan interface are as follows:

Set Plan Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed SetPlanRequestEvent	Plan is found and user-defined field data specified in the request in plan header and plan items is updated successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planID in SetPlanRequestEvent	planID parameter in the request header or XML payload is null or empty.
FOM-DATA-ACCESS-INVALID-REQUEST-9999	Invalid payload in SetPlanRequestEvent	XML payload in the request is not valid.
FOM-DATA-ACCESS-INCONSISTENT-INPUT-9999	Inconsistent input values in request header and payload	Value of planID parameter in the request header and XML payload is not consistent.
FOM-DATA-ACCESS-PLAN-NOT-FOUND-9999	Plan not found for planID <planID value>	Plan is not found against the planID specified in the request.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing SetPlanRequestEvent	Any other exception occurred when processing the request.
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing SetPlanRequestEvent	Resource Failure (DB connection) issues are encountered when processing the request.

Set Plan Item

Overview

The SetPlanItem interface can be used by the process components to add a new user-defined field or update the value of an existing user-defined field of the plan items in a plan in the TIBCO Order Management - Long Running System.

It is a synchronous request/reply interface on a JMS queue that returns a reply either to the specified JMSReplyTo destination on the request message or to the default reply queue if not specified.

If an exception occurs during SetPlanItem operation, then it is logged into the omsServer log. The details of the exception are returned in the response.

Set Plan Item Request

The request specification is as follows:

Queue or Topic	Queue
Destination	<code>tibco.aff.tds.plan.request</code>

Request EndPoint: /v1/planitems

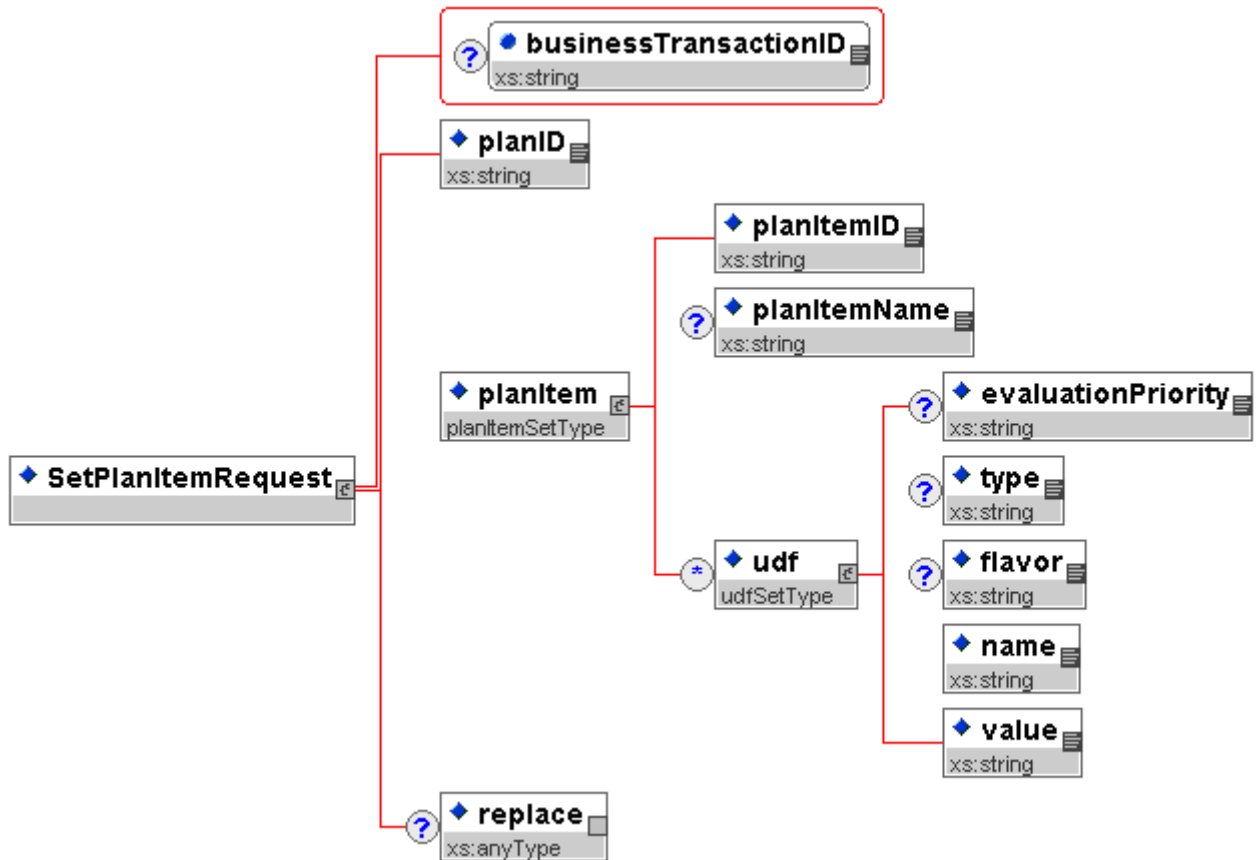
The following properties must be passed in the request message header:

Element	Type	Cardinality	Description
<code>_nm_</code>	String	Required	Interface identifier name; MUST be set as <i>SetPlanItemRequestEvent</i> .
<code>businessTransactionID</code>	String	Optional	Unique identifier for tracing purposes across function calls.
<code>planID</code>	String	Required	Internal unique identifier for the plan to retrieve.
<code>correlationID</code>	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
<code>requestReply</code>	Boolean	Required	<p>If set to true:</p> <p>The response is sent on the temporary queue by default or on the destination set as JMSReplyTo property in the request message.</p> <p>If set to false:</p> <p>The response is sent on <code>tibco.aff.tds.plan.reply</code> queue.</p>

replace	Boolean	Required	<p>If set to true:</p> <p>All the existing User Defined Fields is replaced by the User Defined Fields that are present in the request.</p> <p>If set to false:</p> <p>The User Defined Fields passed in the request is merged with the existing User Defined Fields.</p> <p>In any of the earlier mentioned cases, the uniqueness of a user-defined field is maintained based on the 'name' and 'flavor' combination in the user-defined field. A user-defined field having the same 'name' and 'flavor' do not get duplicated, if the flag <code>EnableUniqueUDFNames</code> is set to true in Order Management Server configurations. In case of multiple User Defined Fields with the same name and flavor in the request, the value from the last encountered user-defined field is considered.</p>
planItemID	String	Required	Internal unique identifier for the plan item to update.
originator	String	Optional	The component, which has originated this call. E.g. the name of the process component <code>PC_BUNDLE_PROVIDE</code> .

The request message body must contain the XML payload as per the following schema:

Set Plan Item Request



The following table lists the details of the elements.

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls.
planID	String	Required	Internal unique identifier for the plan to update.
planItem	Type	Required	Plan item type. Only user-defined field Name and Value are updated. If the Unique User Defined Fields are enabled for the engine an update occurs if disabled the entire current user-defined field payload is dropped and replaced with the new payload.
planItem/planItemID	String	Required	Internal unique identifier for the plan item to update.
planItem/planItemName	String	Optional	Process component name.
planItem/udf	Type	0-M	user-defined field type.

planItem/udf/type	String	Optional	Type of the user defined field.
planItem/udf/flavour	String	Optional	Flavor of the user-defined field. Must be set as output.
planItem/udf/name	String	Required	Field name.
planItem/udf/value	String	Required	Field value.
replace	Any	Optional	If true it completely replaces the plan item, otherwise merges the user-defined field data.

Set Plan Item Response

The response specification is as follows:

Queue or Topic	Queue
Destination	temp queue or JMSReplyTo destination or <code>tibco.aff.tds.plan.reply</code> queue

The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	Unique identifier for tracing purposes across function calls. This is used for logging.
correlationID	String	Optional	Unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
success	Boolean	Required	Flag indicating if the call was successful.
messageCode	String	Required	Result message code.
message	String	Required	Result message.
planID	String	Required	Internal unique identifier for the plan specified in the request.

There is nobody (payload) associated with the response message.

Set Plan Item Messages and Message Codes

The error codes and their respective error messages for the Set Plan Items interface are as follows:

Set Plan Items Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result Status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed SetPlanItemRequestEvent	PlanItem is found and user-defined field data specified in the request is updated successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planID in SetPlanItemRequestEvent	planID parameter in the request header or XML payload is null or empty.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input planItemID in SetPlanItemRequestEvent	planItemID parameter in the request header or XML payload is null or empty.
FOM-DATA-ACCESS-INVALID-REQUEST-9999	Invalid payload in SetPlanItemRequestEvent	XML payload in the request is not valid.
FOM-DATA-ACCESS-INCONSISTENT-INPUT-9999	Inconsistent input values in request header and payload	Value of planID or planItemID parameters in the request header and XML payload are not consistent.
FOM-DATA-ACCESS-PLANITEM-NOT-FOUND-9999	PlanItem not found for planID <planID value> and planItemID <planItemID value>	PlanItem is not found against the planID and planItemID specified in the request header.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing SetPlanItemRequestEvent	Any other exception occurred when processing the request.
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing SetPlanItemRequestEvent	Resource Failure (DB connection) issues are encountered when processing the request.

Get Key Mapping

Overview

The GetKeyMapping interface can be used by the process components to retrieve the orderID, planID and orderRef corresponding to one of the three elements in the request from the TIBCO Order Management - Long Running system. It is a synchronous request and reply interface on a JMS queue that returns a reply either to the specified JMSReplyTo destination on the request message or to the default reply queue if not specified.

If an exception occurs during Get KeyMapping operation, it is logged into the omsServer log. The details of the exception are returned in the response.

Get Key Mapping Request

The request specification is as follows:

Queue or Topic	Queue
Destination	<code>tibco.aff.tds.keymapping.request</code>

The following properties must be passed in the request message header:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	The unique identifier for tracing across function calls. This is used for logging.
correlationID	String	Optional	The unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
orderId/planID/orderRef	String	Choice (One mandatory)	The unique Identifier for the order to retrieve the other two elements.
requestReply	Boolean	Required	<p>If set to <code>true</code>:</p> <p>The response is sent on the temporary queue by default or on the destination set as <code>JMSReplyTo</code> property in the request message.</p> <p>If set to <code>false</code>:</p> <p>The response is sent on the <code>tibco.aff.tds.keymapping.reply</code> queue.</p>

No body (payload) is associated with the request message.

Get Key Mapping Response

The response specification is as follows:

Queue or Topic	Queue
Destination	The temp queue or <code>JMSReplyTo</code> destination or <code>tibco.aff.tds.keymapping.reply</code> queue

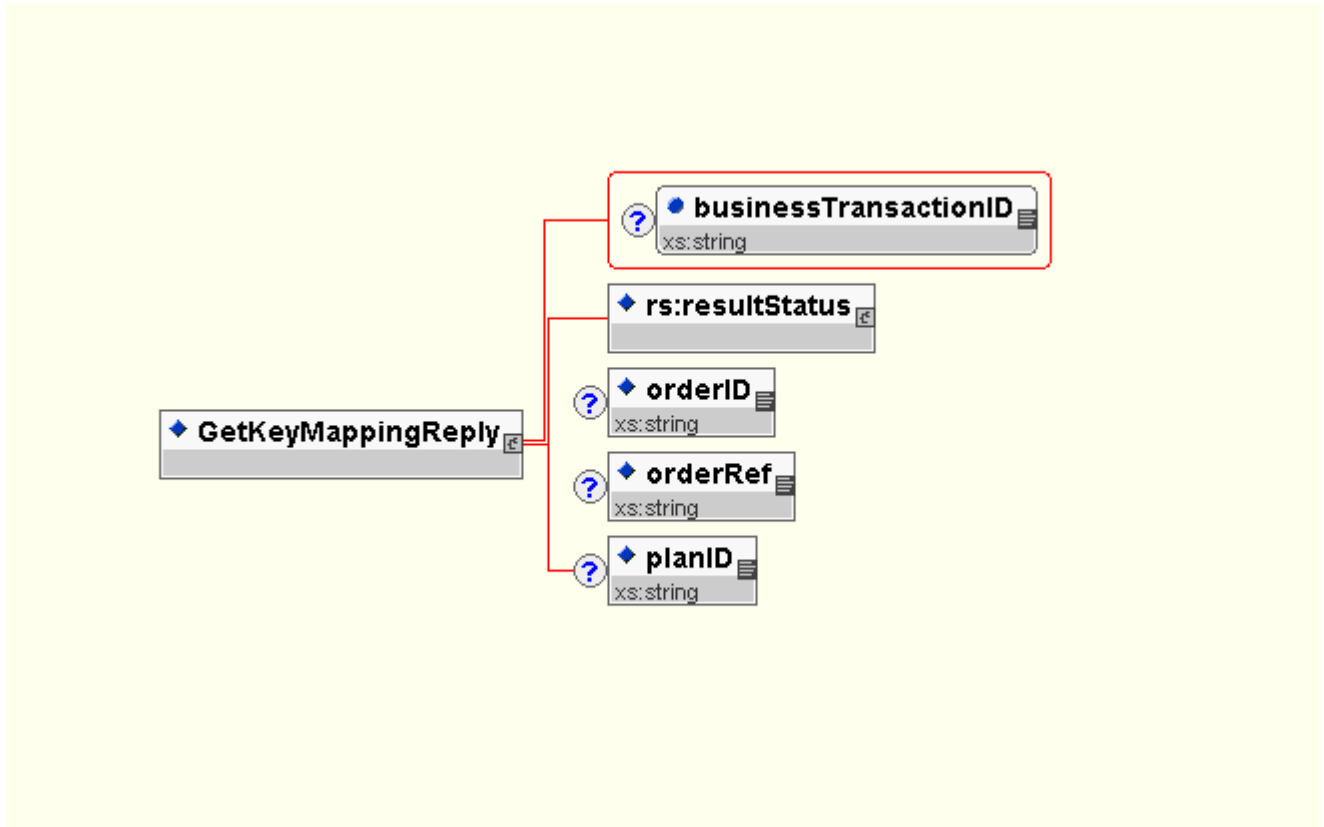
The response message contains the following header properties:

Element	Type	Cardinality	Description
businessTransactionID	String	Optional	The unique identifier for tracing across function calls. This is used for logging.

correlationID	String	Optional	The unique identifier for tracing purposes across a single function call. This is generally used by the calling application to correlate requests and responses.
success	Boolean	Required	The flag indicating if the call was successful.
messageCode	String	Required	The result message code.
message	String	Required	The result message.
orderID	String	Required	The internal unique identifier for the order.
orderRef	String	Required	The external unique identifier for the order.
planID	String	Required	The internal unique identifier for the plan for the order if it exists.

The response message contains the XML payload according to the following schema:

Get Order Response



The following table lists the details of the elements.

Element	Type	Cardinality	Description
---------	------	-------------	-------------

businessTransactionID	String	Optional	The unique identifier for tracing across function calls. This is used for logging.
resultStatus	Type	Required	The result status type. See Appendix A for the specification of this type.
orderID	String	Required	The internal unique identifier for the order.
orderRef	String	Required	The external unique identifier for the order.
planID	String	Required	The internal unique identifier for the plan for the order if it exists.

Get Key Mapping Messages and Message Codes

The error codes and their respective error messages for the Get Key Mapping interface are as follows:

Get Key Mapping Messages and Message Codes

Message Code in Response Header and Result Status	Message in Response Header and Result status	Scenario
FOM-DATA-ACCESS-SUCCESS-0000	Successfully processed GetKeyMappingRequestEvent	Elements found and data is mapped in the response successfully.
FOM-DATA-ACCESS-INVALID-INPUT-9999	Invalid value " for input {orderID/planID/orderRef} in GetKeyMappingRequestEvent	The choice of elements in the request are empty.
FOM-DATA-ACCESS-ORDERPLANKEYMAPPING-NOT-FOUND-9999	OrderPlan Key Mapping not found for orderID/planID/orderRef <value>	Elements not found for the value in the request header.
FOM-DATA-ACCESS-INTERNAL-ERROR-9999	Internal error occurred when processing GetKeyMappingRequestEvent	Another exception occurred when processing the request.
FOM-DATA-ACCESS-DBCONN-ERROR-9999	Database connection issues occurred when processing GetKeyMappingRequestEvent	Resource failure (DB connection) issues are encountered when processing the request.

Best Practices for TIBCO Order Management - Long Running

This section covers the best practices that can serve as guidelines for building a fulfillment solution by using TIBCO Order Management - Long Running.

process component Design Guidelines

This topic talks about different guidelines that can be followed for designing and developing the process components.

Process Component Technology Selection

Process components might be implemented in any JMS-enabled technology that conforms to the interface specification in the product documentation.

Generally speaking process components can be classified by using a combination of the duration of the process component lifecycle and a description of the statefulness.

Process component duration defines how long it takes to execute all tasks in the plan item. There is no absolute number that defines a short vs. long-lived process component. Instead the duration defines whether or not the task can be amended in-flight:

- Short-lived – an in-flight process cannot be amended. When suspended by Orchestrator the process runs to completion and returns a complete response.
- Long-lived – an in-flight process can be amended. When suspended by Orchestrator the process might either run to completion and return a complete response, or it might suspend execution and return a suspend response. If a suspend response is returned, it must handle an activate request from Orchestrator later to resume execution.
- Process component statefulness defines whether or not a process component needs to persist state internally during the life of an invocation. This does not include storing data by using Order Management Server data service interfaces, which is available for all process components. Instead, the determining requirement is whether or not state is stored directly within the process component:
 - Stateless – short-lived process component that is invoked and does not require state persistence.
 - Stateful – short or long-lived process component that is invoked and does require state persistence.

The choice of stateless or stateful process component is not necessarily determined by whether the backend systems being invoked are synchronous or asynchronous. Asynchronous backend system invocation is a common use case for stateful process component. However, it might be possible to pass a correlationID through the backend system that lets the use of a stateless process component. Consequently, a process component is defined as a logical entity that provides a given set of functionality. It does not necessarily translate directly into a single physical process that is invoked and runs to completion.

If a process component requires manual interaction then it is in almost all cases be defined as stateful.

Technology Recommendations based on Requirements

Some technology recommendations for a given set of conditions are as follows:

Requirement	Technology
Short lived process that does one or many synchronous invocations to back-end systems.	BusinessWorks BusinessEvents

Requirement	Technology
Short-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems. The back-end system accepts a correlation ID that comprised of the combination of orderRef and planItemID.	BusinessWorks BusinessEvents
Short-lived process that does one or many asynchronous invocations to back-end systems.	BusinessWorksBW6 Java
Short-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems. The back-end system accepts a correlation ID but cannot be comprised of orderRef and planItemID.	BusinessEvents
Long-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems with no manual interaction.	BusinessEvents
Long-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems with at least one manual interaction.	iProcess
Long-lived process that does one or many asynchronous invocations to back-end systems.	Java

Technology Recommendations based on Requirements

Some technology recommendations for a given set of conditions are as follows:

Requirement	Technology
Short lived process that does one or many synchronous invocations to back-end systems.	BusinessWorks BusinessEvents
Short-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems. The back-end system accepts a correlation ID that comprised of the combination of orderRef and planItemID.	BusinessWorks BusinessEvents
Short-lived process that does one or many asynchronous invocations to back-end systems.	BusinessWorksBW6 Java

Requirement	Technology
Short-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems. The back-end system accepts a correlation ID but cannot be comprised of orderRef and planItemID.	BusinessEvents
Long-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems with no manual interaction.	BusinessEvents
Long-lived process that does one or many synchronous and/or asynchronous invocations to back-end systems with at least one manual interaction.	iProcess
Long-lived process that does one or many asynchronous invocations to back-end systems.	Java

BusinessWorks - Asynchronous process component

This section provides an example on how to approach the implementation of a Business Works process component that is “asynchronous”, in the sense that the back-end call is asynchronous. After a call is made to a backend system, the sending process terminates, after passing in a correlation ID that the receiving process can use to retrieve any context information needed to continue with the processing required. In this example, only one back-end call is made from the process component.



All below mentioned diagrams are created in tibco designer which are supported by BussinessWorks 5.X. Now we support BussinessWorks 6.x and hence all the relevant activities are available in BussinessWorks studio to develop process-component.

Use Case Scenario

A telecommunication company wants to provide a new service for new customers and between all the Use Cases identified there is the Use Case: UserAccount.

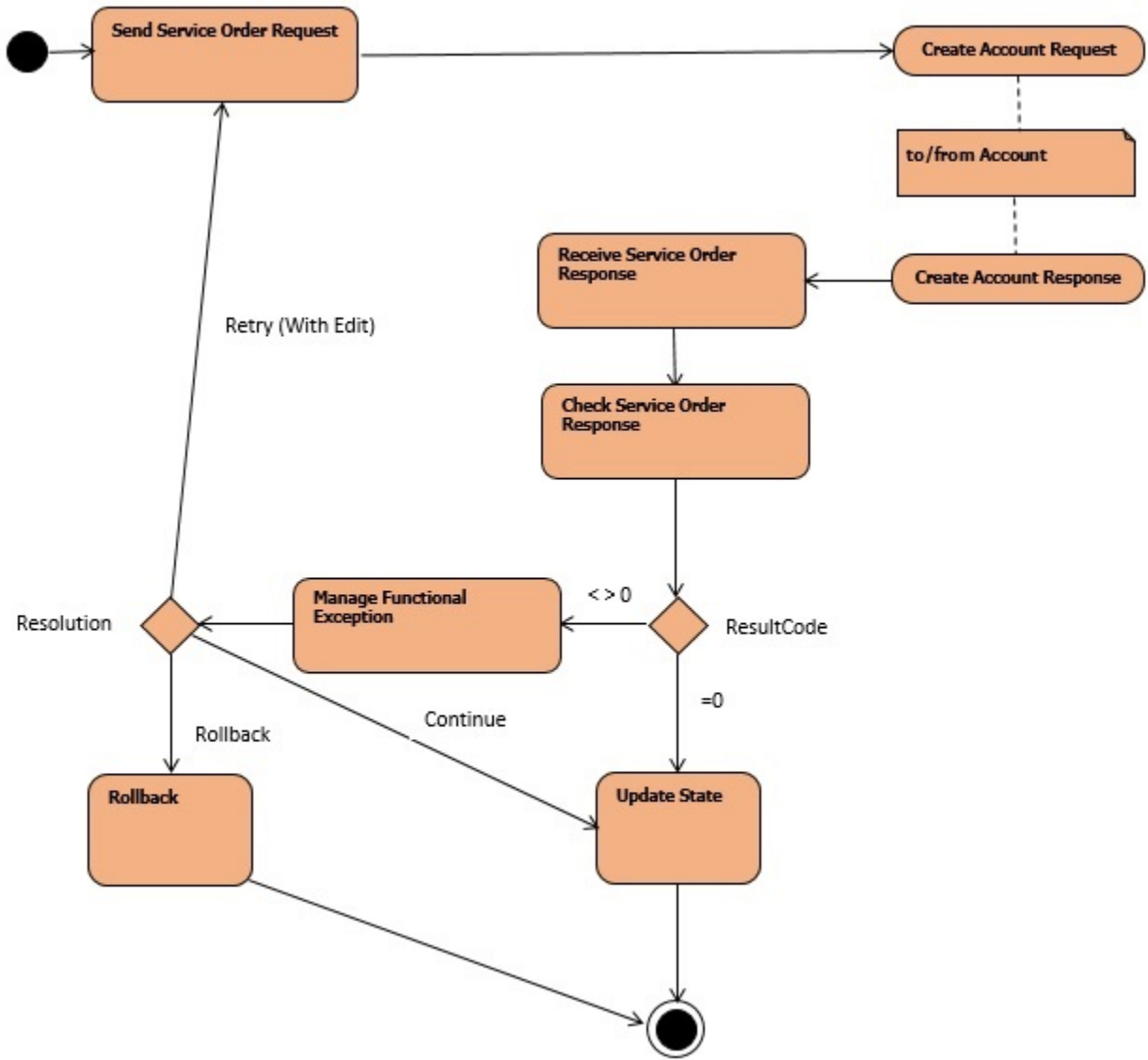
This Use Case creates a new user account by calling a backend application and passing it all the information related to the user. The backend application has an asynchronous interface and after receiving a request sends a reply back to the caller.

Let's assume that the Use Case has got:

- A Functional Product (FP) called: UserAccount
- A Technical Product (TP) called: Account, which is related to UserAccount via a Product-Comprised-Of relationship. To act on the Technical Product Account, an asynchronous backend call is required.

Below there is an Activity Diagram, which describes the steps to be performed to create Technical Product Account.

process component Example - Use Case Activity Diagram



Account Implementation

As described in the previous section, there is one Functional Product UserAccount and one Technical Product Account.

Once the order for the Product UserAccount has been submitted to Order Management Server, Automated Order Plan Development generates the PLAN and the Orchestrator sends a PlanItemExecutionRequest Event with the plan item to be executed based on the sequence defined in TIBCO Product and Service Catalog when the product has been modeled.

The PlanItemExecuteRequest message is received in a main BW process as shown in [process component Example - Use Case Activity Diagram](#) in the process component implementation. This process sends a message on a queue, which is internal to process component implementation.

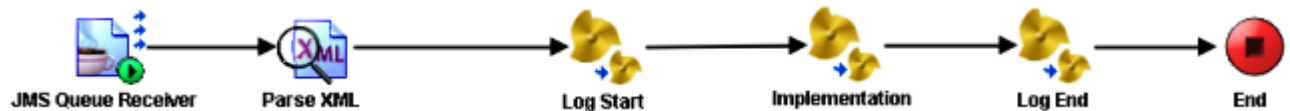
process component Example: Receive PlanItemExecutionRequest from Orchestrator





Most of the example diagrams for the BW based process components are showing BE Send Event and BE Receive Event pallets for interaction with Orchestrator. This was possible with the 1.2 version of TIBCO Active Fulfillment due to the availability of the BE event references in Orchestrator, exposed through a project library. However from TIBCO Order Management - Long Running 2.0.x version onwards, the project library usable in the Designer projects cannot have the BE event references from Orchestrator. Therefore BW based process components must use the corresponding JMS based pallets such as JMS Queue Receiver and JMS Queue Sender to interact with the Orchestrator by exchanging the required request/response messages.

process component Example: Send Backend Request

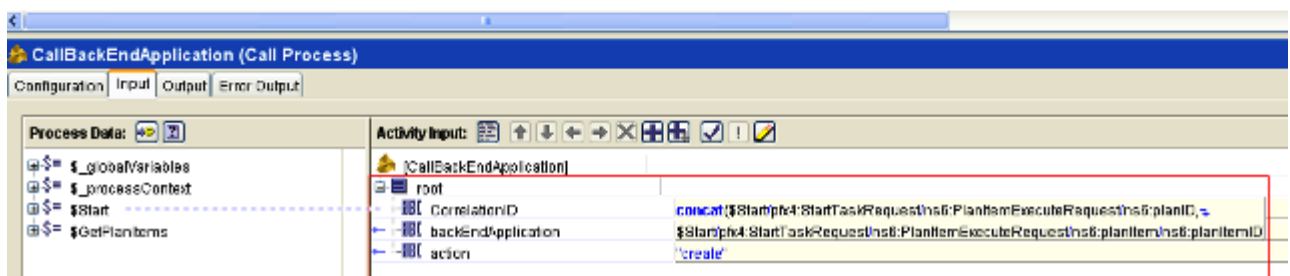


The process in [process component Example: Send Backend Request](#) receives the message from the internal queue and calls the specific BW process that implements the actual process component as mentioned in [process component Example: Backend application call](#).

As mentioned in the previous section, the backend interface is asynchronous so when the process component sends a request to the backend application, it also has to send a CorrelationID, which needs to be returned with the response from the backend application to correlate it with the request. In this example it is assumed that the Backend is capable of doing this.

This can be observed in the [process component Example: Backend application call](#) that represents the “Create Account” process component, which makes a call to the backend application:

process component Example: Backend application call



In the [process component Example: Backend application call](#) it is possible to notice that:

1. The CorrelationID is passed in the call of the CallbackEndApplication.
2. BackendApplicationName is: “UserAccount”.

3. Action is “create”.

Regarding, which CorrelationID to use, one possibility is to use a concatenation of PlanID-PlanItemID. OrderRef or Order ID can also be used instead of PlanID.

PlanID and PlanItemID can be used to set a user-defined field by using Order Management Server data access interfaces when the process component sends the request to the backend application and this user-defined field can either contain:

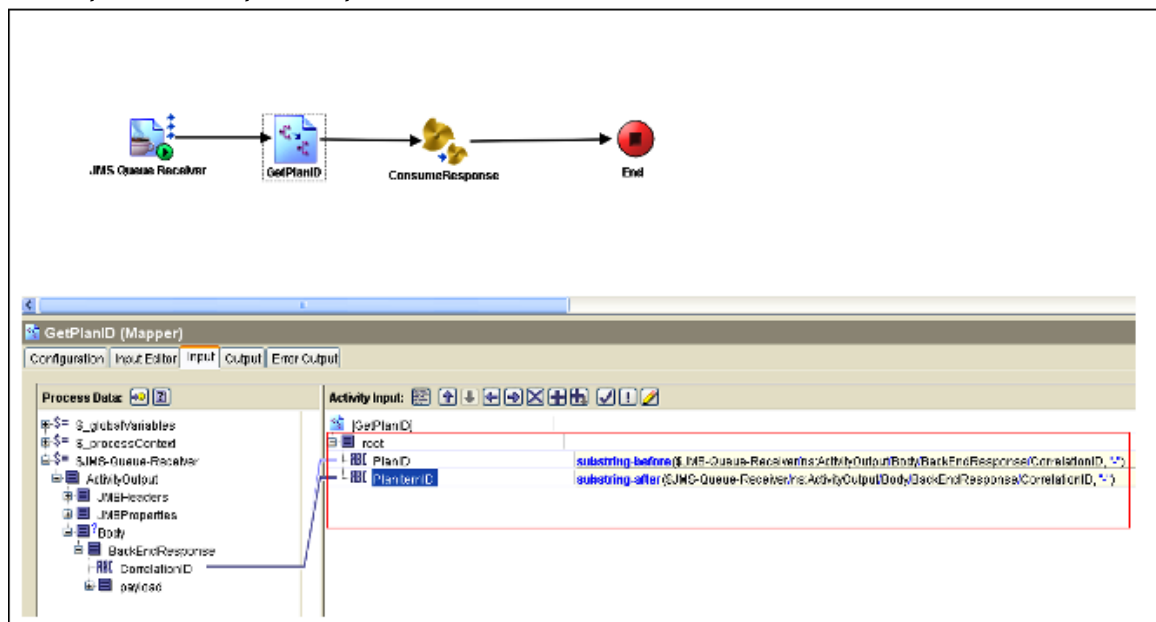
1. A BW process name, which deals with the response.
2. A unique queue name, which is used to send a message into it to trigger the process that deals with the response. This means that each BW process that consumes the reply from the backend application has to have a unique queue name.

Let's assume that there is one BW process that receives all the replies from the backend application and it works as a dispatcher of the messages amongst other BW processes that are the consumers of the replies coming from the backend system.

When the backend application replies it inserts the CorrelationID in the response, which can be used to retrieve information by using Order Management Server data access interfaces and the particular user-defined field, which contains the consumer of the response.

Here it is assumed that Technical Product sends a message to a backend application in a queue and receives the reply by listening to another queue.

process component Example: BW process receiver

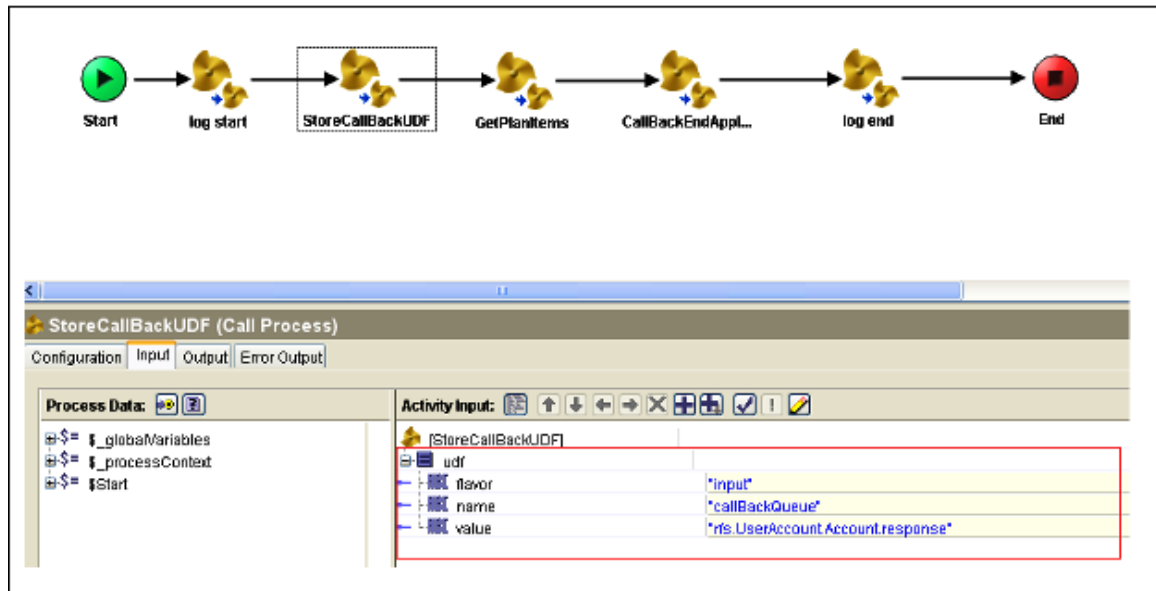


In [process component](#) above the JMS Queue Receiver activity receives the response from the backend application; it is the BW process that receives all the replies from the backend system. The “GetPlanID” mapper activity parses the ID parameter coming from the backend application and gets the planID and planItemID.

At the end of the process there is a call to a BW process: “ConsumeResponse”, which dispatches the response and it is possible to pass the planID and planItemID to it, which is used to retrieve, by using the Order Management Server data access interfaces, the user-defined field we set in the request containing the information regarding, which BW process has to be called or to, which queue it is necessary to send the reply from the backend application to elaborate the response.

The [user-defined field](#) shows how to store the user-defined field by using the “setPlanItem” JMS data access interface of Order Management Server.

process component Example: Set user-defined field for response



In the [user-defined field](#) it is possible to observe that before calling the backend application (CallBackEndApplication), a user-defined field called “callBackQueue” has been set with a queue name: “rfs.UserAccount.Account.response”. A BW process as seen in the [process component](#) listens on this queue and processes the message sent by the “ConsumeResponse” BW process as seen in the [process component](#).

Alternatively, as mentioned before, it was also possible to set the user-defined field with a BW Process name that would have dealt with the response; in this way the main BW process, which receives all the responses, would have redirected it to the BW process contained in the user-defined field by performing a dynamic call to it.

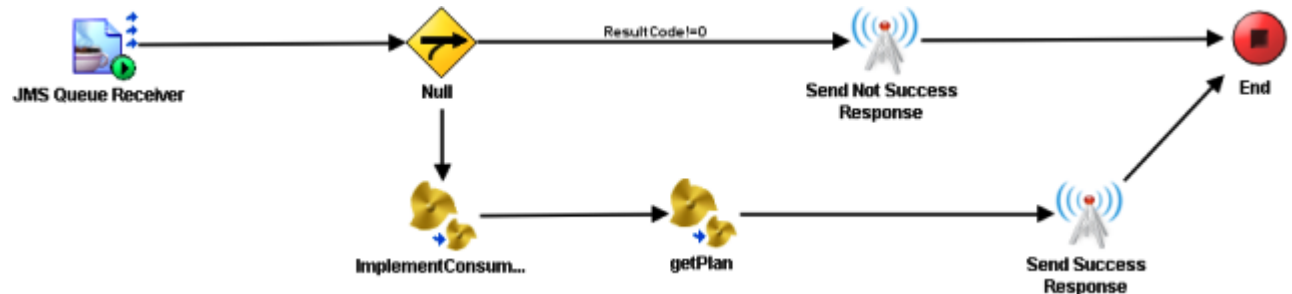
To summarize, these are the steps to be performed by the main BW process that receives all the responses from the backend application:

1. Upon the receiving of a response from the backend application, retrieve the planID/planItemID from it.
2. Use "getPlanItem" Order Management Server data access interface by using planID/planItemID to retrieve the user-defined field containing the BW process that consumes the response.
3. Make a dynamic call to the BW process contained in the user-defined field and pass the replies obtained from the backend application as an input to it.
4. If the user-defined field was set as a queue name, then send a message to the queue specified in the user-defined field. A BW process as seen in the [process component](#) listens on that queue and process the response received from the back-end application.

process component Example: BW ConsumeResponse Process

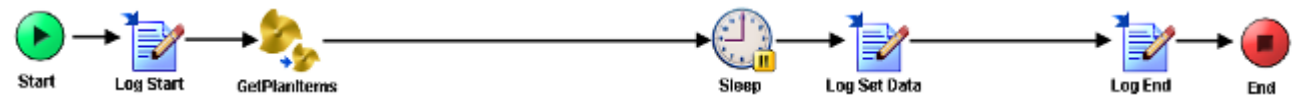


process component Example: Start Specific BW Technical Product Consumer Process



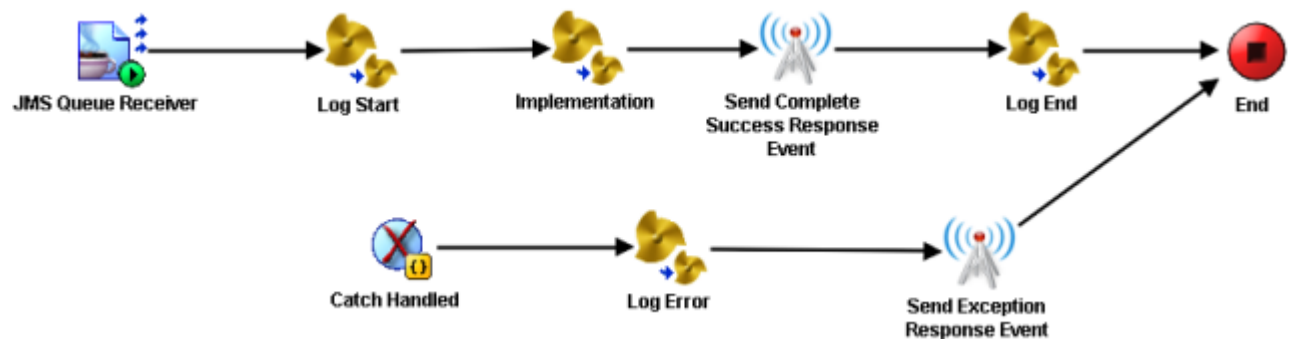
From the [process component](#) it is possible to see that if the resultCode coming from the backend application response is not equal to zero, then a PlanItemExecuteResponse with complete set to true and success set to false is sent back to Orchestrator; otherwise the ImplementConsumer BW process is called and it consumes the response from the backend application and a successful PlanItemExecuteResponse is sent back to Orchestrator.

process component Example: Example of BW process component Consumer



When the TP has been executed and the PlanItemExecutionResponse with complete and success set to true sent back to Orchestrator, this sends the PlanItemExecutionRequest for the Functional Product and the [process component Example: Functional Product process component](#) is an example of the BW process that manage the request for the Functional Product. The Functional Product process component does not make any backend calls, so is simpler than the process component for the Technical Product. There is always a call to an Implementation BW process that consume the request, it returns to the process caller and sends a successful PlanItemExecuteResponse back to the Orchestrator.

process component Example: Functional Product process component



Exception Handling Component

Having a look at the Activity Diagram shown in the [process component Example - Use Case Activity Diagram](#) it is possible to notice that when a request is submitted to the backend application, if the error code is equal to zero then the successful path flow is followed (ex: update a variable state with "Submitted") whereas if the result code is different from zero, it is possible to have a Retry/Continue or Rollback scenario.

In this section it is assumed that the process component sends/receives a message to/from a queue to communicate with a backend application.

When the backend application replies with an error code, the process component sends a PlanItemExecutionResponse Event to the Orchestrator with: completed flag set to "true", success and canceled set to "false".

Here it is necessary to analyze the different kinds of Exception Handling one by one:

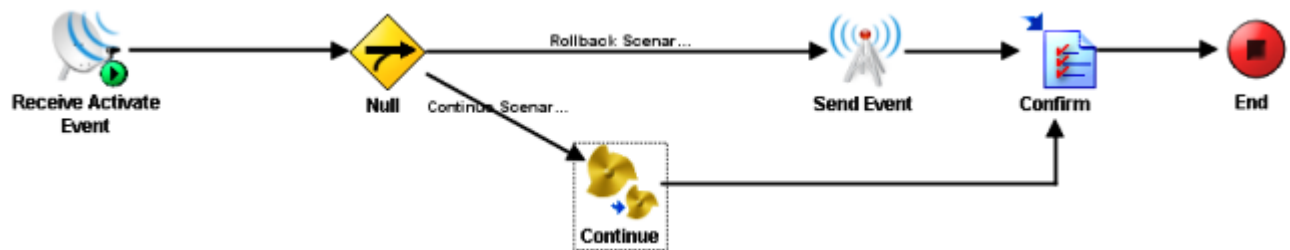
1. Retry with Edit

The process component made a call to the backend application, which replies with an error. Orchestrator then communicates with the Exception Handler (see [Exception Handling Guidelines](#) for more information) to see what the next action is. The Exception Handler has determined that the required action is that the data on the order needs to be edited, and the plan item step retried. The Exception Handler communicates with Orchestrator and asks it to resend the planItem, which failed. Orchestrator resends another PlanItemExecutionRequest for that process component.

2. Continue

In this case, the Exception handler returns to Orchestrator a Resume response. On receipt of the resume, Orchestrator sends an activate message to the process component, with a flag (<resumeExecution>) to indicate that processing must resume. In this case the process component executes the next step that would have normally executed in a success case scenario, for example: update the installed base. The [process component Example: Activate Event](#) shows a BW process that implements the Continue scenario:

process component Example: Activate Event



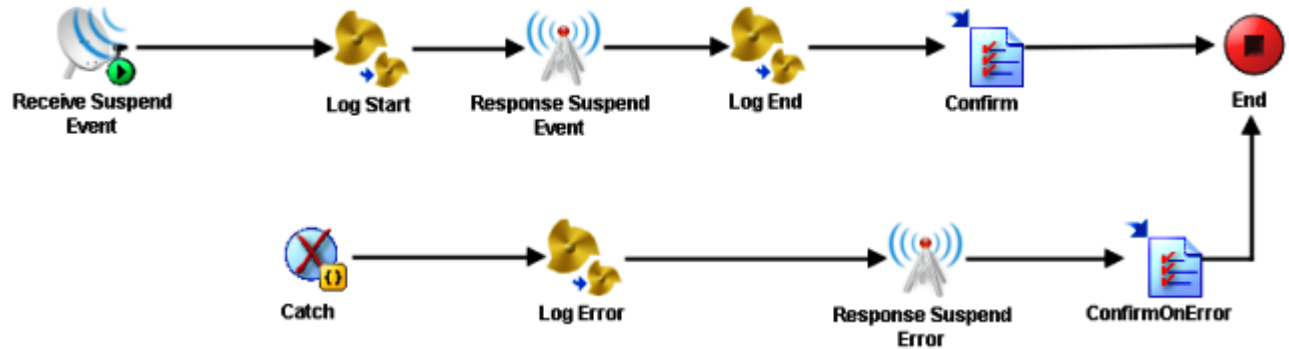
When the process component receives the Activate Event(PlanItemActivateRequest Event) message and the action is not “CANCEL” it means that it has to Resume and carry on with the next step, which in the [process component Example: Activate Event](#) is called the BW process that represent the continue step.

3. Rollback

In case of Rollback, the following steps is executed:

- A Cancel Order is sent to Order Management Server by the Exception Handler.
- Orchestrator sends a PlanItemSuspendRequestEvent to the process component as shown in the [process component Example: Suspend Event](#). The suspend event is sent because a plan in execution state needs to go into suspend state before being compensated.
- The process component replies with a PlanItemSuspendResponseEvent and sets success and completed flags to “true”.
- Cancel order is treated as a special case of amendment. An updated plan is created by Automated Order Plan Development to cancel all the existing plan items. Compensation plan items that are required against some of the existing plan items are also added in the amended plan so as to actually roll back the tasks there were done by the existing plan items.
- Once the amendment plan is received by the Orchestrator from Automated Order Plan Development, the Orchestrator sends PlanItemActivateRequest to all the suspended plan items, for canceling them, by passing the <cancelWithNoRollback> flag.
- Once the activated plan items are successfully canceled, the corresponding compensating plan item (COMP) that were newly added in the amendment plan, is executed so as to roll back the tasks that were done by the original plan items.

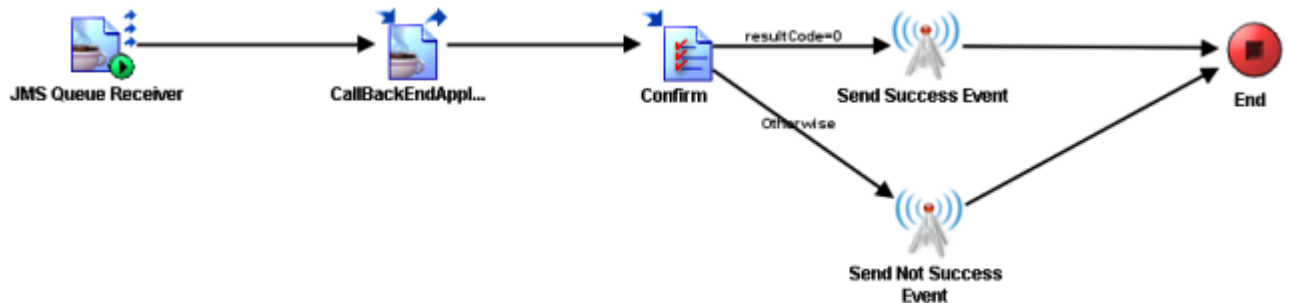
process component Example: Suspend Event



BusinessWorks - Synchronous Process Component

If the backend interface is synchronous, it is possible to implement the process component in a much simpler way. Of course, be aware of the blocking nature of synchronous calls and the impact this can have on performance.

Process Component Example: Simple Synchronous BW component

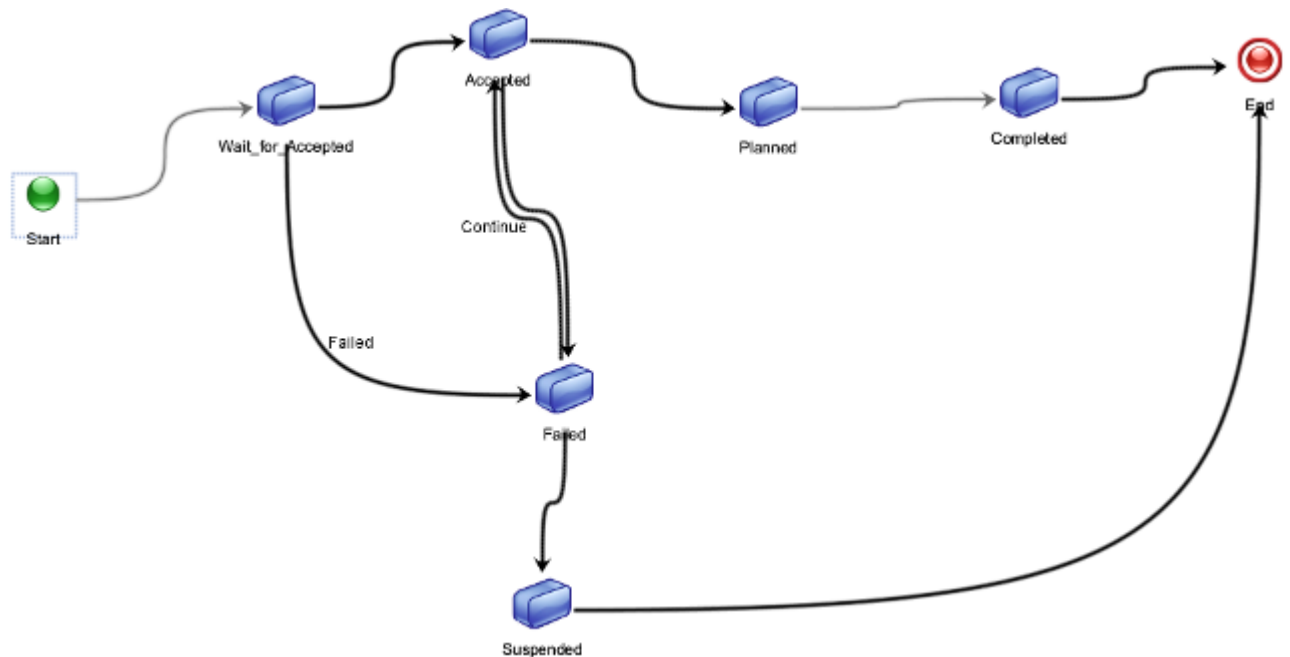


In the [Process Component Example: Simple Synchronous BW component](#) it can be seen that a queue requester activity is used to implement the synchronous call to the back-end application and once the response is received from it, a `PlanItemExecuteResponse` is sent to Orchestrator based on the `resultCode`. If the `resultCode` is equal to zero, it sends a `PlanItemExecuteResponse` with `success` and `completed` equal to "true" otherwise it sends a `PlanItemExecuteResponse` with `completed` set to "true" and `success` set to "false". This triggers the Exception Handler process as described above.

BusinessEvents - Process Component

In case of a BE Process Component it is necessary to create a Concept with a State Machine, which represents the process component lifecycle.

Process Component Life Cycle



The concept can be created in a rule that fires on receiving `PlanItemExecutionRequest` event from Orchestrator to start the Process Component.

This is a good solution in case in the project there are only BE Process Components. In case there are BW and BE Process Components then the rule has to fire only when the Process Component type is BE. To implement this, it is possible to create another channel linked to the `PlanItemExecutionRequest` Event having a selector such as: `processComponentType = 'BE'`. In this way the channel picks up `PlanItemExecutionRequests` coming from the Orchestrator and having BE as `processComponentType`.

The [planItemExecuteRequest \(Destination\)](#) shows how to set the Selector for a BE Channel:

planItemExecuteRequest (Destination)



planItemExecuteRequest (Destination)	
Configuration	
JMS	
Name:	planItemExecuteRequest
Description:	
Default Event:	/atm/orchestrator/events/execution/PlanItemExecuteRequestEvent.event
Serializer/Deserializer:	com.tibco.ccp.driver.jms.serializer.TextMessageSerializer
Queue:	<input checked="" type="checkbox"/>
Name:	%%aff4/cammon/messaging/jms/GLB_MessagingPrefix%%.%%aff4/orchestrator/messaging/jms/destinations/GLB_PlanItemExecuteRequest%%
Selector:	processComponentType = 'BE'
DeliveryMode:	PERSISTENT
AckMode:	EXPLICIT_CLIENT_ACKNOWLEDGE(TIBCO Proprietary)
Priority:	4
TTL:	0
DurableSubscriberName:	

In this case then, to create the BE Concept when the Orchestrator sends the `PlanItemExecutionRequest` Event for a certain `PlanFragment`, it is possible to create a rule function having in the declaration the

PlanItemExecutionRequest Event and in the body the code to create the concept that represents the Process Component.

In the [Arguments for PlanItemExecuteRequestEvent](#) there is the Argument of the Rule Function that creates the BE Concept when the PlanItemExecuteRequestEvent comes:

Arguments for PlanItemExecuteRequestEvent

Arguments	
Type	Identifier
/aff/orchestrator/events/execution/PlanItemExecuteRequestEvent	planitemexecuterequestevent

In the [Rule Function Code](#) there is the body of the Rule Function with the code example that checks first if the Concept that needs to be created already exists and if not, it creates the Concept:

Rule Function Code

```

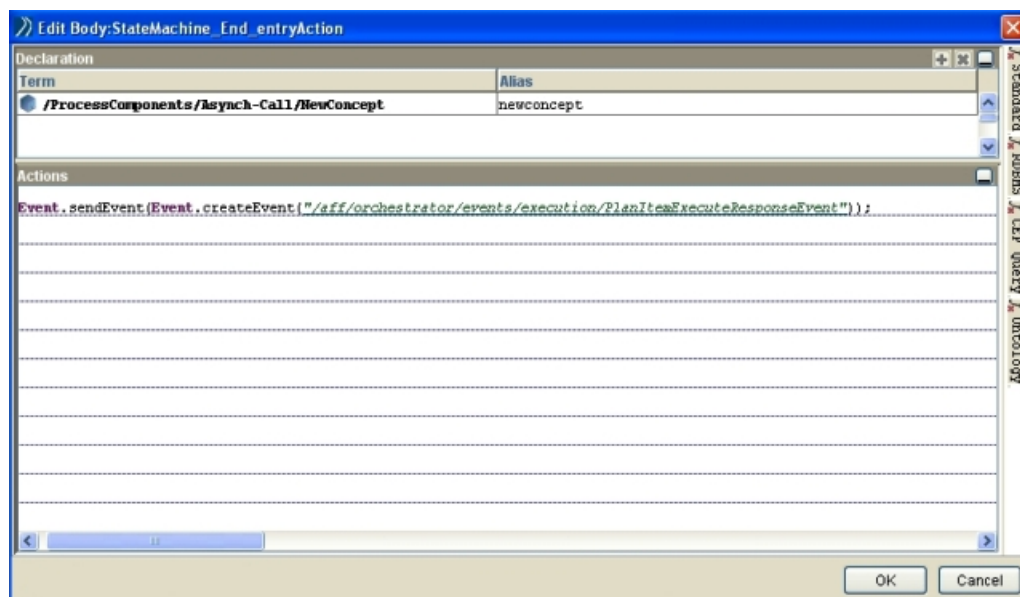
if (req.processComponentID == "TP/UserIntranetAccess/create") {
    Concept c=Instance.getByExtId(planItemID);
    if(c!=null)
    {
        System.debugOut("The value of the concept is : " + Instance.serializeUsingDefaults(c) + "the planitem is" + planItemID);
        Instance.deleteInstance(c);
        OrderPlanExecution.ProcessComponents.be.concepts.userIntranetAccess_create.userIntranetAccess_create();
    }
    else
    {
        OrderPlanExecution.ProcessComponents.be.concepts.userIntranetAccess_create.userIntranetAccess_create();
    }
}

```

Once that the Concept has been created, it is also necessary to send the PlanItemExecuteResponseEvent back to the Orchestrator. This can be done in any state of the State Machine based on the logic of the implementation or alternatively at the END state.

In the [Edit Body: StateMachine_End_entryAction](#) there is the code example that shows how to create the PlanItemExecuteResponseEvent and how to send it:

Edit Body: StateMachine_End_entryAction



Exception Handling Guidelines

Exception Handling Guidelines provides information about guidelines that can be followed for handling the exceptional conditions in process components.

General Approach

TIBCO Order Management - Long Running does not include any out-of-the-box approach for error handling. The product architecture does account for exception handlers, and provides the necessary hooks, where it can be integrated with an existing exception or fallout management system, or to which a custom solution can be connected.

The product capabilities for supporting error handling are fully described in the product documentation, and it is assumed that the reader is familiar with that document. The basics is not covered here.

Plan Item Failed Handler or no Plan Item Failed Handler

A key question is whether to handle exceptions within the process component itself, or whether to manage exception handling via Orchestrator and the Plan Item Failed (PIF) handler. In the first case, process components must only return a success result to Orchestrator, and no Plan Item Failed handler is required.

In the second case, it is necessary to develop a Plan Item Failed handler that receives `PlanItemFailedRequest` from the Orchestrator for direction on how to proceed once an error is encountered. The Plan Item Failed handler must respond to Orchestrator telling it whether to Retry the plan item, or Continue (that is, mark the plan item as completed and continue with the plan).

A consideration here is the type of process component. If a process component makes use of a workflow engine for its implementation, which already includes manual tasks and GUI interaction, it would make sense for errors in the flow to be managed within the workflow engine, rather than have them passed back to Orchestrator. If the process component is BW or BE, the Plan Item Failed handler might be a better option.

Functional Exception against Technical Exception

Any consideration of exception handling must consider the different types of exception that can occur, which typically fall into two broad categories, Functional exceptions, and Technical exceptions. For the purposes of this discussion, we define these as follows:

- A *Functional Exception* occurs when a back-end system returns an error code to the process component, indicating a problem with the request. Functional exceptions always occur in the context of a process component. An example could be a request to allocate a phone number that is already in use. Receiving a Functional Exception is expected to occur under normal circumstances, and the system is built to handle these events.
- A *Technical Exception* occurs when something goes wrong, so that the system is not designed to handle under normal circumstances. They can occur in process components and also TIBCO Order Management - Long Running components such as Orchestrator and Order Management Server. They are typically caused by conditions in the external environment, such as running out of memory, failure in Enterprise Message Service, and hardware failure, but can also be caused by defects.

Different strategies might be considered for each of these types. For instance, as Functional Exceptions occur within the context of a process component, and would typically require an operator to review and decide on remedial action, it makes sense for these to be handled via the Orchestrator and a Plan Item Failed handler, which might defer to an external GUI for a resolution.

Technical exception handling is more difficult, as they can be caused by almost anything. In this case, even if a Technical exception occurs in a plan item, it might make more sense to simply log it and stop execution of the plan item.

Example Approach

This topic describes an approach for implementing exception handling, where order fallout is managed externally to the TIBCO Order Management - Long Running implementation. This is a good approach where the customer site has an existing order fallout management system, providing GUI windows, and so on. whose functionality can be leveraged. In the rest of this topic we refer to this external error handling

system as Exception Management, or EM. Please note this is an example only, and might not be applicable for your particular circumstance.

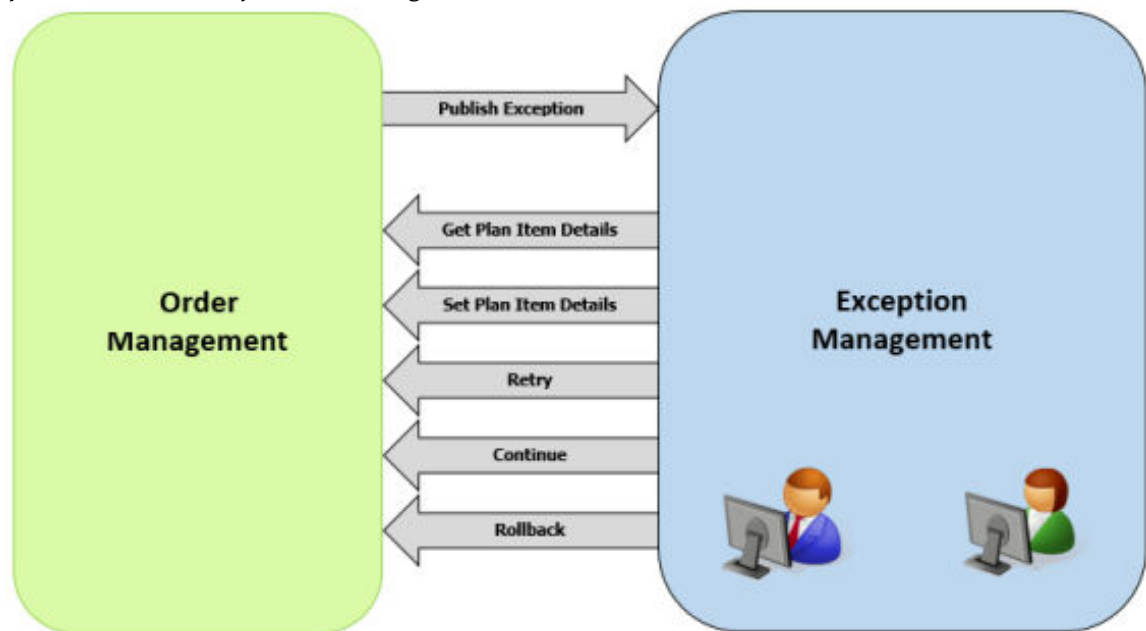
In this case, Functional exceptions are managed via the Plan Item Failed handler, and Technical exceptions via a separate mechanism.

For Functional exceptions the requirements are to forward all to Exception Management, for an operator to analyze. The possible resolutions are:

- Retry the Plan Item step, with the possibility to edit input values for the downstream call that failed. Note this is different to retrying the plan item from the beginning. Some process components could internally be orchestrating multiple steps.
- Continue the Plan Item, with the possibility to edit output values from the downstream call that failed (note this might not be quite the same as the Complete Plan Item Failed handler response, which instructs Orchestrator to complete the plan item. There might be activity that we require the process component to complete after the downstream call but before it completes).
- Rollback the entire order, performing compensating actions if required.

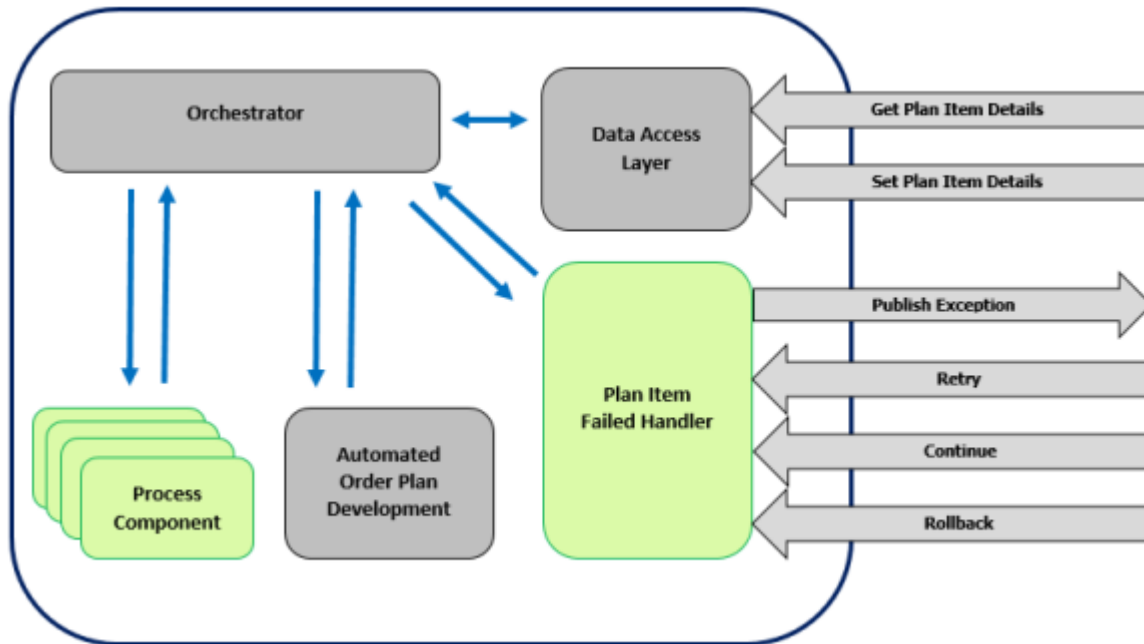
The architectural approach here is to define a clear separation of concerns, between what TIBCO Order Management - Long Running does, and what Exception Management does. The following diagram shows the approach in the case of Functional exceptions. Also, the data access GetPlanItem and SetPlanItem calls are used to support the functionality of editing input/output values.

Example Functional Exception Handling Overview Architecture



The [Example Functional Exception Handling TIBCO Order Management - Long Running Components](#) shows an approach for how this could be implemented within the application:

Example Functional Exception Handling TIBCO Order Management - Long Running Components



Plan Item Failed Handler

In this example, the Plan Item Failed (PIF) Handler is built as a pass-through component. It performs the following:

- On receipt of a `PlanItemFailedRequest` message from Orchestrator, publishes a message (to Exception Management).
- On receipt of a “Retry” or “Continue” resolution type from Exception Management, creates a `PlanItemFailedResponse` message and sends to Orchestrator with an appropriate flag that is either retry, resume, or complete.
- On receipt of a “Rollback” resolution type from Exception Management, send a message to Order Management Server to cancel the entire order. No `PlanItemFailedResponse` message is sent to Orchestrator in this case.

Process Component Considerations

When mapping the selected resolution type to a `PlanItemFailedResponse` to send to Orchestrator, there are some considerations regarding this and the nature of the process component implementation, i.e. whether it executes multiple steps, or is atomic.

For process components that implement multiple steps (Example: a BE process component):

- A Retry action means the entire process component is re-executed. If what is required is just the current step to be retried, a Resume action must be specified, not retry.
- A Complete action means the process component is not invoked again in any way, and the plan item is marked as complete.
- A distinction needs to be made between a Resume where the current step needs to be retried, or skipped. There is no way to do this on the `PlanItemFailedResponse` message, so this needs to be managed another way, Example: by setting a user-defined field on the plan item to indicate, which is required.

Pre-Qualification Failed Handler

Like the Plan Item Failed handler, there is no default implementation of the Pre-Qualification Failed handler provided with the product.

Be aware that the Pre-qualification failed handler deals with errors raised not only in Feasibility, but also, Automated Order Plan Development. Even if in your architecture you don't have a Feasibility step, you have Automated Order Plan Development, and if Automated Order Plan Development raises exceptions, Orchestrator publishes an event to the Pre-Qualification Failed handler and wait for a response. If there is no Pre-Qualification Failed handler implemented, nothing further happens for that order and it is "stuck".

Even if Automated Order Plan Development exceptions are expected to be rare for your application (i.e. you validate the order thoroughly prior to Automated Order Plan Development), consider at the very least, implementing monitoring on the Pre-Qualification Failed request queue, so that operations is aware that Automated Order Plan Development has failed for an order, and some action needs to be taken to move the order on.

You might want to consider making the Pre-Qualification Failed handler "just another" source of Technical Exceptions. In this way, a framework for dealing with automating Technical Exception handling, could be used to also deal with Pre-Qualification Failed requests. This is the approach that is described in the next section.

Technical Exception Handling

For Technical exception handling, it is difficult to define a pattern that always can apply, given the diverse range of possible exceptions that can be raised. Such exceptions can be raised from anywhere – TIBCO Order Management - Long Running components, process components, and any other code that might be developed as part of a total fulfilment solution.

It is of course always good general software engineering practice to build components as resilient as possible to errors. It might make sense, depending on the context, to build in mechanisms such as retry, when events such as timeouts occur. Of course, this needs to be weighed up against the additional complexity this introduces into the solution, and needs to consider the idempotency of interactions. Complex, built-in "self-healing" type mechanisms themselves increase the risk of coding defects, increase the complexity of testing, and is never be able to catch all types of errors.

The underlying principle here is, as with Functional exceptions, Technical exceptions require manual inspection to determine what to do. The default approach is that all technical exceptions are also dealt with manually. This can mean messages being manually copied from one queue to another, database entries being manually edited, and so on.

Nevertheless, it is highly desirable, in some common circumstances, for a fulfilment system to be able to resolve technical exceptions in an automated way. No system can be built to automatically resolve all exceptions, however one approach is to build a mechanism that can support incremental addition of automated technical exception resolutions, as the system evolves and experience is gained into the types of exceptions that can occur, and how best to deal with them. This section outlines a possible approach to building such a mechanism.

As with the handling of Functional exceptions, it is important to define a clear architectural separation between the fulfilment system and the system that determines what to do with exceptions. Again, we term this body Exception Management, see [Technical Exception Handling Architecture Overview](#).

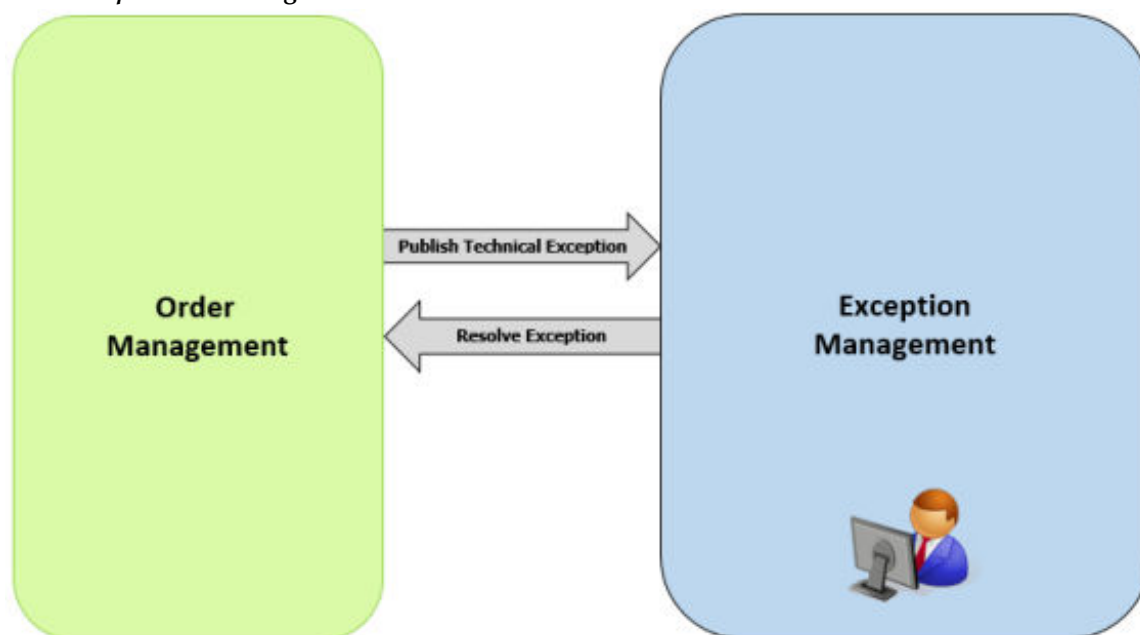
To simplify the interface, we define here a single "Resolve Exception" service, which is used to resolve all Technical exceptions. It expects an argument "Resolution Type", which is a string that defines what specific resolution behavior is required.

It is good practice when building custom components of a fulfilment solution, such as the process components, any database adapters, and enrichment processes, to ensure Technical exceptions are caught and logged in a consistent way. We define a single "Publish Technical Exception" service for TIBCO Order Management - Long Running to use when publishing a Technical exception. This same service would be

invoked regardless of the source of the Technical exception, which can be custom code, TIBCO Order Management - Long Running internal components, or even a Pre-Qualified Failed error.

When publishing an exception to Enterprise Message Service, TIBCO Order Management - Long Running needs to publish along with the exception, all the information that it would have to handle the resolution.

Technical Exception Handling Architecture Overview



Types of Technical Exception

We identify the following types of technical exceptions as candidates for automation via this approach. These are of course not the only types of Technical exception that can occur.

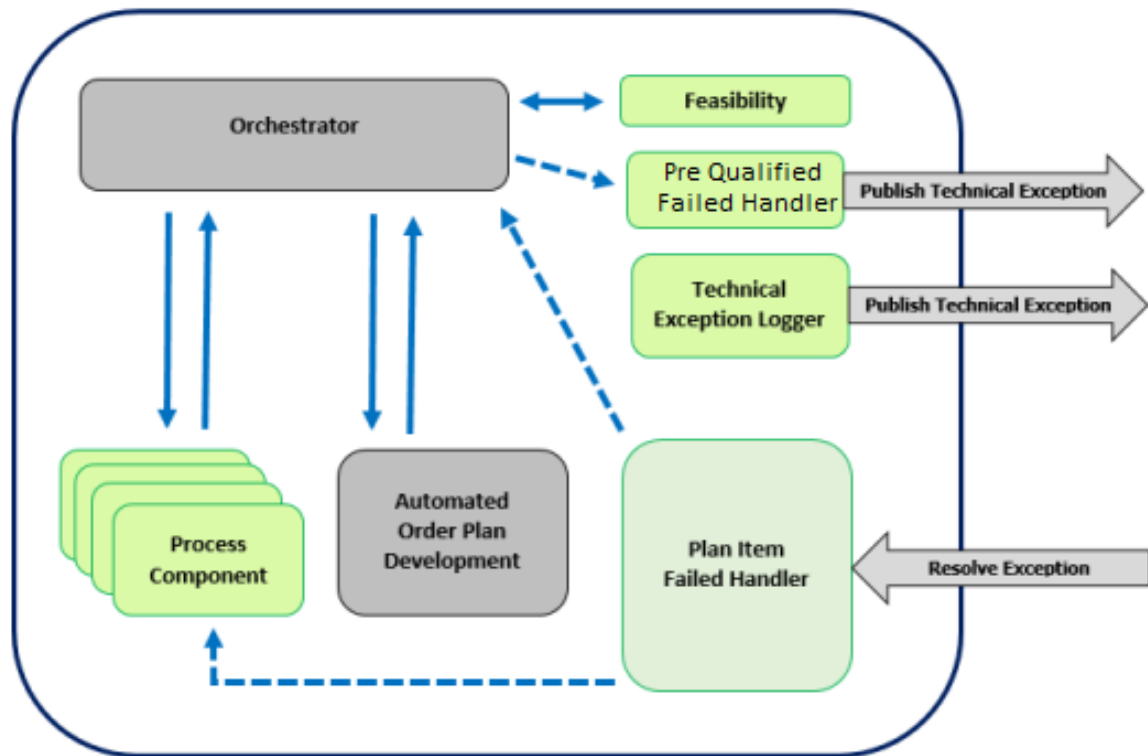
1. Pre order submit (i.e. an error that happens during any order pre-processing or enrichment)
 - a. Resubmit order only action possible – but first state needs to be cleaned from any database tables etc.
 - b. Relatively easy to automate.
2. Pre-qualification Failed Handler
 - a. If an error occurs in plan development (or Feasibility, if present).
3. Process Component
 - a. Most technical exceptions likely to be this type.
 - b. The Process Component could potentially be restarted (retried), continued or completed, depending on how far it has progressed.

TIBCO Order Management - Long Running Components for Technical Exception Handling

The [Components involved in Technical Exception Handling](#) outlines the components within TIBCO Order Management - Long Running that would be involved in handling technical exceptions. Note that the other components not directly involved in the solution for technical exception handling are not shown.

It must be noted however that any component within the TIBCO Order Management - Long Running can generate a technical exception. This includes those shown below, and the core components, such as Orchestrator, data access interfaces, and Automated Order Plan Development.

Components involved in Technical Exception Handling



The following outlines the required behavior of the components that have to be built, in the context of Technical Exception Handling:

Process Component

Technical exceptions occurring during the execution of process components logs a technical exception directly to Exception Management, via the Technical Exception Logger, and stop execution. Orchestrator is not notified when a Technical exception occurs, and considers the Process Component to be in “Execution” state. The process component can be retried or continued by the Technical Exception handler, or the Technical Exception handler can notify Orchestrator directly that the Process Component is complete.

Feasibility

The Feasibility step is invoked by Orchestrator after it has received and stored the order, but before it invokes Automated Order Plan Development to get the plan. Like Automated Order Plan Development, the feasibility component can return an error to Orchestrator, if Feasibility fails. Also like Automated Order Plan Development, in the context of the example, a Feasibility error is regarded as a Technical exception, as Feasibility must always pass under normal circumstances (this might not typically be true though, for instance if order validation is performed at Feasibility).

Technical Exception Logger

This component publishes Technical exceptions to Exception Management. It must capture all Technical exceptions generated from custom components, and publish them in a standard way to Exception Management. A standard set of exception fields must be published, which must include order ids (if available), the component and service that generated the error, and an error code. The message being processed at that time might also be logged. The key requirement is that there must be enough information logged to enable Exception Management to choose a resolution type to be applied to resolve the exception, and enough information to be passed back to the Technical Exception Handler for it to be able to resolve the exception.

Pre-Qualification Failed (PQF) Handler

Its role is to receive notifications that Orchestrator publishes when it receives an error from Feasibility or Automated Order Plan Development, and publish them to Exception Management.

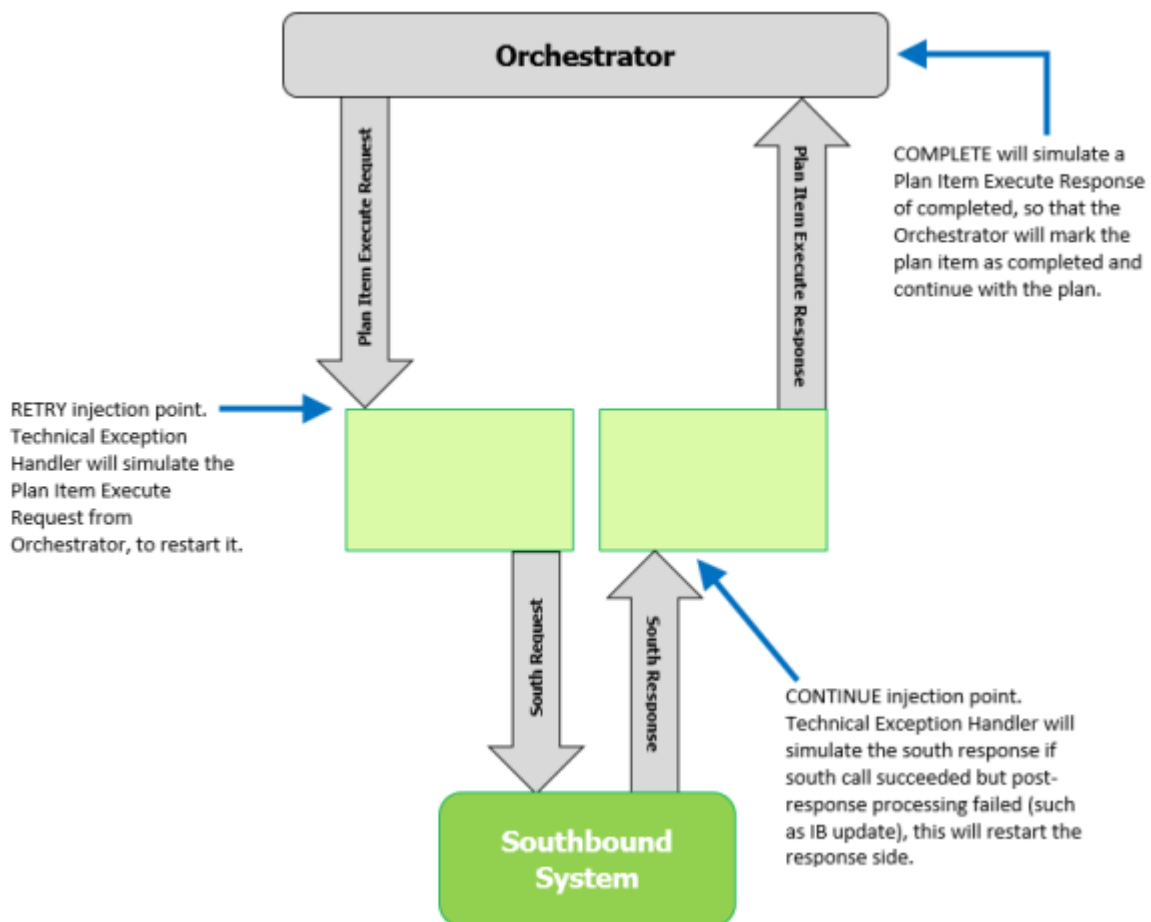
Technical Exception Handler

Its role is to expose the service for resolving Technical Exceptions for Exception Management to call, and to implement the logic for performing the resolution. This might involve sending messages to a process component, or to perform some custom action (such as clean up a database table and resubmit an order). It might also communicate directly with Orchestrator, for instance to send a Pre-Qualification Failed Response message.

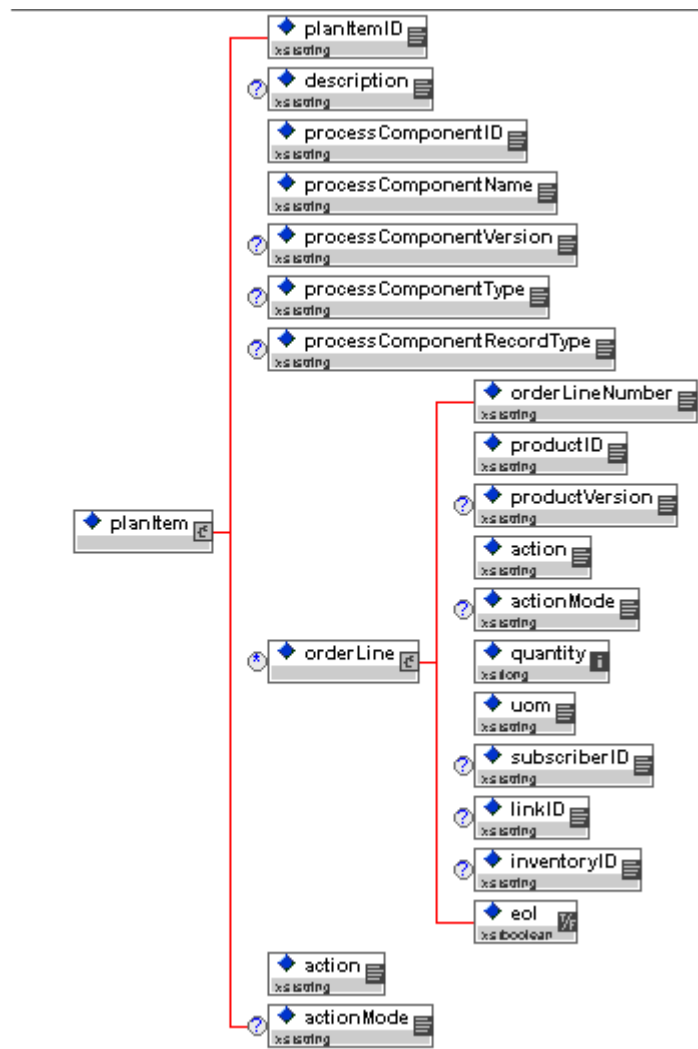
The number of resolution types this component can expose, might grow over time as new resolutions are added.

The [Process Component Technical Exception Services Overview](#) outlines at a high level, how the Retry, Continue and Complete services could potentially be implemented.

Process Component Technical Exception Services Overview



Plan Item

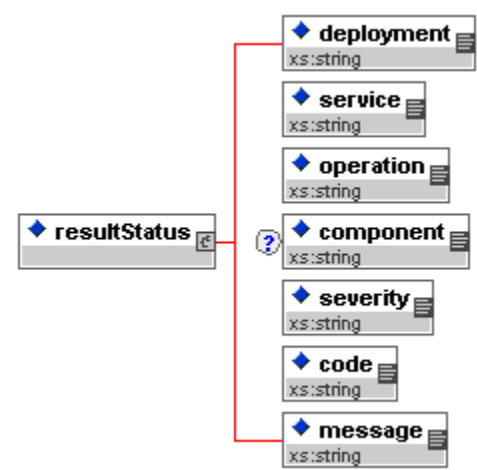


Element	Type	Cardinality	Description
planItem/planItemID	String	Required	Unique identifier for the plan item within the plan to be executed.
planItem/description	String	Optional	Description for the plan item to be executed.
planItem/processComponentID	String	Required	Unique identifier for the process 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/ processComponentRecordType	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.

ResultStatus

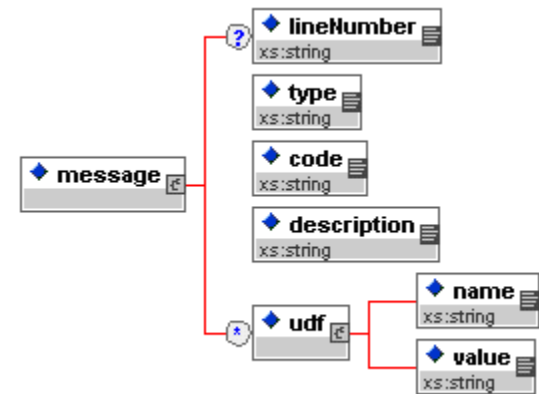
Result Status



Element	Type	Cardinality	Description
deployment	String	Required	The Order Management Server member (value of NODE_ID) that returned this result. For example, Member1.
service	String	Required	Service that returned this result. It is always returned as JMS-BASED-DATA-ACCESS-INTERFACE .
operation	String	Required	Operation within the service that returned this result. For example, GetOrder, GetPlan, or GetPlanItem.
component	String	Required	The component that returned this result. It is always returned as Order Management Server.
severity	String	Required	Severity result. Valid values are: 1. S - Success 2. E - Error.
code	String	Required	Message code for this result.
message	String	Required	Message details for this result.

Message

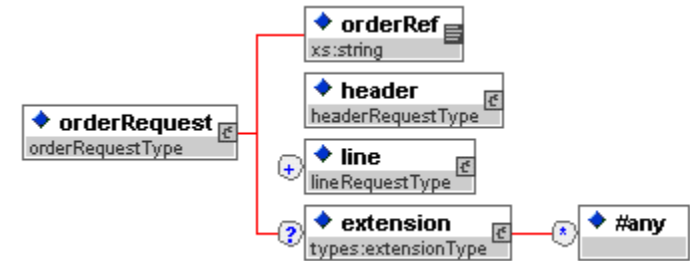
Message



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

Samples

Sample Order XML

The sample order XML is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Order Id="544">
  <orderID>81</orderID>
  <sessionID>CORRELATION-3baee8b0-b483-47aa-89b2-bf7b03d0c41f</sessionID>
  <orderlines Id="545">
    <lineID>1</lineID>
    <productID>CFS TV</productID>
    <action>PROVIDE</action>
    <quantity>1.0</quantity>
    <requiredByDate>2011-04-30T23:50:00+05:30</requiredByDate>
    <LineUsed>false</LineUsed>
    <OrderlinesUDF Id="546">
      <name>OrderRef</name>
      <value>OrderRefID</value>
      <flavor>input</flavor>
    </OrderlinesUDF>
  </orderlines>
  <orderlines Id="547">
    <lineID>2</lineID>
    <productID>CFS Live Box</productID>
    <action>PROVIDE</action>
    <quantity>1.0</quantity>
    <requiredByDate>2011-04-30T23:50:00+05:30</requiredByDate>
    <LineUsed>false</LineUsed>
    <OrderlinesUDF Id="548">
      <name>OrderRef</name>
      <value>OrderRefID</value>
      <flavor>input</flavor>
    </OrderlinesUDF>
  </orderlines>
  <orderlines Id="549">
    <lineID>3</lineID>
    <productID>CFS VOIP</productID>
    <action>PROVIDE</action>
    <quantity>1.0</quantity>
    <requiredByDate>2011-04-30T23:50:00+05:30</requiredByDate>
    <LineUsed>false</LineUsed>
    <OrderlinesUDF Id="550">
      <name>OrderRef</name>
      <value>OrderRefID</value>
      <flavor>input</flavor>
    </OrderlinesUDF>
  </orderlines>
  <status>NewOrder</status>
  <currentTime>2012-07-18T10:19:03+05:30</currentTime>
  <TineDelay>0</TineDelay>
  <customerref>Apple</customerref>
  <OrderHeaderUDF Id="551">
    <name>Company</name>
    <value>Orange</value>
    <flavor>input</flavor>
  </OrderHeaderUDF>
  <Originator>Orchestrator</Originator>
  <OrderRef>OrderRefID</OrderRef>
  <businessTransactionID>a7eb1e1de1fa45c993f65589dba70648</businessTransactionID>
</Order>
```

Sample Plan Item XML

The sample plan item is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<PlanItem Id="2169">
  <planID>PF1</planID>
  <parentID>CORRELATION-1b1260e6-9cdd-4903-a184-d473aa11b622</parentID>
  <lineID>2</lineID>
  <dependentOn>PF10.7747556</dependentOn>
  <planDesc> PROVIDE</planDesc>
  <planItemID>PF10.8786092</planItemID>
  <EOL>N</EOL>
  <TimeDelay>0</TimeDelay>
  <status>addDependency</status>
  <singleUse>false</singleUse>
  <sequence>0</sequence>
  <sequenceName>leaf</sequenceName>
  <action>PROVIDE</action>
  <productID>GSMDDataService</productID>
  <itemMark4Del>false</itemMark4Del>
  <mustComplete>true</mustComplete>
  <affinityType>ConditionalAffinity</affinityType>
  <affintyPlanID>PF1</affintyPlanID>
  <affintyPlanDesc> AFFINITY PROVIDE</affintyPlanDesc>
  <udfs Id="2170">
    <name>TASKID</name>
    <value>PF10.8786092</value>
    <flavor>config</flavor>
  </udfs>
  <udfs Id="2172">
    <name>PRODUCTID</name>
    <value>GSMDDataService</value>
    <flavor>config</flavor>
  </udfs>
  <udfs Id="2173">
    <name>RECORD_TYPE</name>
    <value>SERVICE</value>
    <flavor>config</flavor>
  </udfs>
  <udfs Id="2174">
    <name>MSISDN</name>
    <value>123</value>
    <flavor>input</flavor>
  </udfs>
  <Ancestors>PF10.7747556</Ancestors>
  <cancelUsed>false</cancelUsed>
  <M_EP_UDFS Id="2171">
    <name>M_EP_UDFS</name>
    <value>PF10.8786092</value>
  </M_EP_UDFS>
  <pI_Used>false</pI_Used>
  <isLeaf>false</isLeaf>
  <counter>0</counter>
  <LinkID>1</LinkID>
  <affinityCorrelation>$var/PlanItem[productID='GSMLine']/udfs[name='MSISDN']/value/
text()</affinityCorrelation>
  <affinityParentGroup>false</affinityParentGroup>
  <affinityActionGroup>false</affinityActionGroup>
  <isDynamic>false</isDynamic>
</PlanItem>
```

Sample XPATHs

- <ns0:affinityCondition>exists(\$var/Order/OrderHeaderUDF[name="SubscriberProduct" and value="Product BB Network Access"])</ns0:affinityCondition>
- <ns0:affinityCorrelation>exists(\$var/Order/OrderHeaderUDF[name="SubscriberProduct" and value="Product BB Network Access"])</ns0:affinityCorrelation>

- `<ns0:affinityActionValue>$var/Order/orderlines[productID='CFS STB']/action/text()</ns0:affinityActionValue>`
- `<affinityCorrelation>$var/PlanItem[productID='GSMLine']/udfs[name='MSISDN']/value/text()</affinityCorrelation>`

Global Variables

Automated Order Plan Development Global Variables

This table is a mapping that shows the global variables mapped to configuration properties.

For readability purpose, to property names are abbreviated as follow:



- c.t.a.a for com.tibco.af.aopd
- c.t.f.o for com.tibco.fom.oms

Global Variable Name	Configuration Property	Purpose
AFF/Global/Constants/InternalUDFs	Automated Order Plan Development Application Flags configuration	
UDFList	c.t.a.a.flags.udflist	Internal User Defined Fields to be skipped for affinity merging
AFF/Global/Constants/ InventoryStatus		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ InventoryUserStatus		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ LineItemActionModes		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ LineItemActions		
	NA	These properties are not required in Automated Order Plan Development

Global Variable Name	Configuration Property	Purpose
AFF/Global/Constants/ OfferValidation		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/OrderUDFs		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/PlanUDFs		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ ProductModelCharacteristics		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ProductType		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ RegularExpressions		
	NA	These properties are not required in Automated Order Plan Development
AFF/Global/Constants/ResultStatus		
	NA	These properties are not required in Automated Order Plan Development

Global Variable Name	Configuration Property	Purpose
AffinityUDFNameMerge	c.t.a.a.flags.affinity.affinityudfnamemerge	Controls the flag to merge affinity user-defined field name
CharacterisitcsWithoutAffinityPostfix	c.t.a.a.flags.affinity.characterisitcswithoutaffinitypostfix	To not merge certain User Defined Fields during Affinity Sequencing those User Defined Fields must be added as CSV in the variable
EnableParentIDUdfCheck	com.tibco.af.aopd.flags.enableparentidudfcheck	Enable the Parent ID user-defined field check for plan item equivalence.
LoadInventory		
skipItemSequence	c.t.a.a.flags.skipitemsequence	Within Automated Order Plan Development, if the sequence is -1, it skips the product and all its mandatory children in the Execution Plan
MergeAffinityItemDescription	c.t.a.a.flags.affinity.mergeaffinityitemdescription	MergeAffinityItemDescription
HierarchySingleUse	c.t.a.a.flags.singleuse.hierarchysingleuse	Flag to determine whether hierarchy child of single use product must be deleted
EnableAffinityUDFParent	c.t.a.a.flags.affinity.enableaffinityudfparent	Enable the user-defined field syntax to determine the parent product name and product name of the affinity plan item
AllowMultipleRequiredProducts	c.t.a.a.flags.allowmultiplerequiredproducts	Allow Multiple Required Products for the same link ID
IgnorePDOFirstChildDependency	c.t.a.a.flags.ignorepdofirstchilddependency	Ignore First child dependency for source product in PDO relationship

Global Variable Name	Configuration Property	Purpose
	NA	These properties are not required in Automated Order Plan Development
MIG_Password	c.t.a.a.jms.jndi.security.credentials	Password for the JMS server events.
MIG_QueueConnectionFactory	NA	These properties are not required in Automated Order Plan Development
MIG_TopicConnectionFactory	NA	These properties are not required in Automated Order Plan Development
MIG_Url	c.t.a.a.jms.jndi.url	URL for the JMS server events.
MIG_Username	c.t.a.a.jms.jndi.security.principal	Username for the JMS server events.
AFF/OrderManagement/Constants		
	NA	These properties are not required in Automated Order Plan Development
AFF/OrderManagement/Flags		
	NA	These properties are not required in Automated Order Plan Development
AFF/OrderManagement/HTTP		
	NA	These properties are not required in Automated Order Plan Development
AFF/OrderManagement/JMS		
		The value for the properties is picked up from the JMS config mentioned above

Global Variable Name	Configuration Property	Purpose
GLB_PlanDevelopmentAmendRequestQueue	c.t.f.o.af. aopd.amendplanrequest.sender.queue	
GLB_PlanDevelopmentAmendResponseQueue	c.t.f.o.af. aopd.amendplanresponse.receiver.queue	
GLB_PlanDevelopmentNewRequestQueue	c.t.f.o.af. aopd.newplanrequest.sender.queue	
GLB_PlanDevelopmentNewResponseQueue	c.t.f.o.af. aopd.newplanresponse.receiver.queue	
GLB_ProductCataloguePublishTopic	c.t.f.o.af. productmodel.publish.topic	
GLB_ActionCataloguePublishTopic	c.t.f.o.af. actionmodel.publish.topic	
AFF/OrderManagement/ ProductManagement/ Interfaces/JMS/ Services		
		The value for the properties is picked up from the JMS config mentioned above
CommonServicesClientLib/ CommonServices		
NA		These properties are not required in Automated Order Plan Development
CommonServicesClientLib/Events		
NA		These properties are not required in Automated Order Plan Development
CommonServicesClientLib/Timing		
NA		These properties are not required in Automated Order Plan Development

Orchestrator Global Variables

This table is a mapping that shows the global variables mapped to configuration properties.



For readability purpose, to property names are abbreviated as follow:

- c.t.f for com.tibco.fom

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
affv4/common/ constants/ resultStatus			
Error	NA		These Global Variables are not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.
Fatal	NA		These Global Variables are not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.
Information	NA		These Global Variables are not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
Success	NA		These Global Variables are not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.
Warning	NA		These Global Variables are not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.
affv4/common/messaging/ jms			
GLB_MessagingPrefix	NA.	The string to be prefixed to each JMS destination name.	All the JMS destination name properties in Order Management Server contains this prefix in their value itself. Since beginning, there is no separate prefix property. Hence this Global Variable is not carried forward as a property in Orchestrator.
affv4/orchestrator/configuration			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
Split StateMachine Threshold	c.t.f.orch.generator. splitStateMachineThreshold		This Global Variable is used to enable and disable the action of splitting the state machine into smaller state machines if the state machine with the combined count of plan items and order lines is above the set value. It is split so that a fast execution of orders is possible. The default value is 50.
Generator Thread Count	c.t.f.orch.generator.threadCount		This Global Variable indicates the number of threads to execute in parallel for the generation of state machines. This optimization utilizes the modern CPUs to parallelize the state machine XML generation for orders with large plan items.
GLB_CleanUpObjectsDelay	NA	Time delay in msec after which the instances of orders and plans, which are scheduled to cleanup is actually cleaned up (deleted).	This Global Variable is not relevant in Orchestrator and hence not carried forward in 2.1.0. The Order Management Server DB is the only data store now and so, the cleanup of order data from it is handled by the purge utility.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
Auto purge enabled	c.t.f.orch.autoPurgeOnCompletion	Flag to enable or disable auto purge of orders in Orchestrator	NA
GLB_DefaultProcessComponent ErrorHandler	Default Process Component Error Handler c.t.f.orch.pcErrorHandler	The name of the default error handler component to, which Orchestrator sends the PlanItemExecute FailedRequest for failed plan items.	NA
GLB_FeasibilityRetries	Feasibility Retries c.t.f.orch.feasibilityRetries	Retry count for failed Feasibility step.	NA
GLB_FeasibilityRetryInterval	Feasibility Retry Interval c.t.f.orch.feasibilityRetryInterval	Interval in msec to wait before retrying failed Feasibility step.	NA
GLB_OPDRetries	OPD Retries c.t.f.orch.OPDRetries	Retry count for failed OPD step.	NA
GLB_OPDRetryInterval	OPD Retry Interval c.t.f.orch.opdRetryInterval	Interval in msec to wait before retrying failed OPD step.	NA
GLB_PlanItemSLAThreshold Constant	NA	SLA threshold for percentage of maximum duration to publish an SLA notification.	Plan Item SLA notification feature is removed from Orchestrator in 2.1.0 release. This Global Variable is not relevant and hence not carried forward.
GLB_ProcessComponent Retries	Maximum number of retries for failed process component c.t.f.orch.failedPC.maxRetryCount	Default retry count for failed process components.	NA
GLB_ProcessComponent RetryInterval	Time interval between consecutive retry attempts for failed process component c.t.f.orch.failedPC.retryInterval	Default retry delay in msec for failed process components.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_SchedulerPollingInterval	NA	Polling interval in msec for scheduler.	This property is used in BE Orchestrator to create the BE scheduler threads on startup. It is not relevant in the Orchestrator and hence not carried forward.
GLB_BEProfilerOutput File Path	NA	The path of output file to be used by BE profiler.	This property was added specifically to control the BE engine's profiling. It is not relevant in the Orchestrator and hence not carried forward.
GLB_BEProfilerLevel	NA	The level of BE engine profiling	This property was added specifically to control the BE engine's profiling. It is not relevant in the Orchestrator and hence not carried forward.
GLB_BEProfilerDuration InSeconds	NA	The time duration in seconds for which the profiling is to be done.	This property was added specifically to control the BE engine's profiling. It is not relevant in the Orchestrator and hence not carried forward.
Time Dependency Polling Interval in milliseconds	c.t.f.orch.timeDependency.pollingInterval		The default value is 10000.
Time Dependency Buffer Interval in milliseconds	c.t.f.orch.timeDependency.bufferInterval		The default value is 1000.
Enable JMS connection Check	c.t.f.orch.jms.jndiLookup.connection.enableJMSConnectionValidation		The default value is false. Set the value to true to enable the JMS connection check.

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
Threshold for invalidating the Resource Cached Data	c.t.f.orch.resource.connection.invalidateCacheStatusdataThreshold		The default value is 2000.
Executor Processor Threads for DB Ping	c.t.f.orch.resource.connection.executorProcessorThreadCount		The default value is 5.
JMS/DB connection check Monitor Interval in milliseconds	c.t.f.orch.jms.health.intervalMonitoring		The default value is 1500.
Maximum number of milestones corresponding to StateMachine to be kept in Heap Memory	c.t.f.orch.maxNoMilestonesLoadedinMemory		The default value is 0.
Shutdown Threads Count	c.t.f.orch.shutdownThreadsCnt		This property was added to set the number of threads to be created in Orchestrator for a shutdown task. The default value is 10.
Enable DB Condition Evaluator for Statemachines for SimpleStateMachine	c.t.f.orch.generator.enableDBConditionEvaluatorStateMachine		The default value is false. Set the value to true to enable the idle time after model loading.
Enable Milestone Release during activation	c.t.f.orch.enableMilestoneReleaseDuringActivation		The default value is true. Set the value to false to disable the milestone release during activation action.
affv4/orchestrator/ constants/ actions			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
Cancel	NA	Order line cancel action.	This Global Variable is not exposed for modification in Administrator. In Order Management Server, this is handled internally. Hence it is not carried forward as a property in Orchestrator.
affv4/orchestrator/ constants/ dependencyStatus			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, dependency status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ errors			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Orchestrator, the error messages are not exactly same. Hence these Global Variables are not carried forward in Orchestrator.
affv4/orchestrator/ constants/ Generic			
GLB_OriginatorString	NA	NA	This property was introduced and used in Orchestrator to Transient Data Store integration (pre 2.1.0). It is not relevant in the Orchestrator and hence not carried forward.
affv4/orchestrator/ constants/ Milestone			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, milestone names are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
affv4/orchestrator/ constants/ milestoneStatus			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, milestone status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ notificationEventName			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. These were added in BE Orchestrator to identify the notification event type in the common internal event. None of these Global Variables are relevant and hence not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ notifications			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. The messages to be sent in entity status change notification events are handled in Orchestrator internally. None of these Global Variables are relevant and hence not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ orderAmendmentStatus			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, order amendment status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ orderLineStatus			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, order line status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ orderStatus			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, order status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ planItemStatus			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, plan item status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ planStatus			
NA	NA	NA	None of the Global Variables under this category are exposed for modification through Administrator. In Order Management Server, plan status constants are maintained internally. Hence these Global Variables are not carried forward as a properties in Orchestrator.
affv4/orchestrator/ constants/ processComponentID			
NoReciprocalAction	Have to externalize in ConfigValues_OMS.xml	Name of the process component to indicate no reciprocal action is required on cancellation.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
NonExecuting	Non Executing Planfragment ID c.t.f.orch.nonexecuting. planfragmentID	Name of the process component to indicate that no action have to be sent to process components.	NA
affv4/orchestrator/flags			
GLB_CleanupObjectsAt EndOf Order	NA	Flag to enable cleanup of objects at the end of an order lifecycle.	This Global Variable is not relevant in Orchestrator and hence not carried forward in 2.1.0. Since Order Management Server DB is the only data store now, the cleanup of order data from it is handled by the purge utility.
GLB_EnableExternalOr derID Source	NA	Flag to enable orderID generation external to Orchestrator.	This Global Variable is not relevant in Orchestrator and hence not carried forward in 2.1.0. Order Management Server generates the unique orderID for each incoming order and the Orchestrator uses only that.
GLB_EnableFeasibility Error Handling	Enable Feasibility Error Handling c.t.f.orch.enableFeasibility ErrorHandling	Flag to enable PreQualification Failed Handler for feasibility failures.	NA

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
GLB_EnableOMSIntegration	NA	Flag to enable Order Management Server integration.	This Global Variable is not relevant in Orchestrator and hence not carried forward in 2.1.0. The Orchestrator is an integral part of Order Management Server.
GLB_EnableOPDErrorHandling	Enable OPD Error Handling c.t.f.orch.enableOPD ErrorHandling	Flag to enable PreQualification Failed Handler for OPD failures.	NA
GLB_EnableOrderAmendment StatusNotifications	Enable Order Amendment Status Change c.t.f.orch.enableOrder AmendmentStatusChange	Flag to enable order amendment status notifications.	NA
GLB_EnableOrderAmendments	NA	Flag to enable order amendments.	There is no need to have a flag to enable/disable amendments. Amendment functionality is widely used by customers and is OOTB. Since it is not relevant, this Global Variable is not carried forward in Orchestrator.
GLB_EnableOrderLine Status Notifications	Enable OrderLine Status Change c.t.f.orch.enableOrderLine StatusChange	Flag to enable order line status notifications.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_EnableOrderReject Notifications	NA	Flag to enable order reject notifications.	Order reject functionality is not relevant in 2.1.0. Order Management Server web service interface takes care of rejecting the order by sending the appropriate response. Since it is not relevant, this Global Variable is not carried forward in Orchestrator.
GLB_EnableOrderStatus Notifications	Enable Order Status Change c.t.f.orch.enableOrderStatusChange	Flag to enable order status notifications.	NA
GLB_EnablePlanDevelopment Notifications	Enable Plan development notification c.t.f.orch.enablePlanDevelopment Notification	Flag to enable plan development notifications.	NA
GLB_EnablePlanItemFailed Notifications	NA	Flag to enable plan item failed notifications.	This flag was added in BE Orchestrator specifically from Order Management Server perspective. PlanItem NotificationEvent for a failed plan item is published if this flag is enabled. Based on the notification, Order Management Server could show the status as ERROR_HANDLE R. Since Orchestrator is an integral part of Order Management Server in 2.1.0, it is handled internally. This flag is not relevant and hence not carried forward.

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
GLB_EnablePlanItemSLA Notification	NA	Flag to enable SLA notifications for plan items.	Plan Item SLA notification feature is removed from Orchestrator in 2.1.0 release. This Global Variable is not relevant and hence not carried forward.
GLB_EnablePlanItemStatus Notifications	Enable PlanItem Status Change c.t.f.orch.enablePlanItemStatusChange	Flag to enable plan item status notifications.	NA
GLB_EnablePlanStatus Notifications	Enable Plan Status Change c.t.f.orch.enablePlanStatusChange	Flag to enable plan status notifications.	NA
GLB_EnableSubmitOrder Responses	NA	Flag to enable sending response to submit order events.	Orchestrator is an integral part of Order Management Server now. Order Management Server web service is the only entry point interface and there is no need to send any response event. This Global Variable is not relevant and hence not carried forward.
GLB_FeasibilityRequired	Feasibility Required c.t.f.orch.feasibilityRequired	Flag to enable feasibility step.	NA
GLB_RetryFailedFeasibility	Retry Failed Feasibility c.t.f.orch.retryFailedFeasibility	Flag to enable retry of failed Feasibility step.	NA
GLB_RetryFailedOPD	Retry Failed OPD c.t.f.orch.retryFailedOPD	Flag to enable retry of failed OPD step.	NA
GLB_RetryFailedProcess Components	Enable retries for failed process components c.t.f.orch.failedPC.enableRetry	Flag to enable retry of failed process components.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_StoreFailedPlanItem Messages	NA	Flag to enable storing failed plan item messages.	This flag is not relevant and hence not carried forward in Orchestrator.
GLB_StoreFailedProcesses ComponentsMessages	NA	Flag to enable storing failed process components messages.	This flag is not relevant and hence not carried forward in Orchestrator.
GLB_StorePreQualification FailedMessages	NA	Flag to enable storing failed process Pre-Qualification messages.	This flag is not relevant and hence not carried forward in Orchestrator.
GLB_EnableBEProfiling	NA	Flag to enable BE engine's profiling.	This flag is not relevant and hence not carried forward in Orchestrator.
affv4/orchestrator/flags/notification			
GLB_IncludeOrderOn Order AmendmentNotification	NA	Flag to include the order on order amendment notifications.	Have to decide on whether to support this functionality in Orchestrator.
GLB_IncludeOrderOn Order LineNotification	NA	Flag to include the order on order line notifications.	Have to decide on whether to support this functionality in Orchestrator.
GLB_IncludeOrderOn OrderNotification	NA	Flag to include the order on order notifications.	Have to decide on whether to support this functionality in Orchestrator.
GLB_IncludePlanOnOrder AmendmentNotification	NA	Flag to include the plan on order amendment notifications.	Have to decide on whether to support this functionality in Orchestrator.
GLB_IncludePlanOnOrder LineNotification	NA	Flag to include the plan on order line notifications.	Have to decide on whether to support this functionality in Orchestrator.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_IncludePlanOnOrderNotification	NA	Flag to include the plan on order notifications.	Have to decide on whether to support this functionality in Orchestrator.
affv4/orchestrator/ interfaces/jdbc/ backingstore			
NA	NA	NA	None of the Global Variables under this category are relevant for Orchestrator since starting version 2.1.0, Orchestrator is an integral part of Order Management Server now and uses only Order Management Server DB. Hence these Global Variables are not carried forward in Orchestrator.
affv4/orchestrator/ interfaces/jms/events			
NA	NA	NA	None of the Global Variables under this category are relevant for Orchestrator since starting version 2.1.0, Orchestrator is an integral part of Order Management Server now and uses the JMS connection properties used by Order Management Server. Hence these Global Variables are not carried forward in Orchestrator.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
affv4/orchestrator/ messaging/ jms/ destinations			
GLB_DataDeleteOrder Request	NA	Delete order request destination.	DeleteOrderRequestEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataDeleteOrder Response	NA	Delete order response destination.	DeleteOrderResponseEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataDeletePlan Request	NA	Delete plan request destination.	DeletePlanRequestEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataDeletePlan Response	NA	Delete plan response destination.	DeletePlanResponseEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetOrderRequest	NA	Get order request destination.	GetOrderRequestEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetOrderResponse	NA	Get order response destination.	GetOrderResponseEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_DataGetOrderSummaryRequest	NA	Get order summary request destination.	GetOrderSummaryRequest Event is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetOrderSummaryResponse	NA	Get order summary response destination.	GetOrderSummaryResponse Event is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetPlanItemsRequest	NA	Get plan items request destination.	GetPlanItemsRequest Event is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetPlanItemsResponse	NA	Get plan items response destination.	GetPlanItemsResponse Event is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetPlanRequest	NA	Get plan request destination.	GetPlanRequestEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_DataGetPlanResponse	NA	Get plan response destination.	GetPlanResponseEvent is not relevant in Orchestrator and hence this Global Variable is not carried forward.
GLB_ExternalFeasibilityRequest	External Feasibility request queue c.t.f.orch.feasibility.request.queue	External feasibility handler request destination.	NA

Global Variable Name	Configuration Property (name and proptype in ConfigValues_OMS.xml)	Description	Comments
GLB_ExternalFeasibility Response	External Feasibility reply queue c.t.f.orch.feasibility.reply.queue	External feasibility handler response destination.	NA
GLB_ExternalOPDRequest	OPDRequest from Orchestrator receiver queue c.t.f.oms.af.orch.opdrequest.receiver.queue	External plan development handler request destination.	NA
GLB_ExternalOPDResponse	OPDResponse to Orchestrator sender queue c.t.f.oms.af.orch.opdresponse.sender.queue	External plan development handler response destination.	NA
GLB_ExternalPlanItem Failed Request	PlanItem error handler request queue c.t.f.orch.planItem.errhandler.request.queue	External plan item failed handler request destination.	NA
GLB_ExternalPlanItem Failed Response	PlanItem error handler response queue c.t.f.orch.planItem.errhandler.response.queue	External plan item failed handler response destination.	NA
GLB_ExternalPreQualification FailedRequest	External PreQualificationFailed request queue c.t.f.orch.prequalificationfailed.request.queue	External pre-qualification failed handler request destination.	NA
GLB_ExternalPreQualification FailedResponse	External PreQualificationFailed reply queue c.t.f.orch.prequalificationfailed.reply.queue	External pre-qualification failed handler response destination.	NA
GLB_InternalDependencyTime Delta	NA	Dependency time delta trigger destination.	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_InternalFeasibility Request	NA	Feasibility request trigger destination.	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
GLB_InternalOPDRequest	NA	OPD request trigger destination.	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
GLB_InternalPlanItem RetryTimeDelta	NA	Plan item retry time delta trigger destination.	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
GLB_InternalPlanItem SLANotifyTrigger	NA	Plan item SLA trigger destination.	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.

Global Variable Name	Configuration Property (name and proname in ConfigValues_OMS.xml)	Description	Comments
GLB_ModelProcess ComponentModel	NA	Process component model publish destination.	The JMS topic specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. Orchestrator uses the plan fragment from Order Management Server DB.
GLB_NotificationOrder	Order status change destination c.t.f.orch.order.statusChange.de stination	Order change notification publish destination.	NA
GLB_NotificationOrder Amendment	Order Amendment status change destination c.t.f.orch.orderAmendment. statusChange.destination	Order amendment change notification publish destination.	NA
GLB_NotificationOrder Line	OrderLine status change destination c.t.f.orch.orderLine.statusChang e. destination	Order line change notification publish destination.	NA
GLB_NotificationOrder Reject	NA	Order reject notification publish destination.	The JMS topic specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. Orchestrator uses the plan fragment from Order Management Server DB.
GLB_NotificationPlan	Plan status change destination c.t.f.orch.plan.statusChange.dest ination	Plan change notification publish destination.	NA
GLB_NotificationPlan Development	Plan development notification destination c.t.f.orch.planDevelopment. notification.destination	Plan development notification publish destination.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_NotificationPlanItem	PlanItem status change destination c.t.f.orch.planItem.statusChange . destination	Plan item change notification publish destination.	NA
GLB_NotificationPlanItem SLA	NA	Plan item SLA notification publish destination.	Plan Item SLA notification feature is removed from Orchestrator in 2.1.0 release. This Global Variable is not relevant and hence not carried forward.
GLB_OrchestratorStartup Event	NA	Orchestrator startup event request for process component models	The Orchestrator doesn't use the queue specified in this Global Variable anymore. Hence this Global Variable is not carried forward.
GLB_OrchestratorStartup Service	NA	Orchestrator startup service request for process component models	The Orchestrator doesn't use the queue specified in this Global Variable anymore. Hence this Global Variable is not carried forward.
GLB_OrderActivate	Activate order request queue c.t.f.orch.order.activateRequest.queue	Order activate request publish destination.	NA
GLB_OrderCancel	NA	Order cancel request publish destination.	Order cancellation is implemented and achieved using order amendment functionality. The Orchestrator doesn't use the queue specified in this Global Variable anymore. Hence this Global Variable is not carried forward.

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
GLB_OrderSubmit	Have to externalize in ConfigValues_OMS.xml	Order submit request publish destination.	NA
GLB_OrderSuspend	Suspend order request queue c.t.f.orch.order.suspendRequest.queue	Order suspend request publish destination.	NA
GLB_OrderWithdraw	Have to externalize in ConfigValues_OMS.xml	Order withdraw request publish destination.	NA
GLB_PlanItemActivate Request	PlanItem activate request queue c.t.f.orch.planItem.activate.request.queue	Plan item activate request destination.	NA
GLB_PlanItemExecute Request	PlanItem execution request queue c.t.f.orch.planItem.execute.request.queue	Plan item execute request destination.	NA
GLB_PlanItemExecute Response	PlanItem suspend request queue c.t.f.orch.planItem.suspend.request.queue	Plan item execute response destination.	NA
GLB_PlanItemExternal DependencyReleaseRequest	PlanItem External Dependency Release Request queue c.t.f.orch.planItem.externalDependency.release.request.queue	Plan item external dependency release destination.	NA
GLB_PlanItemMilestone NotifyRequest	MilestoneNotifyRequest from process components to Orchestrator queue c.t.f.orch.planItem.milestone.notifyRequest.queue	Plan item milestone notify destination (Process Component to Orchestrator).	NA
GLB_PlanItemMilestone ReleaseRequest	MilestoneReleaseRequest from Orchestrator to process components queue c.t.f.orch.planItem.milestone.releaseRequest.queue	Plan item milestone release destination (Orchestrator to Process Component).	NA
GLB_PlanItemSuspend Request	PlanItem suspend request queue c.t.f.orch.planItem.suspend.request.queue	Plan item suspend request destination.	NA
GLB_PlanItemSuspend Response	PlanItem suspend response queue c.t.f.orch.planItem.suspend.response.queue	Plan item suspend response destination.	NA

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_PlanOMSSet	NA	Set plan to Order Management Server destination.	Since Orchestrator is an integral part of Order Management Server now, it doesn't use the queue specified in this Global Variable. Hence it is not relevant and not carried forward.
GLB_PlanSetPlanItemStatus Request	NA	Plan item set status request destination.	The Orchestrator doesn't use the queue specified in this Global Variable anymore. Hence this Global Variable is not relevant and not carried forward.
GLB_PlanSetPlanItemStatus Response	NA	Plan item set status reply destination.	The Orchestrator doesn't use the queue specified in this Global Variable anymore. Hence this Global Variable is not relevant and not carried forward.
GLB_PlanOMSSet Response	NA	Set plan Order Management Server response destination.	Since Orchestrator is an integral part of Order Management Server now, it doesn't use the queue specified in this Global Variable. Hence it is not relevant and not carried forward.

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_InternalCleanUp Order Request	NA	CleanUpOrderReq uest destination	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
GLB_InternalCleanUpP lan Request	NA	CleanUpPlanReque st destination	The JMS queue specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
commonServices/ configuration			
GLB_LogLevel	NA	Log level to be used.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.
GLB_LogPublishLevel	NA	Log level to be used for publishing the logs.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.
commonServices/flags			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_EnableError Publish	NA	Flag to enable error publishing.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.
GLB_EnableLogPublish	NA	Flag to enable log publishing.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.
common services/ interfaces/jms/ events			
NA	NA	NA	None of the Global Variables under this category are relevant for Orchestrator since Orchestrator is an integral part of Order Management Server now and uses the JMS connection properties used by Order Management Server. Hence these Global Variables are not carried forward in Orchestrator.
common services/ messaging/ jms/			

Global Variable Name	Configuration Property (name and propname in ConfigValues_OMS.xml)	Description	Comments
GLB_MessagingPrefix	NA	String to be prefixed to each JMS destination name.	All the jms destination name properties in Order Management Server contains this prefix in their value itself. Since beginning, there is no separate prefix property. Hence this Global Variable is not carried forward as a property in Orchestrator.
commonServices/ messaging/ jms/ destinations			
GLB_Exception	NA	Exception publish topic.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.
GLB_GetActiveConfigVariable Request	NA	Get active configuration variable request topic.	The JMS topic specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.

Global Variable Name	Configuration Property (name and proppname in ConfigValues_OMS.xml)	Description	Comments
GLB_GetActiveConfigVariable Response	NA	Get active configuration variable response topic.	The JMS topic specified by this Global Variable is not relevant and hence not carried forward in Orchestrator. It was BE implementation specific.
GLB_Instrumentation	NA	Instrumentation publish topic.	Have to decide on whether to support instrumentation functionality in Orchestrator.
GLB_Log	NA	Log publish topic.	The Orchestrator is Java based and uses the same Log4J configurations used by Order Management Server from omsServerLog4j.xml file. Hence this Global Variable is not carried forward.

Global Variables and Configurations

The following section lists the global variables in the Offer and Price Engine component and the configuration properties in Order Management Server.

Order Management System

The following are the configuration properties in Order Management Server:

Name	Description	Default Value
com.tibco.aff.oms.jmx.rmiport	JMX RMI Port.	10099
com.tibco.af.oms.statusnotification.order.queue	Order Status Notification Queue.	tibco.aff.oms.statusNotification.order.queue
com.tibco.af.oms.statusnotification.order.queue.concurrentConsumersCount	Order Status Notification Queue Concurrent Listener Count.	2
com.tibco.af.oms.statusnotification.order.Line.queue	Order Line Status Notification Queue.	tibco.aff.oms.statusNotification.orderLine.queue

Name	Description	Default Value
com.tibco.af.oms.statusnotification.order Line.queue.concurrentConsumersCount	Order Line Status Notification Queue Concurrent Listener Count.	1
com.tibco.af.oms.statusnotification.plan.queue	Plan Status Notification Queue.	tibco.aff.oms.statusNotification.plan.queue
com.tibco.af.oms.statusnotification.plan. queue.concurrentConsumersCount	Plan Status Notification Queue Concurrent Listener Count.	1
com.tibco.af.oms.statusnotification.planItem.queue	Plan Item Status Notification Queue.	tibco.aff.oms.statusNotification.planItem.queue
com.tibco.af.oms.statusnotification.planItem. queue.concurrentConsumersCount	Plan Item Status Notification Queue Concurrent Listener Count.	6
com.tibco.af.oms.statusnotification.order Amendment.queue	Order Amendment Status Notification Queue.	tibco.aff.oms.statusNotification.orderAmendment.queue
com.tibco.af.oms.statusnotification.order Amendment.queue.concurrent ConsumersCount	Order Amendment Status Notification Queue Concurrent Listener Count.	1
com.tibco.af.oms.statusnotification.dead.queue	Status Notification Dead Queue.	tibco.aff.oms.statusNotification.dead.queue
com.tibco.af.oms.centrallog.queue	Central Log Queue.	tibco.aff.centrallog.queue
com.tibco.af.oms.centrallog.queue . concurrentConsumersCount	Central Log Queue Concurrent Listener Count.	1
com.tibco.af.oms.setplan.queue	Set Plan Queue.	tibco.aff.orchestrator.plan.oms.set
com.tibco.af.oms.setplan.queue.co ncurrent ConsumersCount	Set Plan Queue Concurrent Listener Count.	3
com.tibco.af.oms.setplan.dead.queue	Set Plan Dead Queue.	tibco.aff.orchestrator.plan.oms.set.dead
com.tibco.af.oms.setplan.reply.queue	Set Plan Response Queue.	tibco.aff.orchestrator.plan.oms.set.reply
com.tibco.af.oms.setplanfragment model.queue	Set Plan Fragment Model Queue.	tibco.aff.oms.planfragmentmodel

Name	Description	Default Value
com.tibco.af.oms.setplanfragmentmodel.queue.concurrentConsumersCount	Set Plan Fragment Model Queue Concurrent Listener Count.	1
com.tibco.af.oms.setplanfragmentmodel.dead.queue	Set Plan Fragment Model Dead Queue.	tibco.aff.oms.planfragmentmodel.dead
com.tibco.af.oms.setplanitem.queue	Set Plan Item Queue.	tibco.aff.oms.setplanitem
com.tibco.af.oms.setplanitem.queue.concurrentConsumersCount	Set Plan Item Queue Concurrent Listener Count.	1
com.tibco.af.oms.orderService.syncOrderStatusRecovery.concurrentConsumersCount	Synchronous Order Submission Status Recovery Consumer Count.	1
com.tibco.af.oms.orderService.syncOrderStatusRecovery.queue	Synchronous Order Submission Status Recovery Queue.	tibco.aff.oms.syncorderstatusrecovery
com.tibco.af.oms.setplanitem.dead.queue	Set Plan Item Dead Queue.	tibco.aff.oms.setplanitem.dead
com.tibco.af.oms.ordersService.queue	Queue for receiving SOAP Over JMS Order Service requests.	tibco.aff.oms.orderService
com.tibco.af.oms.webservice.soap.jms.concurrentConsumers	Minimum number of concurrent consumers for listener (default 1).	1
com.tibco.af.oms.webservice.soap.jms.maxConcurrentConsumers	Maximum number of concurrent consumers for listener (default 1).	5
com.tibco.af.oms.milestone.notify.request.queue	Milestone Notify Request Queue.	tibco.aff.oms.planItem.milestone.notify.request
com.tibco.af.oms.milestone.notify.request.queue.concurrentConsumersCount	Milestone Notify Request Queue Concurrent Listener Count.	3
com.tibco.af.oms.milestone.notify.request.dead.queue	Milestone Notify Request Dead Queue.	tibco.aff.orchestrator.oms.planItem.milestone.notify.request.dead
com.tibco.af.oms.milestone.release.request.queue	Milestone Release Request Queue.	tibco.aff.oms.planItem.milestone.release.request
com.tibco.af.oms.milestone.release.request.queue.concurrentConsumersCount	Milestone Release Request Queue Concurrent Listener Count.	3

Name	Description	Default Value
com.tibco.af.oms.milestone.release.request.dead.queue	Milestone Release Request Dead Queue.	tibco.aff.orchestrator.oms.planItem.milestone.release.request.dead
com.tibco.af.oms.events.jeopardy.update.queue	Jeopardy Events Update Queue.	tibco.aff.oms.events.jeopardy.update
com.tibco.af.oms.events.jeopardy.update.queue.concurrentConsumersCount	Jeopardy Events Update Queue Concurrent Listener Count.	1
com.tibco.af.oms.events.jeopardy.update.dead.queue	Jeopardy Events Update Dead Queue.	com.tibco.aff.dead
com.tibco.af.dead.queue	AFF Dead Queue.	com.tibco.aff.dead
com.tibco.af.oms.router.destination.beoPurgeOrder.reply.queue.concurrentConsumersCount	Purge Order Reply Queue for Orchestrator Concurrent Listener Count.	1
com.tibco.af.oms.plan.migrated.request	Enrich Migrated Plan Request Queue.	tibco.aff.oms.plan.migrated.request
com.tibco.af.oms.plan.migrated.response	Enrich Migrated Plan Response Queue.	tibco.aff.oms.plan.migrated.response
com.tibco.af.oms.migratedPlanTimeout	Enrich Migrated Plan Timeout.	30000
com.tibco.af.oms.jeoms.update.rule	Update Jeopardy Configuration Rule Queue.	tibco.aff.oms.jeoms.update.rule
com.tibco.af.oms.statusmessage.max.retries	Maximum number of times the Status Message to be retried.	3
com.tibco.af.oms.statusmessage.delivery.delay	Interval delay in millisecs between Status Message retries.	3000
com.tibco.af.oms.statusmessage.max.delivery.delay	Maximum delay in millisecs For Status Message retries.	30000
com.tibco.af.oms.statusmessage.useExponentialBackOff	Use Exponential backoff For Status Message retries.	FALSE
com.tibco.af.oms.statusmessage.exponentialBackOffMultiplier	Exponential backoff Multiplier For Status Message retries.	2
com.tibco.af.oms.statusmessage.logRetryAttempted	Log retry attempt For Status Message retries.	FALSE

Name	Description	Default Value
com.tibco.af.oms.statusmessage.logStackTr ace	Log retry failed stacktrace For Status Message retries.	FALSE
com.tibco.af.oms.router.type	Router Type.	passthroughRouter
com.tibco.af.oms.router.destination.beoWithdrawOrder	Withdraw Order Queue for Orchestrator.	tibco.aff.orchestrator.order.withdra w
com.tibco.af.oms.router.destination.beoSubmitOrderReply	Submit Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submitR esponse
com.tibco.af.oms.syncSubmitOrderRequestTi meout	Request time out for synchronous order submission.	30000
com.tibco.af.oms.router.destination.beoSubmitOrder	Submit Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.beoAm endOrder	Amend Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.beoCa ncelOrder	Cancel Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.beoRe sumeOrder	Resume Order Queue for Orchestrator.	tibco.aff.orchestrator.order.act ivat e
com.tibco.af.oms.router.destination.beoSuspendOrder	Suspend Order Queue for Orchestrator.	tibco.aff.orchestrator.order.suspend
com.tibco.af.oms.router.destination.beoPurgeOrder	Purge Order Request Queue for Orchestrator.	tibco.aff.orchestrator.data.order.de lete.request
com.tibco.af.oms.router.destination.beoPurgeOrder.reply	Purge Order Reply Queue for Orchestrator.	tibco.aff.orchestrator.data.order.de lete.reply
com.tibco.af.oms.router.recoveryFileFolde rPath	Path to folder containing Router Recovery Files.	.././routerRecoveryFileFolder
com.tibco.af.oms.router.type	Router Type.	filteringRouter
com.tibco.af.oms.router.destination.beoSubmitOrderReply	Submit Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submitR esponse
com.tibco.af.oms.syncSubmitOrderRequestTi meout	Request time out for synchronous order submission.	30000
com.tibco.af.oms.router.filterCondition	Xpath filter Condition.	/SubmitOrderRequest/orderRequest/hea der/udf[name='Orchestrator']/value/text()

Name	Description	Default Value
com.tibco.af.oms.router.destination.beoSubmitOrder	Submit Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.ipcSubmitOrder	Submit Order Queue for iProcess Conductor.	tibco.aff.ipc.order.submit
com.tibco.af.oms.router.destination.beoWithdrawOrder	Withdraw Order Queue for Orchestrator.	tibco.aff.orchestrator.order.withdraw
com.tibco.af.oms.router.destination.ipcWithdrawOrder	Withdraw Order Queue for iPC.	tibco.aff.ipc.order.withdraw
com.tibco.af.oms.router.destination.ipcSubmitOrderReply	Submit Order Queue for Orchestrator.	tibco.aff.ipc.order.submitResponse
com.tibco.af.oms.router.destination.beoAmendOrder	Amend Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.ipcAmendOrder	Amend Order Queue for iProcess Conductor.	tibco.aff.ipc.order.amend
com.tibco.af.oms.router.destination.beoCancelOrder	Cancel Order Queue for Orchestrator.	tibco.aff.orchestrator.order.submit
com.tibco.af.oms.router.destination.ipcCancelOrder	Cancel Order Queue for iPC.	tibco.aff.ipc.order.submit
com.tibco.af.oms.router.destination.beoResumeOrder	Resume Order Queue for Orchestrator.	tibco.aff.orchestrator.order.activate
com.tibco.af.oms.router.destination.ipcResumeOrder	Resume Order Queue for iPC.	tibco.aff.ipc.order.activate
com.tibco.af.oms.router.destination.beoSuspendOrder	Suspend Order Queue for Orchestrator.	tibco.aff.orchestrator.order.suspend
com.tibco.af.oms.router.destination.ipcSuspendOrder	Suspend Order Queue for iPC.	tibco.aff.ipc.order.suspend
com.tibco.af.oms.router.recoveryFileFolderPath	Path to folder containing Router Recovery Files.	.././routerRecoveryFileFolder
com.tibco.af.oms.hibernate.dialect		org.hibernate.dialect.Oracle10gDialect
com.tibco.af.oms.hibernate.cache.use_second_level_cache	Hibernate Second Level Cache Usage.	FALSE
com.tibco.af.oms.hibernate.cache.provider_class	Hibernate Cache Provider Class.	org.hibernate.cache.NoCacheProvider

Name	Description	Default Value
com.tibco.af.oms.hibernate.transaction.factory_class	Hibernate Transaction Factory Class.	org.hibernate.transaction.JDBCTransactionFactory
com.tibco.af.oms.hibernate.current_session_context_class	Hibernate Session Context Class.	thread
com.tibco.af.oms.hibernate.jdbc.batch_size	Hibernate JDBC Batch size.	30
com.tibco.af.oms.hibernate.show_sql	Hibernate Show SQL.	FALSE
com.tibco.af.oms.hibernate.default_catalog	Hibernate Default Catalog.	aff_oms
com.tibco.af.oms.hibernate.default_archive_catalog	Hibernate Default Archive Catalog.	aff_oms_archive
com.tibco.af.omsui.httpChannelType		
com.tibco.af.omsui.http.port		8080
com.tibco.af.omsui.https.port		8443
com.tibco.af.omsui.enableRecordCountFetch	Enable the Record count fetch for pagination. This makes the data fetch slower and enable Last Page option in pagination.	TRUE
com.tibco.af.omsui.session-fixation-protection		
com.tibco.af.omsServer.proxyHost	Host address of the Order Management Server.	localhost
com.tibco.af.fpServer.enableConfiguration	Enable FP configuration.	TRUE
com.tibco.af.fpServer.nodeName	Node name of the FP server.	knode
com.tibco.af.fpServer.proxyHost	Host address of the FP server.	localhost
com.tibco.af.fpServer.fpPlanFragmentType	Owner for FP.	FP
com.tibco.af.omsServer.proxyPort	Port number of the Order Management Server.	8080
com.tibco.af.fpServer.proxyPort	Port number of the FP Server.	8080

Name	Description	Default Value
com.tibco.af.concurrencyControl. maxSessionPerUser		1
com.tibco.af.omsui.shortDateFormat	Order Management Server UI Short Date Format (Day [dd].	Month[MM] and Year [yyyy])
com.tibco.af.omsui.longDateFormat	Order Management Server UI Long Date Time Format (Day [dd].	Month [MM]
com.tibco.af.omsui. longDateFormatTimeZone	Order Management Server UI Long Date Time Format (Day [dd].	Month [MM]
com.tibco.af.omsui. longDateFormatTimeZoneMillis	Order Management Server UI Long Date Time Format (Day [dd].	Month [MM]
com.tibco.af.omsui.planItemExpression.gridView	Order Management Server UI PlanItem Display Template for Grid View. Possible template variables are - PlanItemID, PlanFragmentID, ProductID, Action and Description; these template variables are case sensitive and have to be specified within curly braces {}. Different template variables can be combined with any type of delimiter Example: {PlanFragmentID} {PlanItemID}.	{PlanItemID}
com.tibco.af.omsui.planItemExpression.ganttView	Order Management Server UI PlanItem Display Template for Gantt View. Possible template variables are - PlanItemID, PlanFragmentID,ProductID, Action and Description; these template variables are case sensitive and have to be specified within curly braces {}. Different template variables can be combined with any type of delimiter Example: {PlanFragmentID} {PlanItemID}.	{Description}

Name	Description	Default Value
com.tibco.af.omsui.planItemExpression.dependencyView	Order Management Server UI PlanItem Display Template for Dependency View. Possible template variables are - PlanItemID, PlanFragmentID, ProductID, Action and Description; these template variables are case sensitive and have to be specified within curly braces {}. Different template variables can be combined with any type of delimiter Example: {PlanFragmentID} {PlanItemID}" name="PlanItem Template for Dependency View.	{Description} {PlanItemID}
com.tibco.af.omsui.pagination.enable	Enable Grid View and Gantt View Pagination	FALSE
com.tibco.af.omsui.gridView.pageSize	Order Management Server UI Grid View Page Size	20
com.tibco.af.omsui.ganttView.pageSize	Order Management Server UI Gantt View Page Size	10
com.tibco.af.purge.threads.count	Purge Thread Count.	4
com.tibco.aff.PublishInventoryNotifications	PublishInventoryNotifications.	FALSE
value com.tibco.fom.oms.af.productmodel.receiver.queue	Product model receiver queue.	tibco.aff.catalog.product.request
com.tibco.fom.oms.af.productmodel.receiver.count	Product model receiver count.	3
com.tibco.fom.oms.af.productmodel.publish.topic	Product model publish topic.	tibco.aff.ocv.events.products.publish
com.tibco.fom.oms.af.customermodel.receiver.queue	Customer model receiver queue.	tibco.aff.catalog.customer.request
com.tibco.fom.oms.af.customermodel.receiver.count	Customer model receiver count.	3
com.tibco.fom.oms.af.segmentmodel.receiver.queue	Segment model receiver queue.	tibco.aff.catalog.segment.request
com.tibco.fom.oms.af.segmentmodel.receiver.count	Segment model receiver count.	3

Name	Description	Default Value
com.tibco.fom.oms.af.planfragmentmodel.receiver.queue	PlanFragment model receiver queue.	tibco.aff.catalog.planfragment.request
com.tibco.fom.oms.af.planfragmentmodel.receiver.count	PlanFragment model receiver count.	3
com.tibco.fom.oms.af.planfragmentmodel.publish.topic	ProcessComponent model publish topic.	tibco.aff.orchestrator.model.processComponent.publish
com.tibco.fom.oms.af.actionmodel.receiver.queue	Action model receiver queue.	tibco.aff.catalog.action.request
com.tibco.fom.oms.af.actionmodel.receiver.count	Action model receiver count.	3
com.tibco.fom.oms.af.actionmodel.publish.topic	Action model publish topic.	tibco.aff.ocv.events.actions.publish
com.tibco.fom.oms.af.pcsmodel.event.request.queue	Product customer segment model invocation event request receiver queue.	tibco.aff.catalog.events.request
com.tibco.fom.oms.af.pcsmodel.event.receiver.count	Product customer segment model invocation event request receiver count.	1
com.tibco.fom.oms.af.pfmodel.event.request.queue	Plan fragment model invocation event request receiver queue.	tibco.aff.orchestrator.startup.event.request
com.tibco.fom.oms.af.pfmodel.event.request.receiver.count	Plan fragment model invocation event request receiver count.	1
com.tibco.fom.oms.af.fcintegration.host	Server host.	localhost
com.tibco.fom.oms.af.fcintegration.port	Server port.	8800
com.tibco.fom.oms.af.fcintegration.enterprise	Enterprise name.	AC
com.tibco.fom.oms.af.fcintegration.username	Username.	admin
com.tibco.fom.oms.af.fcintegration.password	Password.	admin
com.tibco.fom.oms.af.fcintegration.workflow.response.success.code	Workflow invocation response success code.	SVC-11045

Name	Description	Default Value
com.tibco.fom.oms.af.fcintegration.workflow.response.success.message	Workflow invocation response success message.	Workflow initiated successfully.
com.tibco.fom.oms.af.fcintegration.product.mctname	Master catalog name.	PRODUCT
com.tibco.fom.oms.af.fcintegration.product.id	Product record ID.	BU_000001
com.tibco.fom.oms.af.fcintegration.product.idext	Product record ID extension.	
com.tibco.fom.oms.af.fcintegration.customer.mctname	Master catalog name.	PARTY
com.tibco.fom.oms.af.fcintegration.customer.id	Customer record ID.	68000001
com.tibco.fom.oms.af.fcintegration.customer.idext	Customer record ID extension.	
com.tibco.fom.oms.af.fcintegration.segment.mctname	Master catalog name.	SEGMENT
com.tibco.fom.oms.af.fcintegration.segment.id	Segment record ID.	PB_000001
com.tibco.fom.oms.af.fcintegration.segment.idext	Segment record ID extension.	
com.tibco.fom.oms.af.fcintegration.planfragment.mctname	Master catalog name.	PLANFRAGMENT
com.tibco.fom.oms.af.fcintegration.planfragment.id	PlanFragment record ID.	PF_00001
com.tibco.fom.oms.af.fcintegration.planfragment.idext	PlanFragment record ID extension.	
com.tibco.fom.oms.af.fcintegration.action.mctname	Master catalog name.	ACTION
com.tibco.fom.oms.af.fcintegration.action.id	Action record ID.	Provide
com.tibco.fom.oms.af.fcintegration.action.idext	Action record ID extension.	
com.tibco.fom.oms.af.offline.common.polling.interval	Polling interval in seconds.	60

Name	Description	Default Value
com.tibco.fom.oms.af.offline.com mon.catalogrequest.interval	Catalog request interval in milliseconds.	30
com.tibco.fom.oms.af.offline.prod uct.use	Use offline product.	FALSE
com.tibco.fom.oms.af.offline.prod uct.master.directory	Offline product catalog master directory.	/usr/tibco/product/master
com.tibco.fom.oms.af.offline.prod uct.master.importsuccess.directory	Offline product catalog import success master directory.	/usr/tibco/product-success/ master
com.tibco.fom.oms.af.offline.prod uct.master.importfailure.directory	Offline product catalog import failure master directory.	/usr/tibco/product-failure/ master
com.tibco.fom.oms.af.offline.prod uct.directory	Offline product catalog directory.	/usr/tibco/product
com.tibco.fom.oms.af.offline.prod uct.importsuccess.directory	Offline product catalog import success directory.	/usr/tibco/product-success
com.tibco.fom.oms.af.offline.prod uct.importfailure.directory	Offline product catalog import failure directory.	/usr/tibco/product-failure
com.tibco.fom.oms.af.offline.cust omer.use	Use offline customer.	FALSE
com.tibco.fom.oms.af.offline.cust omer.master.directory	Offline customer catalog master directory.	/usr/tibco/customer/master
com.tibco.fom.oms.af.offline.cust omer.master.importsuccess.directory	Offline customer catalog import success master directory.	/usr/tibco/customer-success/ master
com.tibco.fom.oms.af.offline.cust omer.master.importfailure.directory	Offline customer catalog import failure master directory.	/usr/tibco/customer-failure/ master
com.tibco.fom.oms.af.offline.cust omer.directory	Offline customer catalog directory.	/usr/tibco/customer
com.tibco.fom.oms.af.offline.cust omer.importsuccess.directory	Offline customer catalog import success directory.	/usr/tibco/customer-success
com.tibco.fom.oms.af.offline.cust omer.importfailure.directory	Offline customer catalog import failure directory.	/usr/tibco/customer-failure
com.tibco.fom.oms.af.offline.seg ment.use	Use offline segment.	FALSE

Name	Description	Default Value
com.tibco.fom.oms.afs.offline.segment.master.directory	Offline segment catalog master directory.	/usr/tibco/segment/master
com.tibco.fom.oms.afs.offline.segment.master.importsuccess.directory	Offline segment catalog import success master directory.	/usr/tibco/segment-success/master
com.tibco.fom.oms.afs.offline.segment.master.importfailure.directory	Offline segment catalog import failure master directory.	/usr/tibco/segment-failure/master
com.tibco.fom.oms.afs.offline.segment.directory	Offline segment catalog directory	/usr/tibco/segment
com.tibco.fom.oms.afs.offline.segment.importsuccess.directory	Offline segment catalog import success directory.	/usr/tibco/segment-success
com.tibco.fom.oms.afs.offline.segment.importfailure.directory	Offline segment catalog import failure directory.	/usr/tibco/segment-failure
com.tibco.fom.oms.afs.offline.planfragment.use	Use offline planfragment.	FALSE
com.tibco.fom.oms.afs.offline.planfragment.master.directory	Offline planfragment catalog master directory.	/usr/tibco/planfragment/master
com.tibco.fom.oms.afs.offline.planfragment.master.importsuccess.directory	Offline planfragment catalog import success master directory.	/usr/tibco/planfragment-success/master
com.tibco.fom.oms.afs.offline.planfragment.master.importfailure.directory	Offline planfragment catalog import failure master directory.	/usr/tibco/planfragment-failure/master
com.tibco.fom.oms.afs.offline.planfragment.directory	Offline planfragment catalog directory.	/usr/tibco/planfragment
com.tibco.fom.oms.afs.offline.planfragment.importsuccess.directory	Offline planfragment catalog import success directory.	/usr/tibco/planfragment-success
com.tibco.fom.oms.afs.offline.planfragment.importfailure.directory	Offline planfragment catalog import failure directory.	/usr/tibco/planfragment-failure
com.tibco.fom.oms.afs.offline.action.use	Use offline action.	FALSE
com.tibco.fom.oms.afs.offline.action.master.directory	Offline action catalog master directory.	/usr/tibco/action/master
com.tibco.fom.oms.afs.offline.action.master.importsuccess.directory	Offline action catalog import success master directory.	/usr/tibco/action-success/master

Name	Description	Default Value
com.tibco.fom.oms.af.offline.action.master.importfailure.directory	Offline action catalog import failure master directory.	/usr/tibco/action-failure/master
com.tibco.fom.oms.af.offline.action.directory	Offline action catalog directory.	/usr/tibco/action
com.tibco.fom.oms.af.offline.action.importsuccess.directory	Offline action catalog import success directory.	/usr/tibco/action-success
com.tibco.fom.oms.af.offline.action.importfailure.directory	Offline action catalog import failure directory.	/usr/tibco/action-failure
com.tibco.fom.oms.af.orch.opdrequest.receiver.queue	OPDRequest from Orchestrator receiver queue.	tibco.aff.orchestrator.provider.order.opd.request
com.tibco.fom.oms.af.orch.opdrequest.receiver.count	OPDRequest from Orchestrator receiver count.	3
com.tibco.fom.oms.af.aopd.newplanrequest.sender.queue	ExecutionPlanNewRequest to Automated Order Plan Development sender queue.	tibco.aff.ocv.events.plan.new.request
com.tibco.fom.oms.af.aopd.amendplanrequest.sender.queue	ExecutionPlanAmendRequest to Automated Order Plan Development sender queue.	tibco.aff.ocv.events.plan.amend.request
com.tibco.fom.oms.af.aopd.newplanresponse.receiver.queue	ExecutionPlanNewResponse from Automated Order Plan Development receiver queue.	tibco.aff.ocv.events.newplan.reply
com.tibco.fom.oms.af.aopd.newplanresponse.receiver.count	ExecutionPlanNewResponse from Automated Order Plan Development receiver count.	3
com.tibco.fom. af.aopd.amendplanresponse.receiver.queue	ExecutionPlanAmendResponse from Automated Order Plan Development receiver queue.	tibco.aff.ocv.events.amendplan.reply
com.tibco.fom.oms.af.aopd.amendplanresponse.receiver.count	ExecutionPlanAmendResponse from Automated Order Plan Development receiver count.	3
com.tibco.fom.oms.af.orch.opdresponse.sender.queue	OPDResponse to Orchestrator sender queue.	tibco.aff.orchestrator.provider.order.opd.reply
com.tibco.fom.oms.af.aopd.merge.inventory	Merge inventory in Automated Order Plan Development request.	FALSE
com.tibco.fom.oms.tds.getorderrequest.receiver.queue	GetOrderRequest receiver queue.	tibco.aff.tds.order.read.request

Name	Description	Default Value
com.tibco.fom.oms.tds.getorderrequest.receiver.count	GetOrderRequest receiver count.	3
com.tibco.fom.oms.tds.getorderrequest.receiver.deadqueue	GetOrderRequest receiver dead queue.	tibco.aff.oms.tds.order.read.request.dead
com.tibco.fom.oms.tds.getorderresponse.sender.queue	GetOrderResponse sender queue.	tibco.aff.tds.order.reply
com.tibco.fom.oms.tds.getplanrequest.receiver.queue	GetPlan/GetPlanItem Request receiver queue.	tibco.aff.tds.plan.read.request
com.tibco.fom.oms.tds.getplanrequest.receiver.count	GetPlan/GetPlanItem Request receiver count.	3
com.tibco.fom.oms.tds.getplanrequest.receiver.deadqueue	GetPlan/GetPlanItem Request receiver dead queue.	tibco.aff.oms.tds.plan.read.request.dead
com.tibco.fom.oms.tds.getplanresponse.sender.queue	GetPlan/GetPlanItem Response sender queue.	tibco.aff.tds.plan.reply
com.tibco.fom.oms.tds.setplanrequest.receiver.queue	SetPlan/SetPlanItem Request receiver queue.	tibco.aff.tds.plan.request
com.tibco.fom.oms.tds.setplanrequest.receiver.count	SetPlan/SetPlanItem Request receiver count.	3
com.tibco.fom.oms.tds.setplanrequest.receiver.deadqueue	SetPlan/SetPlanItem Request receiver dead queue.	tibco.aff.oms.tds.plan.request.dead
com.tibco.fom.oms.tds.setplanresponse.sender.queue	SetPlan/SetPlanItem Response sender queue.	tibco.aff.tds.plan.reply
com.tibco.fom.oms.tds.enable.unique.udfname	Flag to enable unique user-defined field names to permit updates on user-defined field values.	TRUE
com.tibco.afs.destination.listen.timeout	Wait Timeout for listening to response from JMS Destination.	60000
tibco.aff.ocv.events.offer.validate.request	Queue where request for validating the offer is posted.	tibco.aff.ocv.events.offer.validate.request
com.tibco.af.oms.jms.jndi.ConnectionFactory	JNDI Connection factory JNDI Name.	GenericConnectionFactory
com.tibco.af.oms.jms.jndi.initialContextFactory	JNDI Initial Context Factory.	com.tibco.tibjms.naming.TibjmsInitialContextFactory
com.tibco.af.oms.jms.jndi.url	JNDI URL for JMS Service.	tibjmsnaming://localhost:7222

Name	Description	Default Value
com.tibco.af.oms.jms.jndi.security.principal	JNDI Username.	admin
com.tibco.af.oms.jms.jndi.security.credentials	JNDI Password.	admin
com.tibco.af.oms.jms.jndi.cf.beo	BE Orchestrator Queue Connection factory JNDI Name.	QueueConnectionFactory
com.tibco.af.oms.jms.cf.beo.deliverymode	BE Orchestrator JMS Delivery Mode.	1
com.tibco.af.oms.jms.cf.beo.qosEnabled	BE Orchestrator JMS QoS Enabled?	FALSE
com.tibco.af.oms.jms.jndi.cf.ipc	iProcess Conductor Queue Connection factory JNDI Name.	GenericConnectionFactory
com.tibco.af.oms.jms.cf.ipc.deliverymode	iProcess Conductor JMS Delivery Mode.	1
com.tibco.af.oms.jms.cf.ipc.qosEnabled	iProcess Conductor JMS QoS Enabled.	FALSE
com.tibco.af.oms.jms.jndi.cf.ope	Offer and Price Engine Queue Connection factory JNDI Name	QueueConnection Factory
com.tibco.af.oms.jms.cf.ope.deliverymode	Offer and Price Engine JMS Delivery Mode.	1
com.tibco.af.oms.jms.cf.OPE.qosEnabled	Offer and Price Engine JMS QoS Enabled.	TRUE
com.tibco.af.oms.pooledDataSource.driverClassName	Pooled Data Source Driver Class Name.	oracle.jdbc.driver.OracleDriver
com.tibco.af.oms.pooledDataSource.host	Pooled Data Source Host.	localhost
com.tibco.af.oms.pooledDataSource.port	Pooled Data Source Port.	1521
com.tibco.af.oms.pooledDataSource.database	Pooled Data Source Database.	orcl
com.tibco.af.oms.pooledDataSource.username	Pooled Data Source Username.	aff_oms
com.tibco.af.oms.pooledDataSource.password	Pooled Data Source Password.	aff_oms

Name	Description	Default Value
com.tibco.af.oms.pooledDataSource.url	Pooled Data Source URL.	jdbc:oracle:thin:@//\${com.tibco.af.oms.pooledDataSource.host}:\${com.tibco.af.oms.pooledDataSource.port}/\${com.tibco.af.oms.pooledDataSource.database}
com.tibco.af.oms.pooledDataSource.initializeSize	Pooled Data Source Initialize Size.	2
com.tibco.af.oms.pooledDataSource.maxIdle	Pooled Data Source Max Idle.	8
com.tibco.af.oms.pooledDataSource.maxActive	Pooled Data Source Max Active.	12
com.tibco.af.oms.pooledDataSource.maxWait	Pooled Data Source Max Wait.	10000
com.tibco.af.oms.pooledDataSource.validationQuery	Pooled Data Source Validation Query.	select 1 from dual
com.tibco.af.oms.pooledDataSource.testOnBorrow	Pooled Data Source Test OnBorrow.	FALSE
com.tibco.af.oms.pooledDataSource.testWhileIdle	Pooled Data Source Test WhileIdle.	TRUE
com.tibco.af.oms.pooledDataSource.timeBetweenEvictionRunsMillis	Pooled Data Source Eviction Interval.	1200000
com.tibco.af.oms.pooledDataSource.minEvictableIdleTimeMillis	Pooled Data Source Minimum Evictable Idle Time.	1800000
com.tibco.af.oms.pooledDataSource.numTestsPerEvictionRun	Pooled Data Source Tests Per Eviction Run.	5
com.tibco.af. .pooledDataSource.defaultAutoCommit	Pooled Data Source Default AutoCommit.	FALSE
com.tibco.af.oms.pooledArchiveDataSource.driverClassName	Pooled Data Source Driver Class Name.	oracle.jdbc.driver.OracleDriver
com.tibco.af.oms.pooledArchiveDataSource.host	Pooled Data Source Host.	localhost

Name	Description	Default Value
com.tibco.af.oms.pooledArchiveDataSource.port	Pooled Data Source Port.	1521
com.tibco.af.oms.pooledArchiveDataSource.database	Pooled Data Source Database.	orcl
com.tibco.af.oms.pooledArchiveDataSource.username	Pooled Data Source Username.	aff_oms_archive
com.tibco.af.oms.pooledArchiveDataSource.password	Pooled Data Source Password.	aff_oms_archive
com.tibco.af.oms.pooledArchiveDataSource.url	Pooled Data Source URL.	jdbc:oracle:thin:@/\${com.tibco.af.oms.pooledArchiveDataSource.host}: \${com.tibco.af.oms.pooledArchiveDataSource.port}/ \${com.tibco.af.oms.pooledArchiveDataSource.database}
com.tibco.af.oms.pooledArchiveDataSource.initializeSize	Pooled Data Source Initialize Size.	2
com.tibco.af.oms.pooledArchiveDataSource.maxIdle	Pooled Data Source Max Idle.	8
com.tibco.af.oms.pooledArchiveDataSource.maxActive	Pooled Data Source Max Active.	12
com.tibco.af.oms.pooledArchiveDataSource.maxWait	Pooled Data Source Max Wait.	10000
com.tibco.af.oms.pooledArchiveDataSource.validationQuery	Pooled Data Source Validation Query.	select 1 from dual
com.tibco.af.oms.pooledArchiveDataSource.testOnBorrow	Pooled Data Source Test OnBorrow.	FALSE
com.tibco.af.oms.pooledArchiveDataSource.testWhileIdle	Pooled Data Source Test WhileIdle.	TRUE
com.tibco.af.oms.pooledArchiveDataSource.timeBetweenEvictionRunsMillis	Pooled Data Source Eviction Interval.	1200000

Name	Description	Default Value
com.tibco.af.oms.pooledArchiveDataSource. minEvictableIdleTimeMillis	Pooled Data Source Minimum Evictable Idle Time.	1800000
com.tibco.af.oms.pooledArchiveDataSource. numTestsPerEvictionRun	Pooled Data Source Tests Per Eviction Run.	5
com.tibco.af.oms.pooledArchiveDataSource. defaultAutoCommit	Pooled Data Source Default AutoCommit.	FALSE
com.tibco.af.oms.security.authProvider	Default Authentication Provider.	defaultAuthenticationProvider
com.tibco.af.oms.security.authProvider	Ldap Authentication Provider.	ldapAuthenticationProvider
com.tibco.af.oms.security.authProvider.ldap.server.url	Lightweight Directory Access Protocol Server URL.	ldap://localhost:389/dc=oms
com.tibco.af.oms.security.authProvider.ldap.server.userDn	Lightweight Directory Access Protocol User Manager ID.	uid=admin
com.tibco.af.oms.security.authProvider.ldap.server.password	Lightweight Directory Access Protocol User Manager Password.	password
com.tibco.af.oms.security.authProvider.ldap.user.searchBase	User SearchBase.	ou=users
com.tibco.af.oms.security.authProvider.ldap.user.userSearchFilter	User Search Filter.	(uid={0})
com.tibco.af.oms.security.authProvider.ldap.role.groupSearchBase	Group Search Base.	ou=Groups
com.tibco.af.oms.security.authProvider.ldap.role.groupRoleAttribute	Group Role Attribute.	cn
com.tibco.af.oms.security.authProvider.ldap.role.groupSearchFilter	Group Search Filter.	uniqueMember={0}
com.tibco.af.oms.security.authProvider.ldap.role.searchSubTree	Search Sub Tree.	FALSE
com.tibco.af.oms.security.authProvider.ldap.role.convertToUpperCase	convert values To UpperCase.	TRUE
com.tibco.af.oms.OCVEEnabled	Enable Offer and Price Engine	FALSE
com.tibco.af.oms.OCVTimeOut	Offer and Price EngineTimeout.	60000

Name	Description	Default Value
com.tibco.af.oms.ocv.request	Offer and Price Engine Request Queue.	tibco.aff.ocv.events.offer.validate.request
com.tibco.af.oms.ocv.response	Offer and Price Engine Response Queue. Specify a different queue name for Order Management Server than other Offer and Price Engine clients.	tibco.aff.ocv.events.offer.validate.reply.oms
com.tibco.af.oms.OCVExceptionLoggingEnabled	Enable or disable logging of exception stack trace in Order Management Server when Offer and Price Engine fails.	TRUE
com.tibco.af.oms.webservice.security.userNameTokenBased	Enable User Name token based Security.	TRUE
com.tibco.af.oms.webservice.schema.validation	Enable Schema validation.	TRUE
com.tibco.af.oms.webservice.idempotency	Enable Submit Order Web Service Idempotency.	TRUE
com.tibco.af.oms.webservice.httpChannelType	HTTP Channel type.	
com.tibco.af.oms.http.port		8080
com.tibco.af.oms.https.port		8443
com.tibco.af.oms.webservice.useExternalBusinessTransactionId	Use external business transactionId as business transaction id within Order Management Server.	FALSE
com.tibco.af.oms.updateToken.timeout	Time out value in millisecond for order lock.	150000
com.tibco.af.oms.nonce.validityWindow	Nonce validity time in Milliseconds.	1000
com.tibco.af.oms.persistenceDelay	Delay to be introduced before persisting order objects.	0
com.tibco.af.oms.orderValidationDelay	Delay to be introduced before validating orders.	0
com.tibco.af.oms.summaryDataCollection.scheduled.cronExpress	schedule interval for order summary data collection.	0 0/10 * * * ?

Name	Description	Default Value
com.tibco.af.oms.submitorder.isDuplicateOrderRequestValidationEnabled	Enable Concurrent Order Request Detection	FALSE
com.tibco.fom.oms.modelLoadingMaxIdle	Enable idle time after Model loading	FALSE
com.tibco.af.oms.model.ems.post.disabled	Post Models on Enterprise Message Service for Automated Order Plan Development and Offer and Price Engine Disabled	TRUE
com.tibco.af.oms.model.cache.enabled	Model Cache Persistence	TRUE
com.tibco.af.oms.model.loading.member	Designated model loading member instance	member1
com.tibco.fom.oms.tds.enable.udfUpdate.DuringAmendment	Flag to enable update of user-defined field during order amendment.	TRUE
com.tibco.af.oms.lock.retryInterval	Lock Retry Interval	5000
com.tibco.af.oms.lock.retryCount	Lock Retry Count	5
com.tibco.af.oms.databaseType	Database Type. Allowed values are oracle or postgres	TRUE
com.tibco.fom.oms.modelLoadingMaxIdle	Enable idle time after Model loading	FALSE
com.tibco.fom.oms.afi.purgemodel.internal.queue	Purge model internal queue	tibco.aff.catalog.purgemodel.internal
com.tibco.af.oms.dataSource.pooledDataSource.customProperty	Pooled Database Custom Property	
com.tibco.fom.oms.tds.getkeymappingrequest.receiver.queue	GetKeyMappingRequest receiver queue	tibco.aff.tds.keymapping.request
com.tibco.fom.oms.tds.getkeymappingrequest.receiver.count	GetKeyMappingRequest receiver count	3
com.tibco.fom.oms.tds.getkeymappingrequest.receiver.deadqueue	GetKeyMappingRequest receiver dead queue	tibco.aff.oms.tds.keymapping.read.request.dead
com.tibco.fom.oms.tds.getkeymappingresponse.sender.queue	GetKeyMappingResponse sender queue	tibco.aff.tds.keymapping.reply

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.