

TIBCO® Order Management - Long Running

Concepts and Architecture

Version 5.0.1

April 2022



Contents

Figures	4
About this Product	6
Introduction	7
About TIBCO Order Management - Long Running	7
TIBCO Fulfillment Orchestration Suite Overview	7
TIBCO Fulfillment Orchestration Suite Components	8
User Interface Integration	9
TIBCO Order Management - Long Running Overview	12
Basic Order Management Concepts	14
Order	14
Characteristics	15
Product	16
Plan	16
Plan Item	16
Milestone	17
Dependency	17
Plan Fragment	19
Error Handling	19
SLA Notification	20
Plan Development	21
Lifecycle	22
Order	22
Order Line	24
Order Amendment	25
Plan	26
Plan Item	27
Milestone	28
Dependency	28
Sequences	28
Standard Order	28
Successful Completion	30
Feasibility Failed	31
Plan Development Failed	33
Plan Item Execution Failed	34
Processing New Order with Manual Order Plan Development Enabled	36
Amend Order Fulfillment	39

- Before Plan Creation39
- Amend Order Fulfillment 41
- Processing Amended Order with Manual Order Plan Development Enabled 43
- Cancel Order45
- Architecture46**
 - Order Management Server47
 - Orchestrator 48
 - Automated Order Plan Development51
 - Manual Order Plan Development 54
 - Order Capture System 55
 - Process Components 58
 - Feasibility Provider59
 - Jeopardy Management59
 - Router 62
 - Key Functionality62
- TIBCO Documentation and Support Services 64**
- Legal and Third-Party Notices 65**

Figures

Fulfillment Orchestration Architecture	8
Provisioning in the Orchestration Suite	9
Component Integration	10
UI Integration	11
TIBCO Order Management - Long Running-TIBCO Fulfillment Provisioning- TIBCO Product and Service Catalog Integration	12
Object Model	14
Order Logical Components	15
Plan Item Milestones	17
Point Dependencies	18
Plan Fragment and Process Component Logical Components	19
Plan Fragment Sections	20
Plan Fragment Logical Components	21
Order and Plan Logical Components	22
Order Lifecycle	23
Order Item Lifecycle	24
Order Amendment Lifecycle	25
Plan Lifecycle	26
Plan Item Lifecycle	27
Milestone Lifecycle	28
Dependency Lifecycle	28
Standard Order Fulfillment – Successful Completion Sequence	29
Standard Order Fulfillment – Successful Completion Sequence	30
Standard Order Fulfillment – Feasibility Failed Sequence	32
Standard Order Fulfillment – Plan Development Failed Sequence	33
Standard Order Fulfillment – Plan Item Execution Failed Sequence	35
Sequence Diagram for Processing New Order with Manual Order Plan Development Enabled	36
Amend Order Fulfillment – Before Plan Creation Sequence	40
Amend Order Fulfillment – After Plan Creation Sequence	42
Sequence Diagram for Processing Amended Order with Manual Order Plan Development Enabled	43
TIBCO Order Management - Long Running Architecture	47
Order Management Server	47
Orchestrator	49
Order and Execution Plan	50
Order, Plan, Plan Fragment and Process Component	51
Automated Order Plan Development	52
Plan Generation by Automated Order Plan Development Inputs and Outputs	53
Plan Generation and Execution Sequence	54

Order Capture System Architecture	56
Architecture of TIBCO Order Management - Long Running	60
Plan and Order Execution	60
Working of in-memory cache	61

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 the processing of long-running orders

Introduction

This chapter gives an overview of TIBCO® Fulfillment Orchestration Suite and its internal component TIBCO Order Management - Long Running and its infrastructure software. This includes information regarding all of the TIBCO Fulfillment Orchestration Suite components, user interface integration for the suite components, and TIBCO Order Management - Long Running components.

About TIBCO Order Management - Long Running

TIBCO Order Management - Long Running is a metadata driven order management and fulfillment system. TIBCO Order Management - Long Running is a component of the TIBCO Fulfillment Orchestration Suite.

TIBCO Fulfillment Orchestration Suite Overview

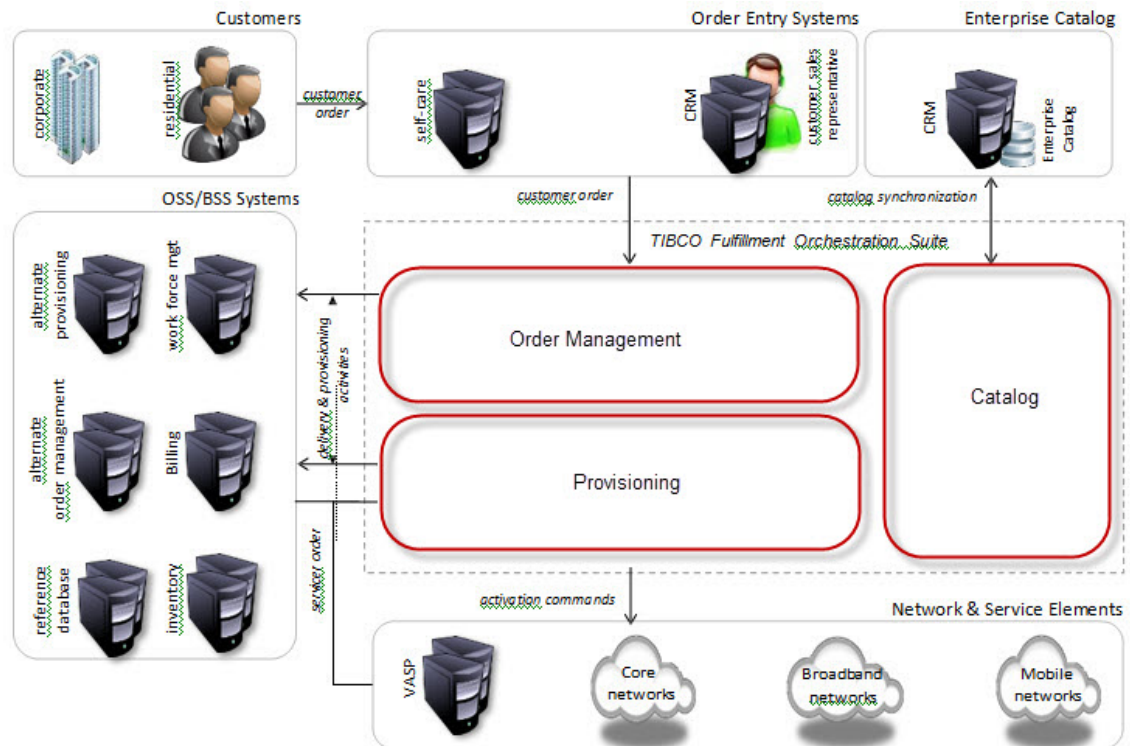
New technologies and network architectures enable communications service providers (CSP) to create innovative converged product and service offerings, which are introduced have faster and shorter life cycles than previous service offerings to address a very changing and competitive market.

In view of the rapid pace of change in the technology, the industry is evolving to become a contributor and not remain a mere consumer of technology. In this environment, communications service providers face the challenge of defining, managing, and delivering numerous complex products and variations to the market in the most effective way to differentiate themselves. TIBCO has concentrated and structured its services around the following points:

- New product offerings are designed and rolled out in a few weeks including implementation in the entire fulfillment chain.
- Customer orders are instantly fulfilled and provisioned in the network to maximize customer experience.
- Customer orders come from a large variety of order entries such as customer self-care portals, customer sales representative desks or even network elements detecting service access to leverage fulfillment chain investment and support hardware rationalization.

TIBCO provides communications service providers with a comprehensive and integrated solution ready for complete end-to-end fulfillment automation. The TIBCO Fulfillment Orchestration Suite defines new product and service offerings, associated fulfillment rules and processes, and automates the delivery from order capture down to the service activation in the network.

Fulfillment Orchestration Architecture



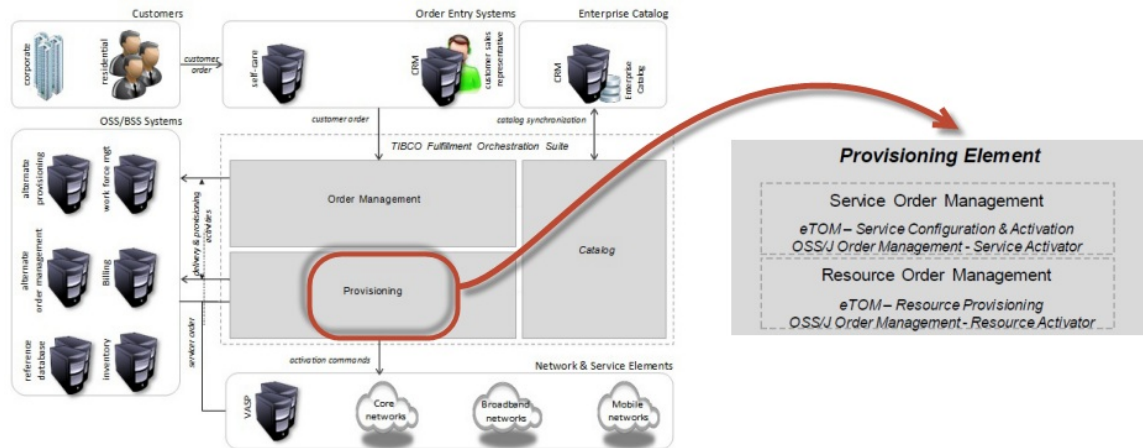
TIBCO Fulfillment Orchestration Suite Components

The TIBCO Fulfillment Orchestration Suite solution is capable of supporting end-to-end order fulfillment with order management, provisioning, catalog, and inventory capabilities.

The following are the constituents of the TIBCO Fulfillment Orchestration Suite:

- TIBCO Order Management - Long Running:** TIBCO Order Management - Long Running is a metadata driven order management and fulfillment system, which allows development of fulfillment plans based on meta-data specified in product catalogs. Order fulfillment and service provisioning is no longer a simple single-service or product workflow. The dynamic bundled offerings along with the explosion of devices, applications, real-time inventory management, and third-party content providers require a complex order fulfillment system, which can adapt to the changes in processes, metadata, and inventory. Traditional Operation Support System or Business Support System approach with the data silos fail to provide a dynamic and agile solution.
- TIBCO® Fulfillment Provisioning:** TIBCO Fulfillment Provisioning is a provisioning component that automates the activation of the underlying network services and allocation of all the network resources. This provisioning element implements service order management and resource order management TAM applications, aligns to service and resource activator Operation Support System order management components, uses eTOM service configuration and activation, and implements resource provisioning processes and functions.
- TIBCO® Product and Service Catalog:** TIBCO Product and Service Catalog is a catalog function that defines and manages life cycles of commercial and technical offerings.
- TIBCO® Fulfillment Subscriber Inventory:** TIBCO Fulfillment Subscriber Inventory maintains a current image of customer products at any given point in time that is capable of supporting fast concurrent read or write access while ensuring data consistency. TIBCO Fulfillment Subscriber Inventory also provides a rich, user-friendly web interface that can be used to explore and modify the contents of the system in a safe and secure manner.

Provisioning in the Orchestration Suite

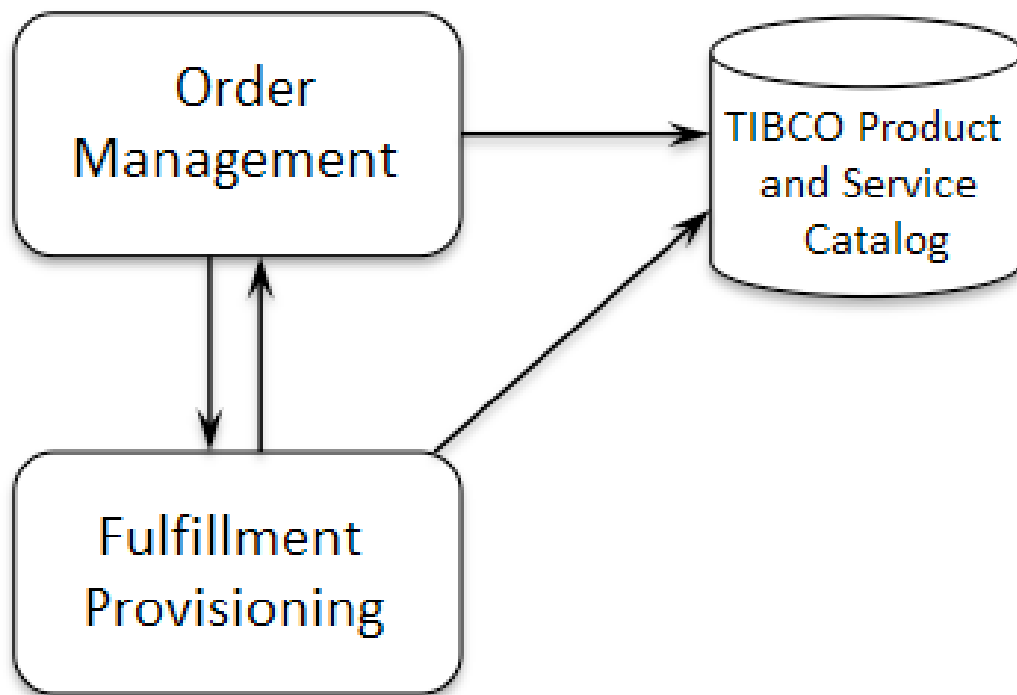


To enable TIBCO Fulfillment Orchestration Suite to provide a truly unified and cohesive solution suite, different components of the suite have been integrated. For instance, the suite provides predefined interconnectivity between TIBCO Fulfillment Provisioning, TIBCO Fulfillment Subscriber Inventory, TIBCO Fulfillment Subscriber Inventory, and TIBCO Product and Service Catalog TIBCO Product and Service Catalog, a catalog concept alignment between TIBCO Product and Service Catalog and TIBCO Fulfillment Provisioning through data synchronization process, and a GUI integration for a similar look-and-feel.

User Interface Integration

TIBCO Order Management - Long Running, TIBCO Fulfillment Subscriber Inventory and TIBCO Product and Service Catalog interact with each other to provide complete fulfillment solution. TIBCO Fulfillment Provisioning, another optional component, enhances the overall fulfillment capability by providing a network provisioning system capable to connect directly to network elements.

The interaction among the three components is fully configurable through the user interface to maintain their anonymity and to ensure that they can work independently.

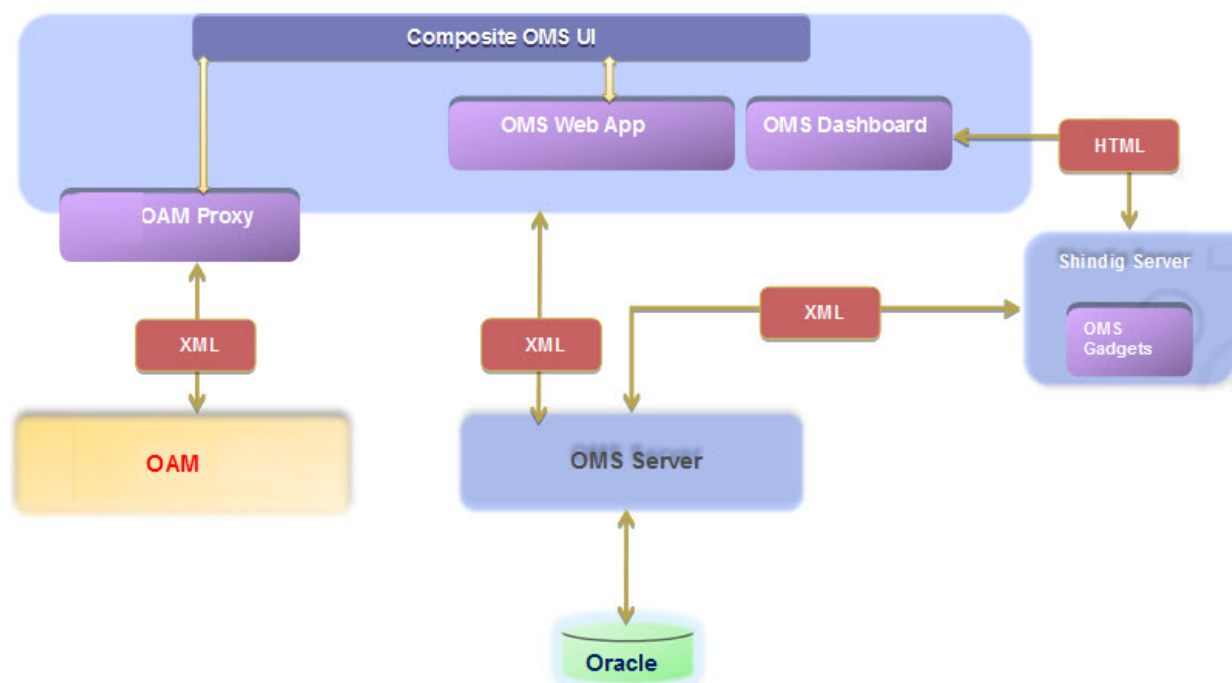
Component Integration

TIBCO Fulfillment Provisioning configuration is done through editable files including script files, such as data mappers, cartridge configuration files, and routing services. The catalog configuration allows you to define all the product catalog related concepts, service catalog related concepts and both depending on what catalog components are deployed. The service catalog determines services and orders associated with a rendering process.

The BPMN standard is used for process modeling. It provides a single and standard process modeling interface for both TIBCO Order Management - Long Running and TIBCO Fulfillment Provisioning.

The following diagram shows the different integration components:

UI Integration



TIBCO Order Management - Long Running and TIBCO Fulfillment Subscriber Inventory Integration

TIBCO Fulfillment Subscriber Inventory component allows external users to assign inventory for customers against orders. These inventories are stored in the TIBCO Fulfillment Subscriber Inventory store. When an order for the same customer and product is made again, the TIBCO Order Management - Long Running Automated Order Plan Development engine synchronizes the inventories for the customer from TIBCO Fulfillment Subscriber Inventory and generates the plan accordingly. The TIBCO Order Management - Long Running interface supplies the inventory to Automated Order Plan Development to be taken into account when the plan generation occurs or when a request is made for an execution plan. For more information, see *TIBCO Fulfillment Subscriber Inventory User's Guide*

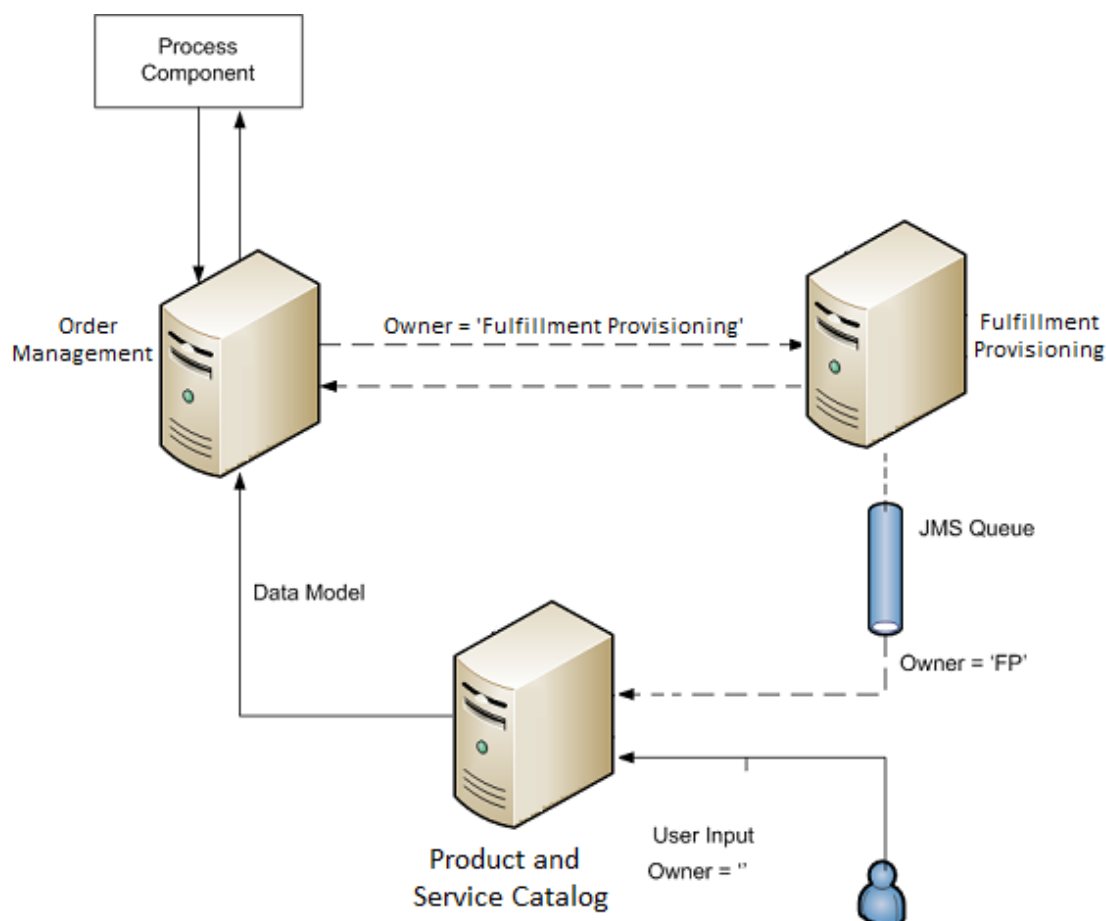
TIBCO Product and Service Catalog and TIBCO Fulfillment Provisioning Integration

The TIBCO Product and Service Catalog and TIBCO Fulfillment Provisioning component integration helps synchronize the TIBCO Fulfillment Provisioning service orders data into the fulfillment catalog with a defined ownership. For more details, see the *TIBCO Order Management - Long Running User's Guide*.

TIBCO Order Management - Long Running and TIBCO Fulfillment Provisioning Integration

The TIBCO Fulfillment Provisioning deployment and configuration with TIBCO Order Management - Long Running adds another distinct feature to the TIBCO Order Management - Long Running. TIBCO Order Management - Long Running (Orchestrator) are routed to TIBCO Fulfillment Provisioning or a user defined process component. The routing is automated by using the message owner information. Based on the owner name, a fulfillment request is sent to a particular component. For example, a fulfillment request is sent to the TIBCO Fulfillment Provisioning component for all the orders with the owner name 'FP'.

TIBCO Order Management - Long Running-TIBCO Fulfillment Provisioning- TIBCO Product and Service Catalog Integration



The user interface invokes the following TIBCO Fulfillment Provisioning features:

- Service Order Tracking
- Service Catalog Editor
- Interface Management

See the *TIBCO Fulfillment Provisioning User's Guide* for more details.

TIBCO Order Management - Long Running Overview

TIBCO Order Management - Long Running is a metadata driven order management and fulfillment system, which allows development of fulfillment plans based on meta-data specified in product catalogs. Order fulfillment and service provisioning is no longer a simple single-service or product workflow. The dynamic bundled offerings along with the explosion of devices, applications, real time inventory management and third-party content providers requires a complex order fulfillment system, which can adapt to the changes in processes, metadata and inventory. Traditional Operation Support System or Business Support System approach with their data silos fail to provide a dynamic and agile solution. An end-to-end order management system based on product and service catalogs is a key differentiator of TIBCO Order Management - Long Running.

TIBCO Order Management - Long Running is a comprehensive software solution to design, deploy and maintain high-performance scalable enterprise level business processes for advanced and dynamic order fulfillment. TIBCO Order Management - Long Running enables companies to quickly introduce new product offerings and in most cases requiring little or no change to fulfillment processes. The product bundles are decomposed into existing products to automatically generate a plan specific to the order.

TIBCO Order Management - Long Running also enables companies to efficiently manage changes to the existing business process to meet rapidly changing business environment.

Product model can be defined following SID 9 guidelines by using TIBCO Product and Service Catalog or any other catalog management system and imported into TIBCO Order Management - Long Running.

Basic Order Management Concepts

In order to understand how TIBCO Order Management - Long Running works and how to use it, it is important to understand the key concepts. Those concepts are generic and are used extensively through the documentation.

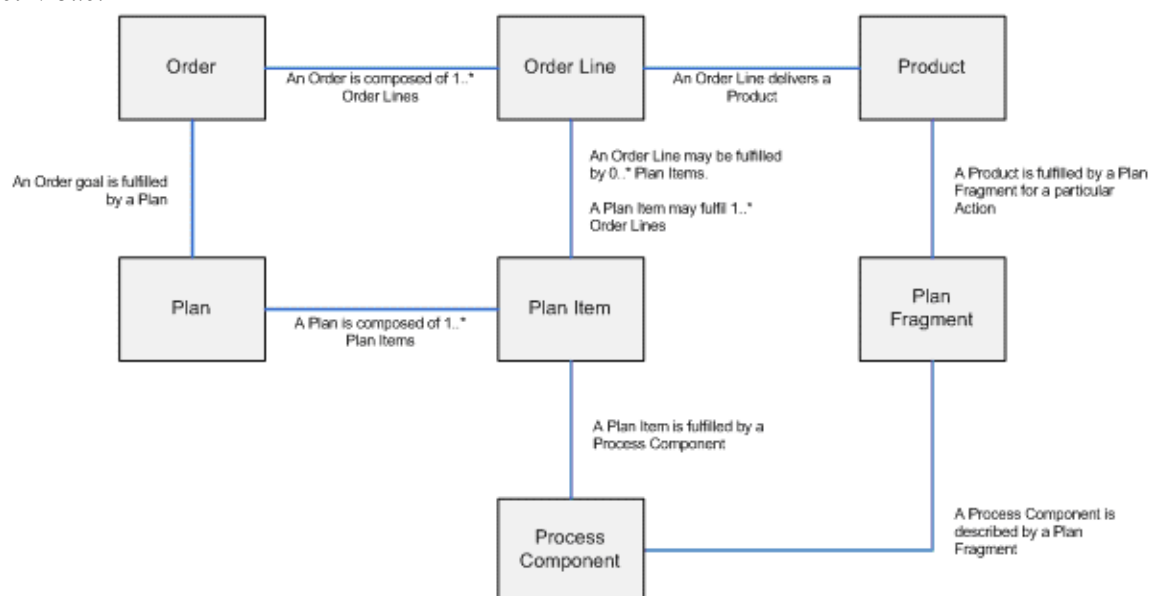
Order

In Order Management, the *order* is a key concept. The TIBCO Order Management - Long Running fulfills *orders*. Typically, an *order* lists products or services that required to be fulfilled. External systems might submit updates to the *order* as amendments, but the components within TIBCO Order Management - Long Running might not change an *order*.

In the context of TIBCO Order Management - Long Running as an *order* is composed of one or more *order lines*. Each *order line* corresponds to a requested *product*.

TIBCO Order Management - Long Running creates an execution *plan* for each *order* received. The *plan* is computed by using a *product model* that is stored in a *Product Catalog*. The *plan* is composed of one or more *plan items*.

Object Model

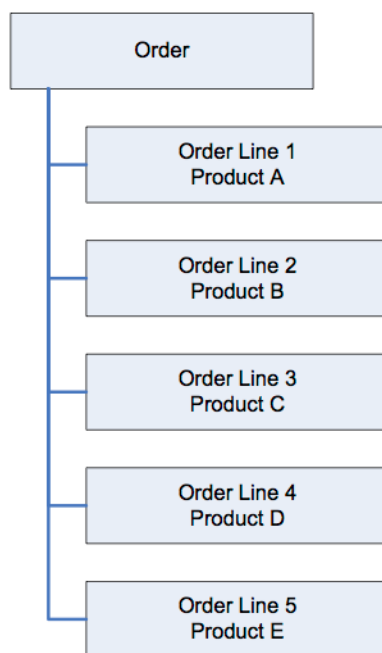


Plan fragment, plan item, and process component are all inter-related concepts. To clarify:

- Plan Item is one step in a plan that must be executed to reach the goal of fulfilling an order. The plan item is configured with the name of the Process Component that must be invoked to fulfill a product. The name of the Process Component is provided by Automated Order Plan Development during plan development and gathered from the Product Catalog by using the name of the product.
- Plan Fragment is the model definition of a Process Component that fulfills a particular product. Products are linked to plan fragments in the Product Catalogue. The name of a plan fragment is the same as the name of the Process Component that it describes.
- Process Component is the physical implementation of the tasks required to fulfill a product. It is described by a plan fragment and invoked as a plan item step in a plan.

The logical relationship between order and order lines is shown in the following diagram:

Order Logical Components



Characteristics

Characteristics might be of the following common predefined types:

- **Feature** – A distinct feature or capability of a product. In general, features distinguish a product from other products of the same class. For example features of a mobile device might include: SMS, Voice, MMS, 4G, Stereo Wireless Headset, Keyboard, etc. Features could also be chargeable or non-chargeable, e.g. For billing purposes a device that provides SMS capability could mean it might need an SMS capable billing plan.
- **Instance** – Instance characteristics are similar to Features, the feature in question has measurable quantity that is defined for each related product. An example would be a discrete “Free 500 SMS Package” product could have an “Instance” characteristic called “Free SMS”. This characteristic would have a relationship value = 500. Another similar product could be created called “Free 1000 SMS Package”. It would have the same “Free SMS” characteristic associated with it but have a relationship value = 1000.
- **Input** – These characteristics represent information values that has to be captured and associated with the product at time of order/order fulfillment. they generally represent information that needs to be propagated to other systems or impacts the fulfillment process. Input characteristics generally have no values until the order is placed/fulfilled. An example of an Input characteristic could be a Mobile Station International Subscriber Directory Number (phone number) allocated to a mobile device, or a “Contact Address” captured for a business internet product at time of order.
- **Shared** - Indicates that the attribute is shared.

Identifying Common Characteristics

The common characteristics/udf in a plan item can be identified in two ways. You can also switch between these two syntaxes by using global variable `EnableAffinityUDFParent` in the Automated Order Plan Development component.

1. One way to identify the common characteristics/udf in a plan item is explained as follows :

For example, consider a scenario where there is an affinity between the two plan items with User Defined Fields sharing the same name but different values and they have plan items decomposed from the same orderline. It was difficult to identify which plan item the characteristic belongs to with the previous syntax. In the new syntax, the value at the end of the last colon (:) is always comma-separated

line numbers and it becomes easier to correlate the characteristic on the basis of `affinityProductID` and `parentProductID`. The changed User Defined Field syntax is as follows:

```
<name:parentProductID:affinityProductID:linenumber>
```

Where:

- `name` = Name of the User Defined Field.
- `parentProductID` = Parent of the affinity product ID.
- `affinityProductID` = Affinity product ID.
- `linenumber` = Comma-separated line numbers to denote from which order lines these characteristics appeared.

This functionality can be enabled by setting the value of global variable `EnableAffinityUDFParent` to `true`. By default, this behavior is disabled.

2. The other way to identify the common characteristics/udf in a plan item is:

```
<UDF>: <OrderLineNumber>
```

Where:

UDF = Name of User Defined Field `OrderLineNumber` = Comma-separated line numbers to denote from which order lines these characteristics appeared

Product

A product is modeled in TIBCO Product and Service Catalog. Orders are composed of order lines, with each order line corresponding to a particular product that is requested by a customer.

For each product that a customer orders, a series of plan items must be completed in order for that product to be provided. The link between product and plan item is maintained in the TIBCO Product and Service Catalog. The rules defining how different products depend on one another is also maintained in the TIBCO Product and Service Catalog. This then translates into dependencies between plan items in the overall execution plan.

Plan

A *plan* represents the tasks to be completed to reach the fulfillment goal. An *order* only ever has one *plan* associated with it, and one *plan* is only ever associated with a single *order*.

To fulfill an order, a series of tasks must be executed in a defined sequence and to a defined schedule. Sequencing and scheduling is modelled by dependencies between tasks. Once all dependencies for a task have been satisfied, then that task might be executed, with the net result being the *order* is fulfilled by following the required steps in the correct order.

Within this software, the series of tasks is represented by a *plan* object. The *plan* defines how to fulfill an *order* as a series of tasks, or *plan items*. *Plan items* are the smallest units of work recognized within this software; however, they might be composed of a series of sub-tasks that actually implement the work required. This can take the form of automated back-end system invocations, manual tasks, or any other unit of work that might be required.

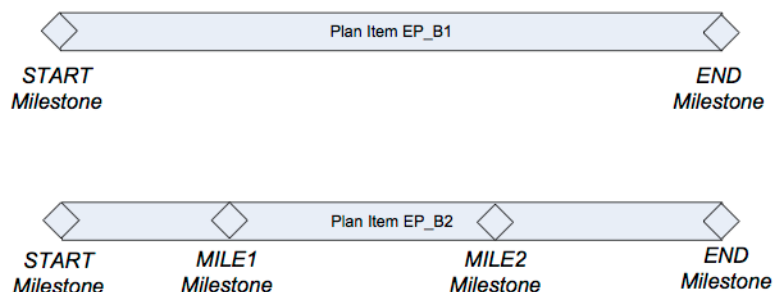
Plan Item

As an *order* consists of *order lines*, a *plan* consists of *plan items*. Each *plan item* represents a set of work that must be performed to fulfill an *order*. Order lines might map onto plan items, but not necessarily. However, plan items always map onto at least one order line. One order line might require multiple plan items to be fulfilled, and likewise multiple order lines might be fulfilled by a single plan item. An order line that doesn't require the completion of any physical tasks is classified as non-executing and does not require a plan item. The abstraction and logic required to map order lines into plan items is implemented in Automated Order Plan Development as part of core TIBCO Order Management - Long Running.

Milestone

Plan items are composed of a series of *milestones*, which represent critical points of execution. All plan items have *start* and *end* milestones, and might have zero to many *intermediate* milestones. This is shown in the following diagram:

Plan Item Milestones



In this example, plan item EP_B1 has only start and end milestones. This represents the basic set of milestones that all plan items contain. Plan item EP_B2 has start milestone, end milestone, and two intermediate milestones MILE1 and MILE2.

Milestones represent the critical points during execution of a plan item and represent the points where Orchestrator can control execution of the plan item. Milestones are points where dependencies might be attached to a plan item. Dependencies might be attached only to start and intermediate milestones and might not be attached to any other point on a plan item.

Dependency

Dependencies are conditions that must be satisfied *before* a milestone can be considered ready. If a milestone is not yet ready, then execution might not proceed past the milestone. In the case of a start milestone, Orchestrator does not request execution of the associated plan item until all attached dependencies are satisfied. In the case of an intermediate milestone, the Process Component must halt execution at the milestone point within its internal process model until notified by Orchestrator that a milestone is ready to fire. At that point the Process Component might continue execution. This notification might occur while the Process Component is waiting at the milestone, or at any point before execution reaches the milestone.

There are three different dependency types:

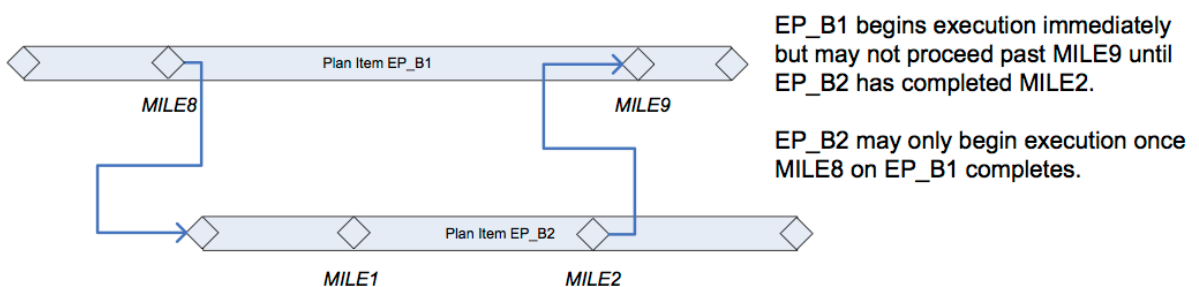
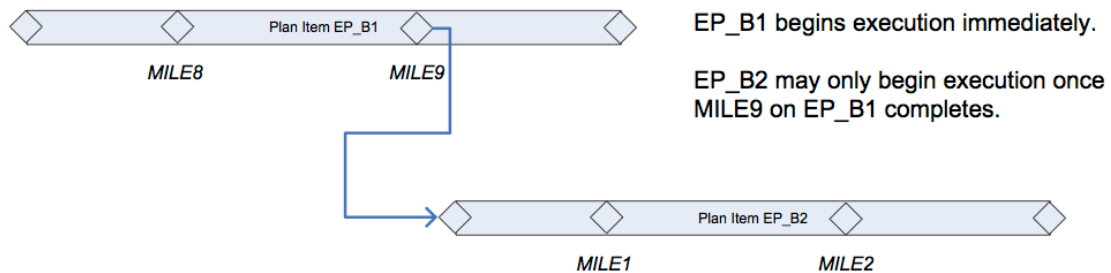
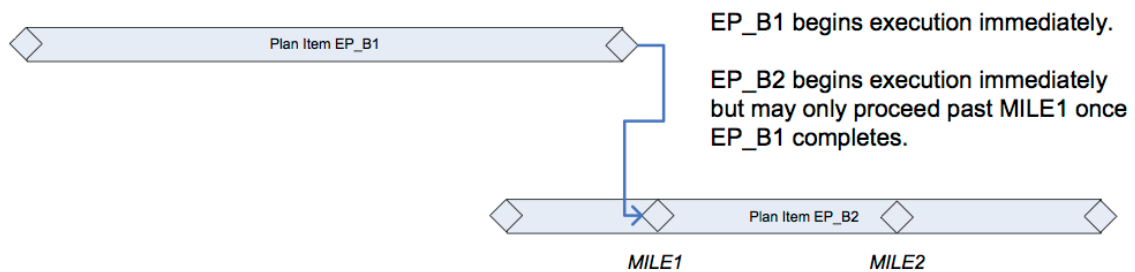
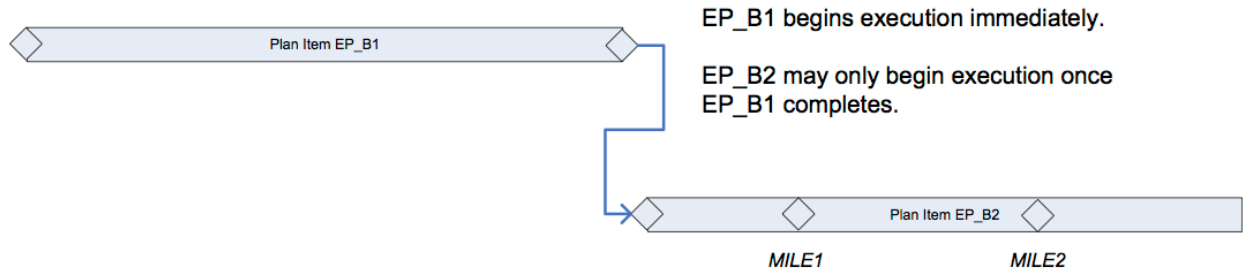
- **External** – satisfied when an external event is received by Orchestrator from an external system.
- **Time** – satisfied when a certain time period has elapsed, or a certain absolute date and time has been reached.
- **Point** – this dependency is satisfied by some milestone in another plan item having made ready.

External	<p>External dependencies rely on an outside event occurring before it is satisfied. This event is identified by the following parameters:</p> <ul style="list-style-type: none"> • Event Name – name identifying the type of event that must occur • Event ID – unique identifier for a given Event Name that identifies an external dependency as being satisfied.
Time	<p>Time dependencies might take the form of an absolute date time, or a relative time delta. If an absolute date and time is specified then this is translated into a time delta from the point where plan execution begins. The time delta is specified in milliseconds. Once the time delta period has passed, then the dependency is considered satisfied.</p>

Point

Point dependencies rely on the execution sequence of other plan items to be satisfied. When a parent plan item reaches a certain intermediate or end milestone, then the dependency is satisfied. Point dependencies might use parent plan items in the current plan or in a completely different plan. If a milestone is not specified, then it is assumed that the end milestone must be made ready for the dependency to be satisfied.

Some common point dependency scenarios are shown in the following diagram:

Point Dependencies

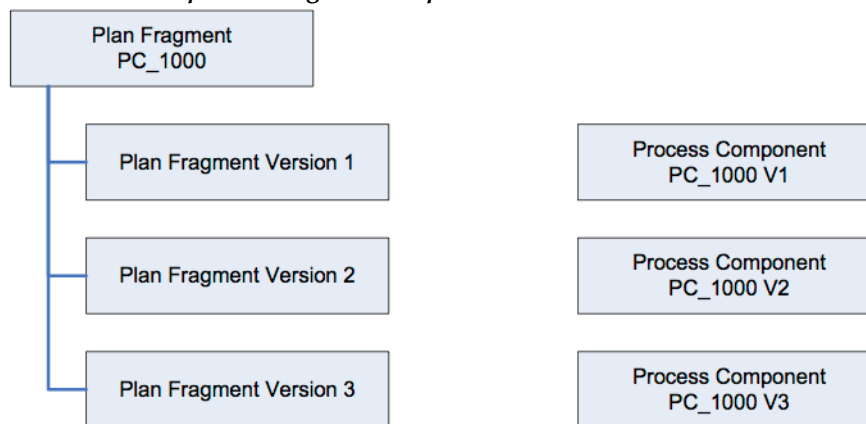
A milestone might have zero to many dependencies attached and dependency types might be mixed between external, time, and point. A milestone might have multiple external and point dependencies, but only one time dependency is permitted. If a milestone does not have any dependencies then it is made ready immediately. Otherwise the milestone is only made ready once all dependencies are satisfied.

Plan Fragment

A *plan fragment* is an abstraction of a Process Component that contains configuration information that Orchestrator requires to handle errors and SLA notifications. Plan fragments are optional. If no plan fragment is defined for a particular Process Component then Orchestrator uses engine defaults to handle errors and no SLA notifications occurs.

During the evolution of a system it might be necessary to deploy multiple versions of a Process Component simultaneously. To support this plan fragments might be versioned. The relationship between plan fragments, versions, and process components is shown in the following diagram:

Plan Fragment and Process Component Logical Components



In this example plan fragment PC_1000 describes Process Component PC_1000. This Process Component might be invoked by a plan item by using different versions. Therefore plan fragment PC_1000 has Version 1 that maps the Process Component of the same version. The base plan fragment always defines the currently active Process Component version.



The jeopardy detection and rules for consequential action are not applied for any execution containing process component mapping with no valid plan fragment model.

Error Handling

In the event that a Process Component returns a failed or incomplete execution response Orchestrator handles the error by using engine default configurations. Standard error handling functionality is to retry the plan item for a defined number of times, with a defined delay interval between invocations before referring it to the Plan Item Error Handler with the name of a default error handler for manual intervention. If a plan fragment is defined for a Process Component, these error handling properties might be overridden.

If the retry override flag is set to true, then the plan fragment configuration for retry is used instead of the engine configuration. The following parameters might be set:

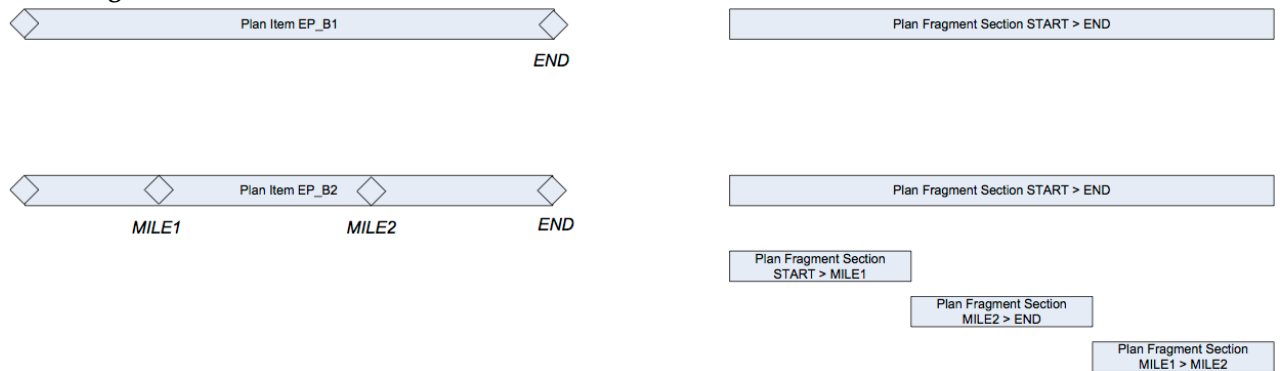
- **Retry Count** – the number of times to retry the plan item on failure before referring it to the Plan Item Error Handler.
- **Retry Delay** – the delay in msec to wait between plan item retries.

When invoking the Plan Item Error Handler Orchestrator specifies the error handling module that is relevant for the plan item being submitted. Generally this is a default error handler, but the name of the error handler might be overridden by specifying it in the plan fragment.

SLA Notification

SLA notifications are sent out by Orchestrator when plan items exceed expected typical, threshold [specific percentage of maximum duration] and maximum execution durations. These typical and maximum durations are modelled on plan fragment sections, which represent the part of a plan item that executes between two milestones. For example:

Plan Fragment Sections

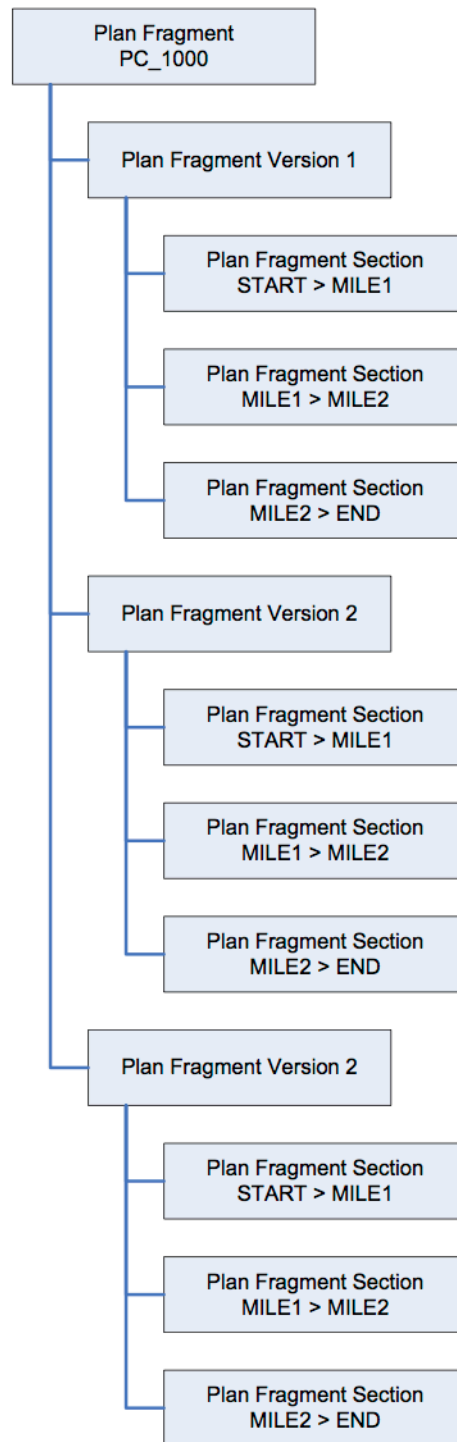


In this example plan item EP_B1 has an associated plan fragment with a single plan fragment section that defines the typical and maximum execution duration between the start and end milestones.

Plan item EP_B2 has an associated plan fragment also with a plan fragment section that defines the typical and maximum execution duration between the start and end milestones. But it also has 3 other sections that define typical and maximum execution durations between start and mile1, mile1 and mile2, and mile2 and end.

Plan fragment sections are also versioned, so they exist within the plan fragment hierarchy as follows:

Plan Fragment Logical Components



Plan Development

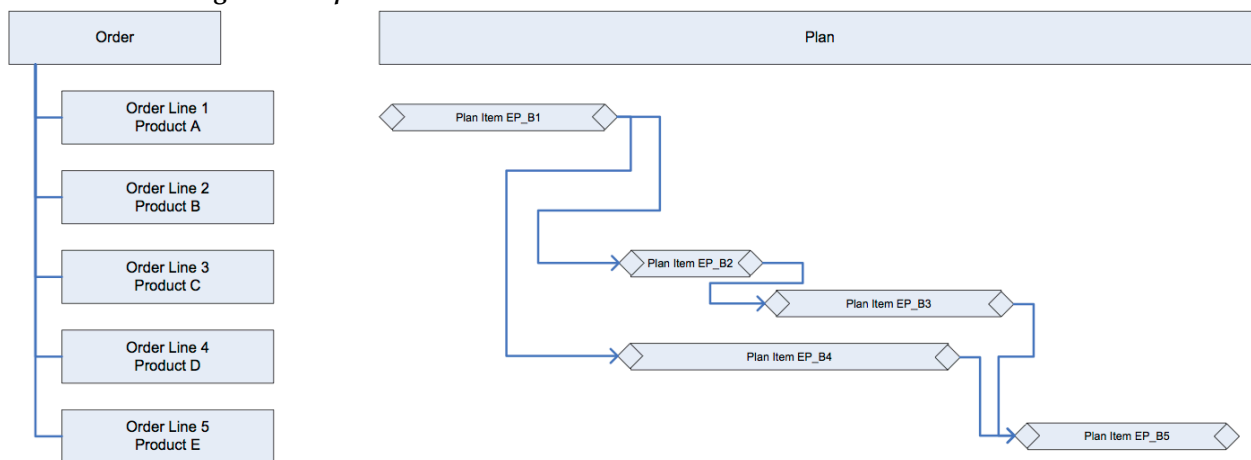
During order plan development, Orchestrator calls out to Automated Order Plan Development to design the plan of action to fulfill an order. Automated Order Plan Development analyzes the order and the Product Catalog and determine what plan items must be executed and in what sequence to fulfill an Order.

Automated Order Plan Development looks at the product on an order line and determine one of the following scenarios:

- The product is non-executing, in which case no plan item is required and the order line is marked as complete.
- The product requires a single plan item. When this plan item completes then the order line is marked as complete.
- The product requires multiple plan items to be executed in a defined sequence. The final plan item in the sequence is flagged as end-of-line (EOL). When this plan item completes then the order line is marked as complete.
- The product has previously been provisioned. The plan item or items is created as above, but the status set immediately to complete. The order line is also marked as complete.
- Plan item dependencies are determined by rules and the configuration of the product model hierarchy

The logical relationship between order, order lines, plan, and plan items is shown in the following diagram:

Order and Plan Logical Components



In this example, Automated Order Plan Development has analyzed the order and determined the following:

1. Order Line 1 orders Product A. This product is fulfilled by Process Component EP_B1. This component has no dependencies so it begins execution immediately.
2. Order Line 2 orders Product B. This is a non-executing product so there is no associated plan item and no execution occurs.
3. Order Line 3 orders Product C. This product is fulfilled by two Process Components that execute in series EP_B2 followed by EP_B3. Component EP_B2 might only begin execution once EP_B1 completes, and component EP_B3 might only begin execution once EP_B2 completes.
4. Order Line 4 orders Product D. This product is fulfilled by Process Component EP_B4. This component might only begin execution once EP_B1 completes.
5. Order Line 5 orders Product E. This product is fulfilled by Process Component EP_B5. This component might only begin execution once both EP_B3 and EP_B4 complete.

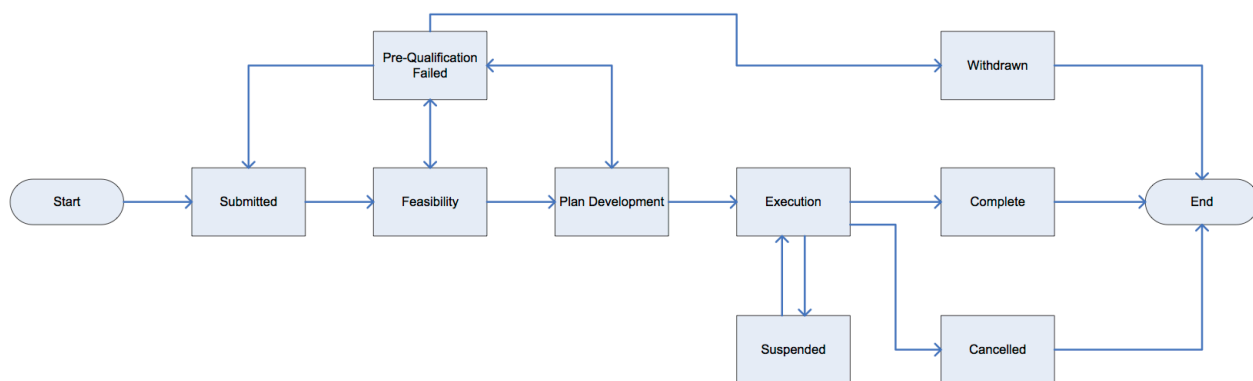
Lifecycle

During fulfillment the different entities managed by Orchestrator go through a defined status lifecycle.

Order

The order lifecycle is shown in the following diagram:

Order Lifecycle



The lifecycle steps are summarized below:

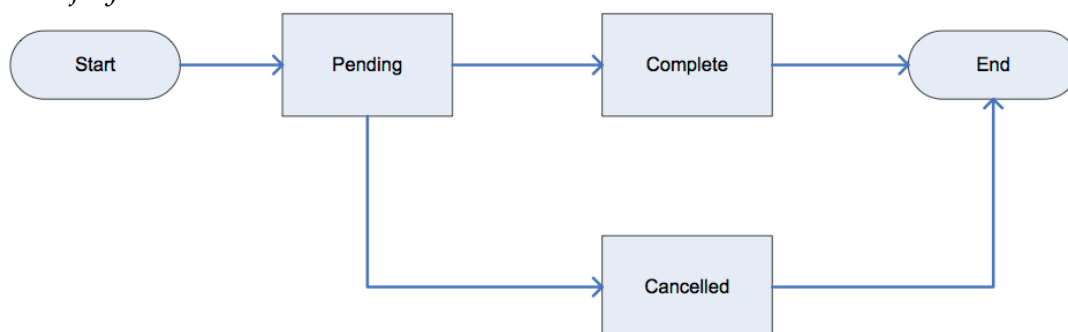
Step	Description
Submitted	The order has been submitted to Orchestrator and successfully stored in the database. Processing has started.
Feasibility	<p>The order is currently being evaluated for feasibility.</p> <p>If feasibility has been enabled, Orchestrator has requested a feasibility analysis from the Feasibility Provider. If the Feasibility Provider deems the order not feasible then the next status is Pre-Qualification Failed if feasibility error handling is enabled, or Withdrawn if error handling is not enabled. Otherwise the next status is Plan Development.</p> <p>Otherwise the order is automatically deemed feasible and the next status is Plan Development.</p>
Plan Development	<p>The order has been deemed feasible and is currently being analyzed for plan development. Orchestrator has sent a request to Automated Order Plan Development to design a plan.</p> <p>If Automated Order Plan Development is unable to design a plan then the next status is Pre-Qualification Failed if OPD error handling is enabled, or Withdrawn if error handling is not enabled. Otherwise the next status is Execution.</p>
Execution	<p>The order is feasible and a plan has been created and linked to the order. The plan has been successfully stored in cache and is progressing through the plan lifecycle. All order lines are Pending. The order is being fulfilled by Orchestrator.</p> <p>If the order completes successfully then the next status is Complete. This includes any order amendment except for a full order cancel. If the order has been canceled completely then the next status is Cancelled.</p> <p>If a suspend request is made for the order then the next status gets Suspended.</p>

Step	Description
Complete	The order has completed fulfillment successfully. The plan and all plan items in the plan are Complete, and all order lines are Complete. The order status in the database has been updated to completed.
Pre-Qualification Failed	<p>The order does not meet the criteria for feasibility or for plan development and has been referred to a Pre-Qualification Failed Handler for further analysis before proceeding.</p> <p>If the Pre-Qualification Failed Handler returns Order Request then that order is submitted as an amendment and the next order status is Submitted.</p> <p>If the Pre-Qualification Failed Handler returns Retry OPD then the next order status is OPD.</p> <p>If the Pre-Qualification Failed Handler returns Retry Feasibility then the next order status is Feasibility. Likewise if it returns Retry Plan Development then the next order status is Plan Development.</p> <p>If the Pre-Qualification Failed Handler returns Withdraw then the next order status is Withdrawn.</p>
Suspended	The order has stopped fulfillment and is awaiting further instructions. A suspended order might be returned to Execution either with or without an amendment applied.
Cancelled	<p>The order has canceled fulfilment successfully. The plan and all plan items in the plan are Cancelled, and all order lines are Cancelled. The order status in the database has been updated to canceled.</p> <p>Note that the order only goes to this status in the event of a full order cancel. If a partial order cancel occurs then the order goes to Complete status.</p>
Withdrawn	The order has been withdrawn and deleted from the database.

Order Line

The order line lifecycle is shown in the following diagram:

Order Item Lifecycle



The lifecycle steps are summarized below:

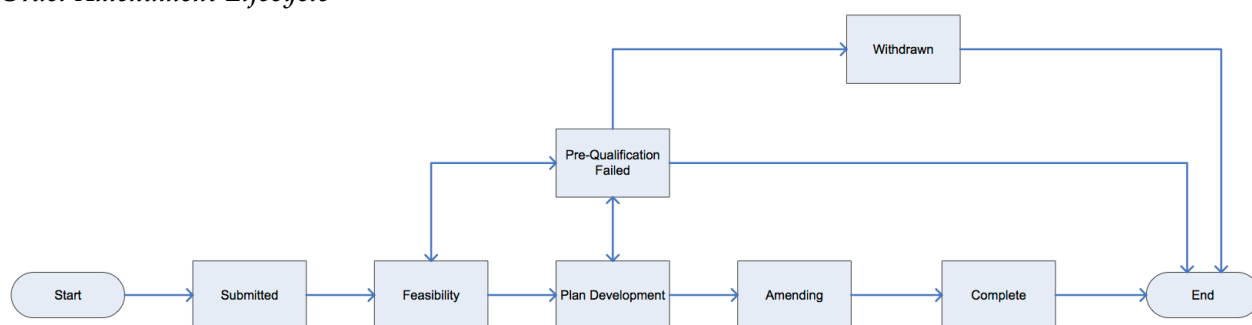
Step	Description
Pending	The order has been successfully submitted and the plan has been developed. The associated plan items for the order line either are ready to execute or currently executing.
Complete	The order line has been successfully completed. The plan item that has specified this order line as end-of-line (EOL) has successfully completed.
Cancelled	The order line has been successfully cancelled. The plan item that has specified this order line as end-of-line (EOL) has successfully cancelled.

Order Amendment

The order amendment lifecycle only occurs for orders that are currently executing. For amendments that occur before execution, then any previous plan is simply discarded and recreated by using the new order.

For amendments during execution the order amendment lifecycle is shown in the following diagram:

Order Amendment Lifecycle



The lifecycle steps are summarized below:

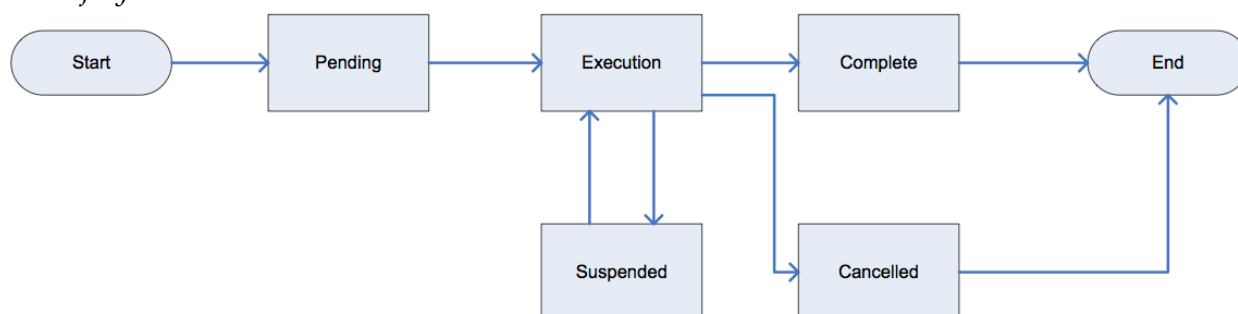
Step	Description
Submitted	The order amendment has been submitted to Orchestrator and successfully stored in cache. Processing has started.
Feasibility	<p>The order amendment is currently being evaluated for feasibility.</p> <p>If feasibility has been enabled, Orchestrator has requested a feasibility analysis from the Feasibility Provider. If the Feasibility Provider deems the order amendment not feasible then the next status is Pre-Qualification Failed if feasibility error handling is enabled, or Withdrawn if error handling is not enabled. Otherwise the next status is Plan Development.</p> <p>Otherwise the order amendment is automatically deemed feasible and the next status is Plan Development.</p>

Step	Description
Plan Development	<p>The order amendment has been deemed feasible and is currently being analyzed for plan development. Orchestrator has sent a request to Automated Order Plan Development to design a plan.</p> <p>If Automated Order Plan Development is unable to design a plan then the next status is Pre-Qualification Failed if Automated Order Plan Development error handling is enabled, or Withdrawn if error handling is not enabled. Otherwise the next status is Execution.</p>
Amending	The order amendment is being processed by Orchestrator and the plan updated.
Complete	The order amendment has completed successfully. The plan has been updated.
Pre-Qualification Failed	<p>The order amendment does not meet the criteria for feasibility or for plan development and has been referred to a Pre-Qualification Failed Handler for further analysis before proceeding.</p> <p>If the Pre-Qualification Failed Handler returns New Order then that order amendment is submitted as a new amendment and the current amendment stops.</p> <p>If the Pre-Qualification Failed Handler returns Retry Feasibility then the next status is Feasibility. Likewise if it returns Retry Plan Development then the next status is Plan Development.</p> <p>If the Pre-Qualification Failed Handler returns Withdraw then the next status is Withdrawn.</p>
Withdrawn	The order amendment has been withdrawn and deleted from cache.

Plan

The overall plan lifecycle is show in the following diagram:

Plan Lifecycle



The lifecycle steps are summarized below:

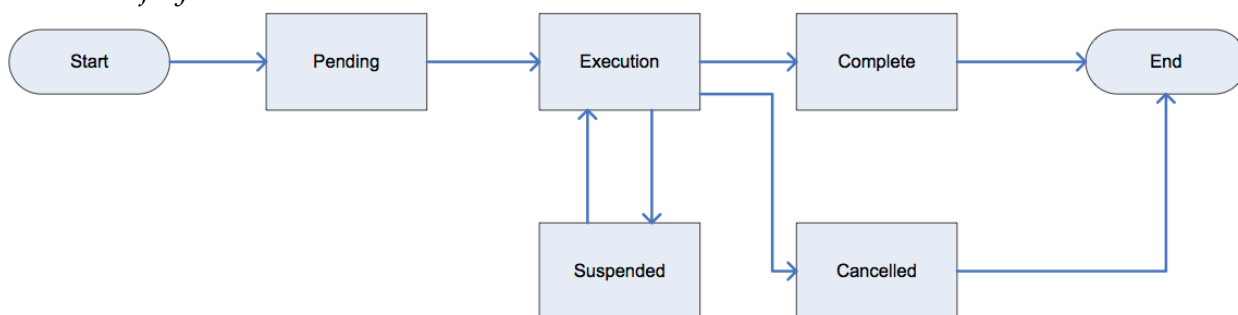
Step	Description
Pending	The plan has been successfully created and is awaiting execution.

Step	Description
Execution	<p>The plan is currently being executed. Plan item requests are being sent to the appropriate Process Components in the correct sequence.</p> <p>If the plan completes successfully then the next status is Complete. This includes any order amendment except for a full plan cancel. If the plan has been canceled completely then the next status is Cancelled.</p> <p>If a suspend plan request is made then the next status is Suspended.</p>
Complete	The plan has completed.
Suspended	<p>The plan has stopped fulfillment and is awaiting further instructions. A plan order might be returned to Execution either with or without an amendment applied.</p>
Cancelled	The plan has been successfully canceled.

Plan Item

The overall plan item lifecycle is shown in the following diagram:

Plan Item Lifecycle



The lifecycle steps are summarized below:

Step	Description
Pending	The plan item has been successfully created and is awaiting execution. This waits for required dependencies to be satisfied before proceeding.
Execution	<p>The plan item is currently being executed. A request has been issued to the Process Component to execute.</p> <p>If the Process Component returns a result that it completed successfully then the next status is Complete. If the Process Component returns a result that it canceled successfully then the next status is Cancelled.</p> <p>If a suspend plan item request is made and the Process Component returns a result indicating that the suspend was successful then the next status is Suspended.</p>

Step	Description
Complete	The plan item has completed. The Process Component has returned a response and indicated that it completed successfully.
Suspended	The plan item has stopped fulfillment and is awaiting further instructions. A request was sent to the Process Component to suspend execution, and it returned a successful suspend response.
Cancelled	The plan item has been successfully canceled.

Milestone

The overall milestone lifecycle is shown in the following diagram:

Milestone Lifecycle



The lifecycle steps are summarized below:

Step	Description
Pending	The milestone is waiting for all its dependencies to be satisfied.
Complete	All dependencies have been satisfied for the milestone and execution has continued.

Dependency

The overall dependency lifecycle is shown in the following diagram:

Dependency Lifecycle



The lifecycle steps are summarized below:

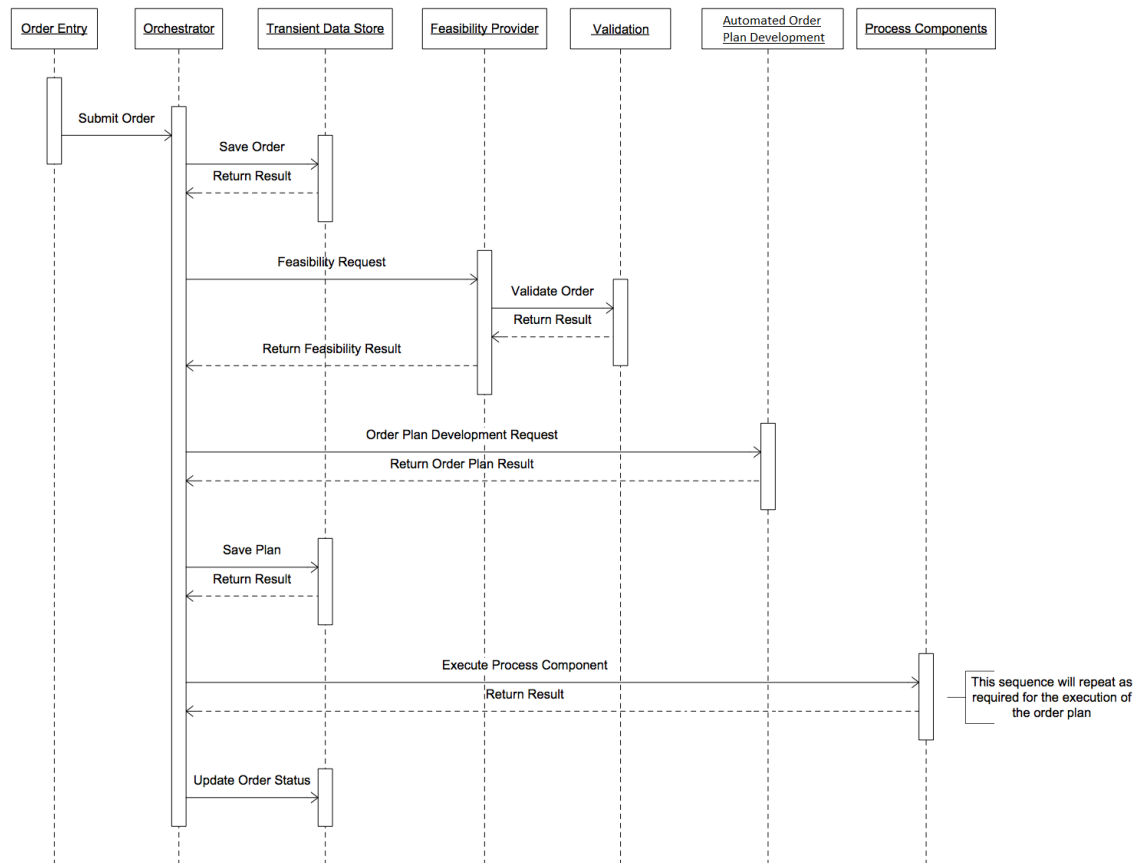
Step	Description
Pending	The dependency is waiting to be satisfied.
Complete	The dependency has been satisfied.

Sequences

Standard Order

This is the normal flow of events through standard order fulfillment with no exceptions. The sequence is shown in the following diagram:

Standard Order Fulfillment – Successful Completion Sequence



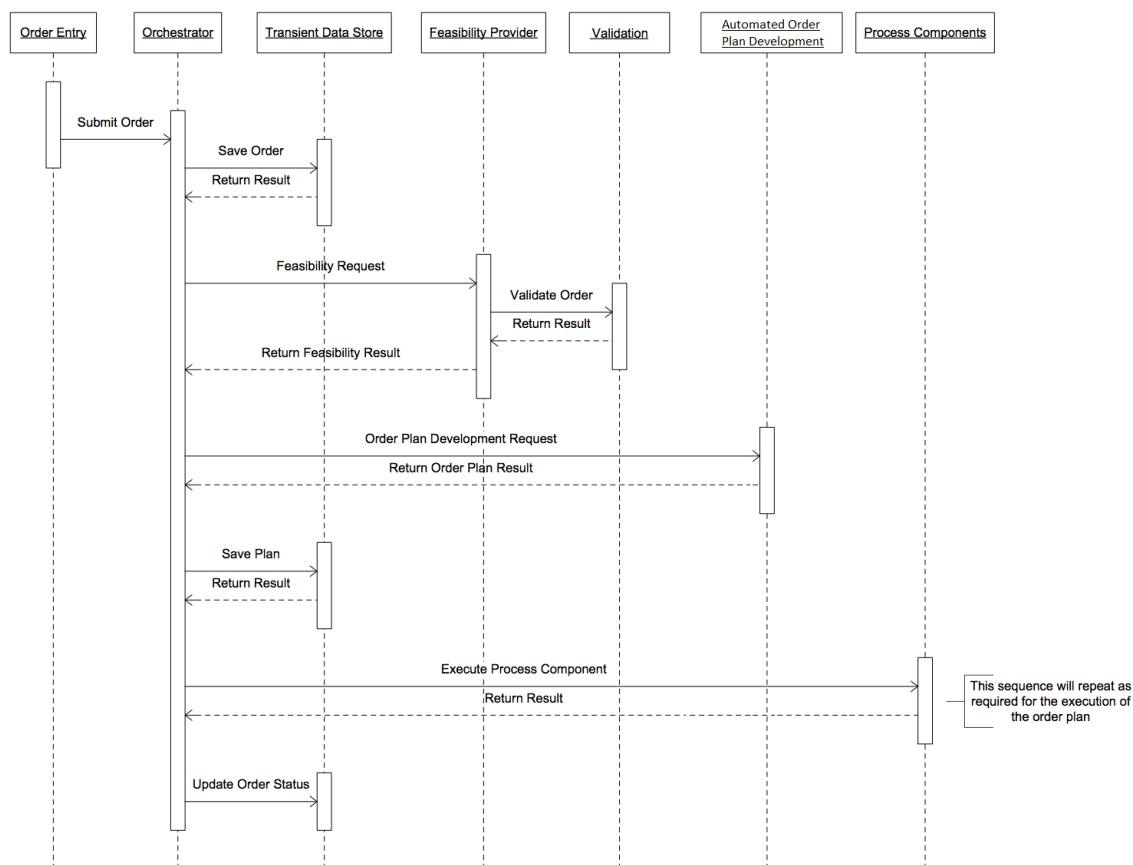
1. The order is submitted from Order Entry through the Orchestrator Submit Order HTTP protocol. Note that this might use intermediate service layers. The order now has Start status
2. Orchestrator sends a request to cache to store the order. The order now has Submitted status.
3. Cache saves the order and returns a response to Orchestrator. The order now has Feasibility status.
4. If feasibility is enabled, Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
 - a. Feasibility Provider might delegate the feasibility checking call to Validation and perform some internal checking.
 - b. Validation returns the result of the order validation back to Feasibility Provider.
 - c. Feasibility Provider aggregates all order feasibility checks and concludes that the order is feasible and sends a response back to Orchestrator.
5. If feasibility is not enabled or the Feasibility Provider has returned the result then the order status is now Plan Development.
6. Orchestrator sends a request to Automated Order Plan Development to analyze the order and design an execution plan.
7. Automated Order Plan Development sends a response back to Orchestrator with the execution plan definition. Orchestrator then generates a plan based on this definition. The order now has Execution status and the plan now has Start status.
8. Orchestrator sends a request to cache to store the plan. The plan now has Submitted status.
9. Cache saves the plan and returns a response to Orchestrator. The plan now has Pending status.

10. Orchestrator changes the plan status to Execution and begins invoking Process Components in the correct sequence. This is repeated for each plan item.
11. Process Component returns a response to Orchestrator for each invocation.
12. Once all plan items have completed, the plan is set to Complete status. Orchestrator sends a request to cache to change the order status to complete for archiving. The order status then goes to Complete in Orchestrator.

Successful Completion

This is the normal flow of events through standard order fulfillment with no exceptions. The sequence is shown in the following diagram:

Standard Order Fulfillment – Successful Completion Sequence



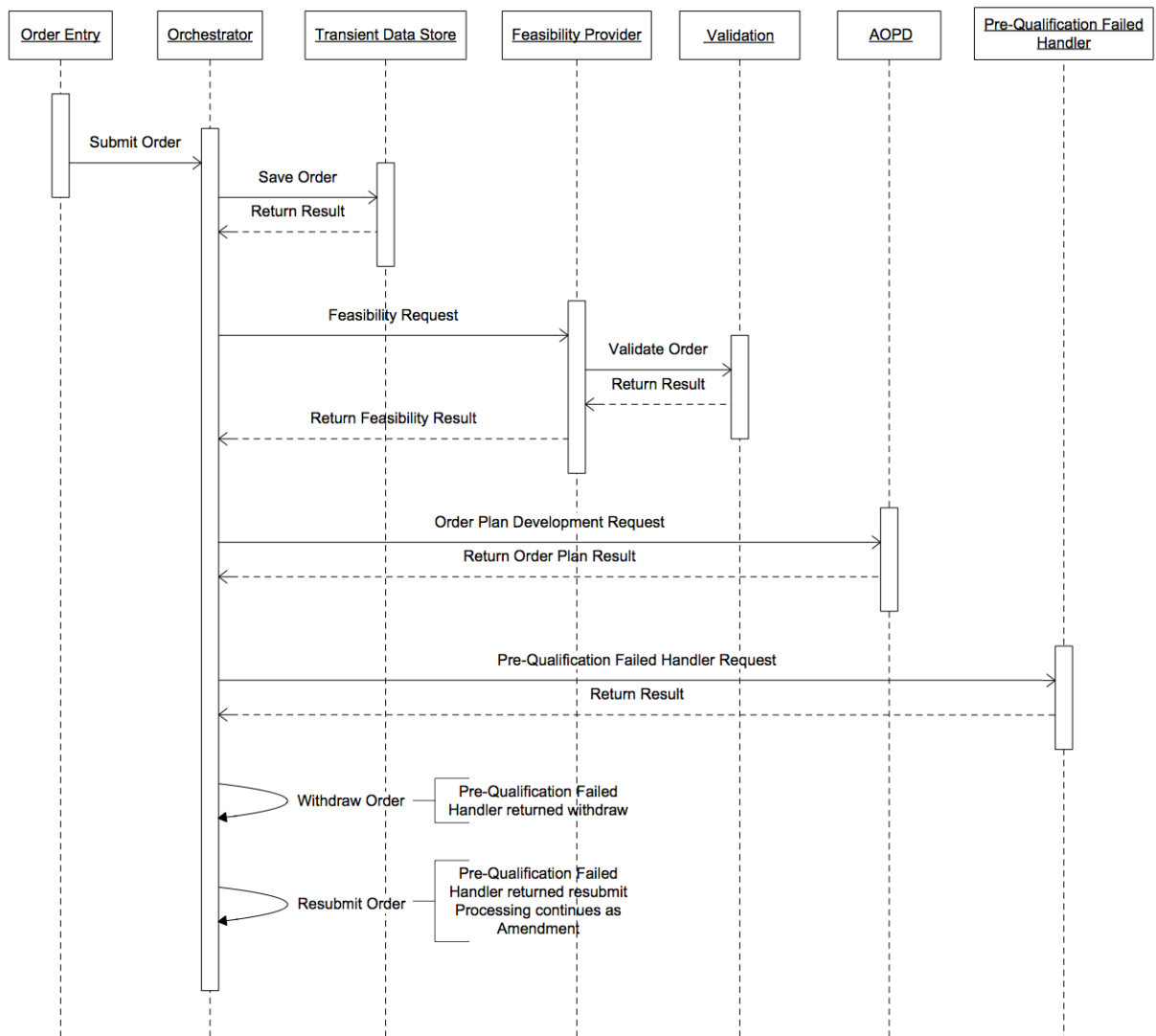
1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status.
2. Orchestrator sends a request to Transient Data Store to store the order. The order now has Submitted status.
3. Transient Data Store saves the order and returns a response to Orchestrator. The order now has Feasibility status.
4. If feasibility is enabled, Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
 - a. Feasibility Provider might delegate the feasibility checking call to Validation and perform some internal checking.

- b. Validation returns the result of the order validation back to Feasibility Provider.
 - c. Feasibility Provider aggregates all order feasibility checks and concludes that the order is feasible and sends a response back to Orchestrator.
- 5. If feasibility is not enabled or the Feasibility Provider has returned the result then the order status is now Plan Development.
- 6. Orchestrator sends a request to Automated Order Plan Development to analyze the order and design an execution plan.
- 7. Automated Order Plan Development sends a response back to Orchestrator with the execution plan definition. Orchestrator then generates a plan based on this definition. The order now has Execution status and the plan now has Start status.
- 8. Orchestrator sends a request to Transient Data Store to store the plan. The plan now has Submitted status.
- 9. Transient Data Store saves the plan and returns a response to Orchestrator. The plan now has Pending status.
- 10. Orchestrator changes the plan status to Execution and begins invoking Process Components in the correct sequence. This is repeated for each plan item.
- 11. Process Component returns a response to Orchestrator for each invocation.
- 12. Once all plan items have completed, the plan is set to Complete status. Orchestrator sends a request to Transient Data Store to change the order status to complete for archiving. The order status then goes to Complete in Orchestrator.

Feasibility Failed

This is the flow of events through standard order fulfillment with the Feasibility Provider indicating that the order is not feasible. The sequence is shown in the following diagram:

Standard Order Fulfillment – Feasibility Failed Sequence



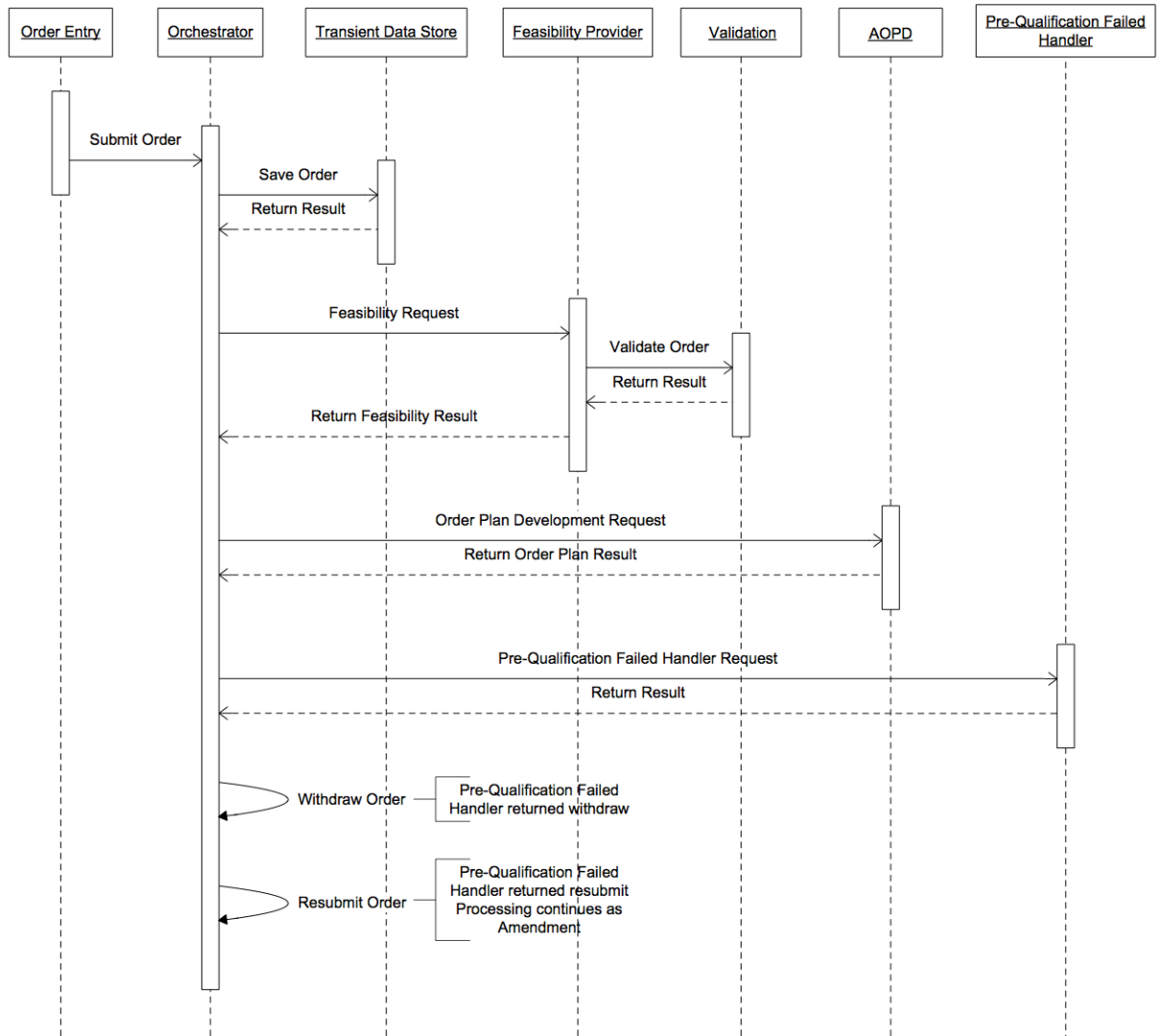
1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status.
2. Orchestrator sends a request to Transient Data Store to store the order. The order now has Submitted status.
3. Cache saves the order and returns a response to Orchestrator. The order now has Feasibility status.
4. Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
5. Feasibility Provider might delegate the feasibility checking call to AFF Validation as well as perform some internal checking.
6. Validation returns the result of the order validation back to Feasibility Provider.
7. Feasibility Provider aggregates all order feasibility checks and concludes that the order is not feasible and sends a response back to Orchestrator. The order status is now Pre-Qualification Failed.
8. Orchestrator sends the order and the validation messages from the Feasibility Provider to the Pre-Qualification Failed Handler for manual intervention.
9. Pre-Qualification Failed Handler sends a response back to Orchestrator with one of two possible actions:

- a. Withdraw the order, which terminates execution and the order status is now Withdrawn.
- b. Resubmit the order with an updated version. This flow continues as a normal order amendment flow.

Plan Development Failed

This is the flow of events through standard order fulfillment with Automated Order Plan Development indicating that the order is not valid for plan development. The sequence is shown in the following diagram:

Standard Order Fulfillment – Plan Development Failed Sequence



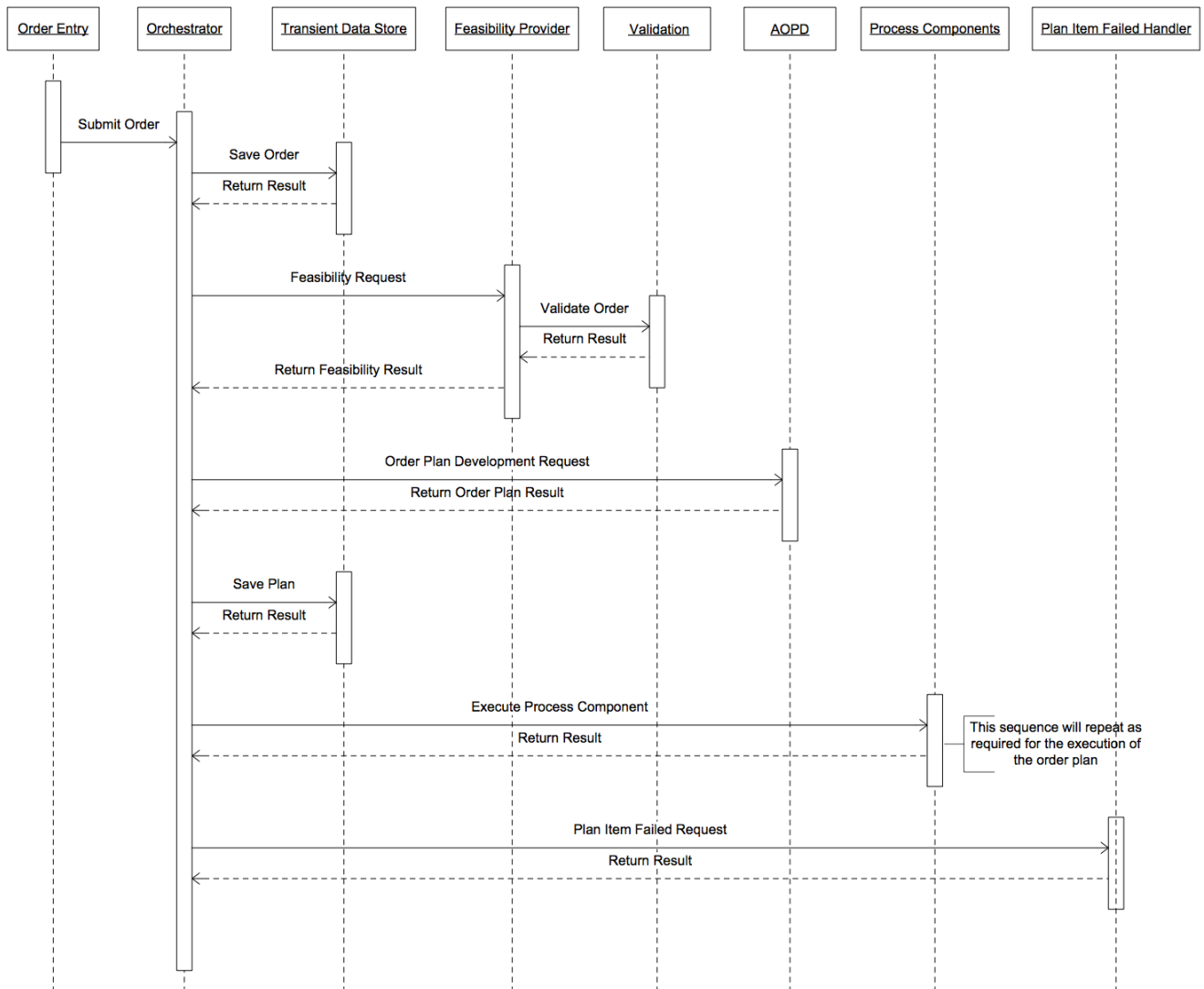
1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status.
2. Orchestrator sends a request to Transient Data Store to store the order. The order now has Submitted status.
3. Transient Data Store saves the order and returns a response to Orchestrator. The order now has Feasibility status.

4. If feasibility is enabled, Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
 - a. Feasibility Provider might delegate the feasibility checking call to Validation as well as perform some internal checking.
 - b. Validation returns the result of the order validation back to Feasibility Provider.
 - c. Feasibility Provider aggregates all order feasibility checks and concludes that the order is feasible and sends a response back to Orchestrator.
5. If feasibility is not enabled or the Feasibility Provider has returned the result then the order status is now Plan Development.
6. Orchestrator sends a request to Automated Order Plan Development to analyze the order and design an execution plan.
7. Automated Order Plan Development sends a response back to Orchestrator indicating the order is not valid for plan design. The order status is now Pre-Qualification Failed.
8. Orchestrator sends the order and the validation messages from Automated Order Plan Development to the Pre-Qualification Failed Handler for manual intervention.
9. Pre-Qualification Failed Handler sends a response back to Orchestrator with one of two possible actions:
 - a. Withdraw the order, which terminates execution and the order status is now Withdrawn.
 - b. Resubmit the order with an updated version. This flow continues as a normal order amendment flow.

Plan Item Execution Failed

This is the flow of events through standard order fulfillment with one or many Process Components indicating that execution of a plan item has failed. The sequence is shown in the following diagram:

Standard Order Fulfillment – Plan Item Execution Failed Sequence



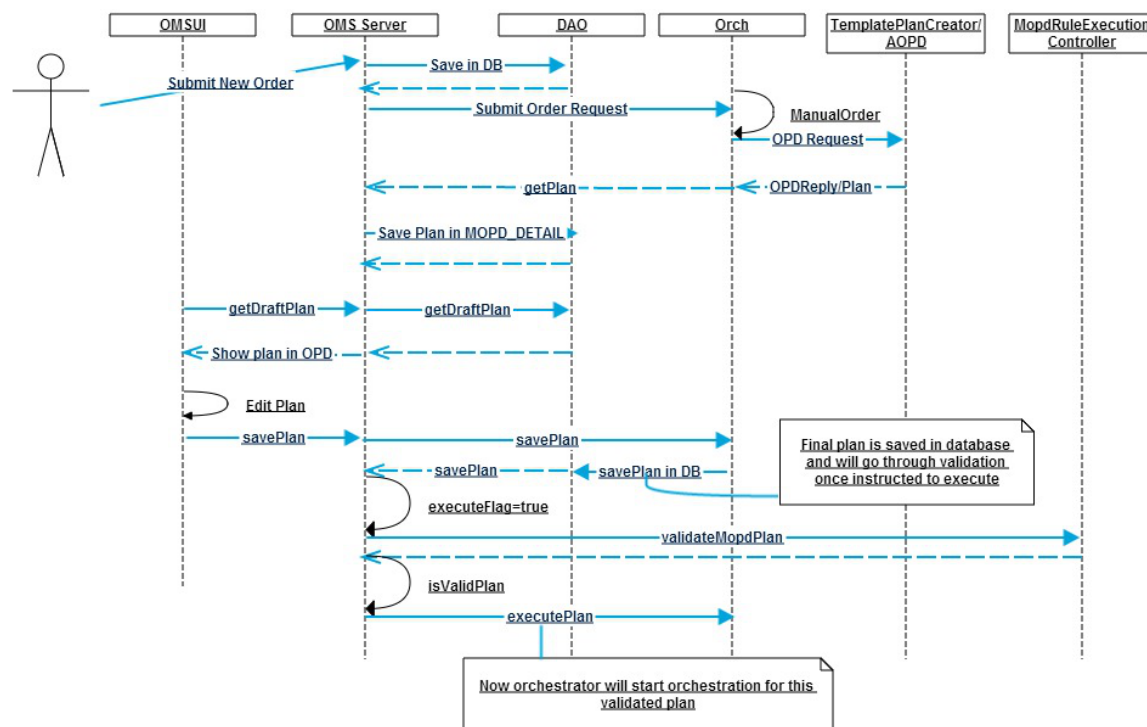
1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status
2. Orchestrator sends a request to Transient Data Store to store the order. The order now has Submitted status.
3. Transient Data Store saves the order and returns a response to Orchestrator. The order now has Feasibility status.
4. If feasibility is enabled, Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
 - a. Feasibility Provider might delegate the feasibility checking call to Validation as well as perform some internal checking.
 - b. Validation returns the result of the order validation back to Feasibility Provider.
 - c. Feasibility Provider aggregates all order feasibility checks and concludes that the order is feasible and sends a response back to Orchestrator.
5. If feasibility is not enabled or the Feasibility Provider has returned the result then the order status is now Plan Development.

6. Orchestrator sends a request to Automated Order Plan Development to analyze the order and design an execution plan.
7. Automated Order Plan Development sends a response back to Orchestrator with the execution plan definition. Orchestrator then generates a plan based on this definition. The order now has Execution status and the plan now has Start status.
8. Orchestrator sends a request to Transient Data Store to store the plan. The plan now has Submitted status.
9. Transient Data Store saves the plan and returns a response to Orchestrator. The plan now has Pending status.
10. Orchestrator changes the plan status to Execution and begins invoking Process Components in the correct sequence. This is repeated for each plan item.
11. Process Component returns a response to Orchestrator for each invocation.
12. If a Process Component response indicates that the invocation was incomplete or resulted in an error, then Orchestrator sends a request to the Plan Item Error Handler with the details of the plan item failure for manual intervention.
13. The Plan Item Error Handler sends a response back to Orchestrator with one of three possible actions:
 - a. Complete the plan item and continue execution as normal.
 - b. Resume the plan item from the point the error occurred and continue execution as normal.
 - c. Retry the plan item from the beginning and continue execution as normal.

Processing New Order with Manual Order Plan Development Enabled

The sequence diagram for processing a new order with Manual Order Plan Development enables is as follows:

Sequence Diagram for Processing New Order with Manual Order Plan Development Enabled



For a newly submitted order, a template plan is generated which can be edited by the user from TIBCO Order Management - Long Running UI. This template plan can be generated using following template implementation:

- No Template
- Default Template
- Custom Template
- JMS Template
- Using Automated Order Plan Development

The following table provides the implementation details for generating template plan. Some of these implementations are supported by the application and user customization of implementation is also supported.

Property Name	Property Value	Usage
com.tibco.fom.orch.mopd.templatePlan	No	No Template Implementation is used.
	Default	Default Template Implementation is used.
	Custom	Custom Template Implementation is used. Fully qualified class name of custom implementation is used by specifying property com.tibco.fom.orch.mopd.customTemplatePlanCreator.
	JMS	JMS Template Implementation is used
	AOPD	AOPD is used to generate template plan

No Template Implementation

No Template Implementation is supported by the application. The template plan generated have a plan wrapper without any plan item details. Configure the property by using the following code:

```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="AOPD" value="No" />
</ConfValue>
```

Default Template Implementation

Default Template Implementation is also supported by the application. The template plan generated have plan details with one plan item details for each order line mentioned in the order details. Configure the property by using the following code:

```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="AOPD" value="Default" />
</ConfValue>
```

Custom Template Implementation

Custom Template Implementation can be implemented by the user. The details for implementing custom template to generate plan are as follows:

- Create a new java project in IDE with omsCommon.jar in build path.

- Implement an interface with name `com.tibco.aff.oms.server.jms.orch.mopd.MOPDTemplate`.
- Build the code and include the `.class/jar` file in `WEB-INF/classes(.class)` or `WEB-INF/lib(jar)` folder of `omsServer.war`.
- Configure the following property through the Configurator:


```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="AOPD" value="Custom" />
</ConfValue>
```
- Configure the implemented class name through Configurator:


```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
```
- Redeploy the Order Management Server.
- A sample do-nothing implementation is included with distribution in `omsCommon.jar` with the name `com.tibco.aff.oms.server.jms.orch.mopd.custom.CustomMOPDTemplateImpl`. This is only for testing and it does not generate a valid plan to proceed further. If enabled, the following logs are generated:


```
Generate template plan through {} for orderID {}.
```
- By default, blank value is present in the Configurator that indicates no plan generation by custom implementation.

JMS Template Implementation

JMS Template Implementation is also supported by application. The request to generate plan is published to a JMS destination and the generated plan is published back to the JMS destination. Configure the property by using the following code:

```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="AOPD" value="Jms" />
</ConfValue>
```

You can configure the JMS destination for publishing request payload to generate plan by using the following property:

```
<ConfValue description="Queue for sending orderrequest to generate template plan for MOPD" isHotDeployable="true" name="Queue for sending orderrequest to generate template plan for MOPD"
propname="com.tibco.fom.oms.orch.mopd.template.orderrequest.sender.queue"
readonly="false" sinceVersion="3.0" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
```

You can configure JMS destination for receiving the template plan payload by using the following property:

```
<ConfValue description="Queue to receive response of generated template plan for MOPD" isHotDeployable="true" name="Queue to receive response of generated template plan for MOPD"
propname="com.tibco.fom.oms.orch.mopd.template.orderresponse.response.queue"
readonly="false" sinceVersion="3.0" visibility="Basic">
  <ConfString default="" value="" />
</ConfValue>
```

The default value is always blank for the configuration.

Using Automated Order Plan Development

The plan is generated by Automated Order Plan Development. Configure the property by using the following code:

```
<ConfValue description="Template for Mopd plan generation" isHotDeployable="true"
name="Template for Mopd plan generation"
propname="com.tibco.fom.orch.mopd.templatePlan" readonly="false" sinceVersion="3.0"
visibility="Basic">
  <ConfString default="AOPD" value="AOPD" />
</ConfValue>
```

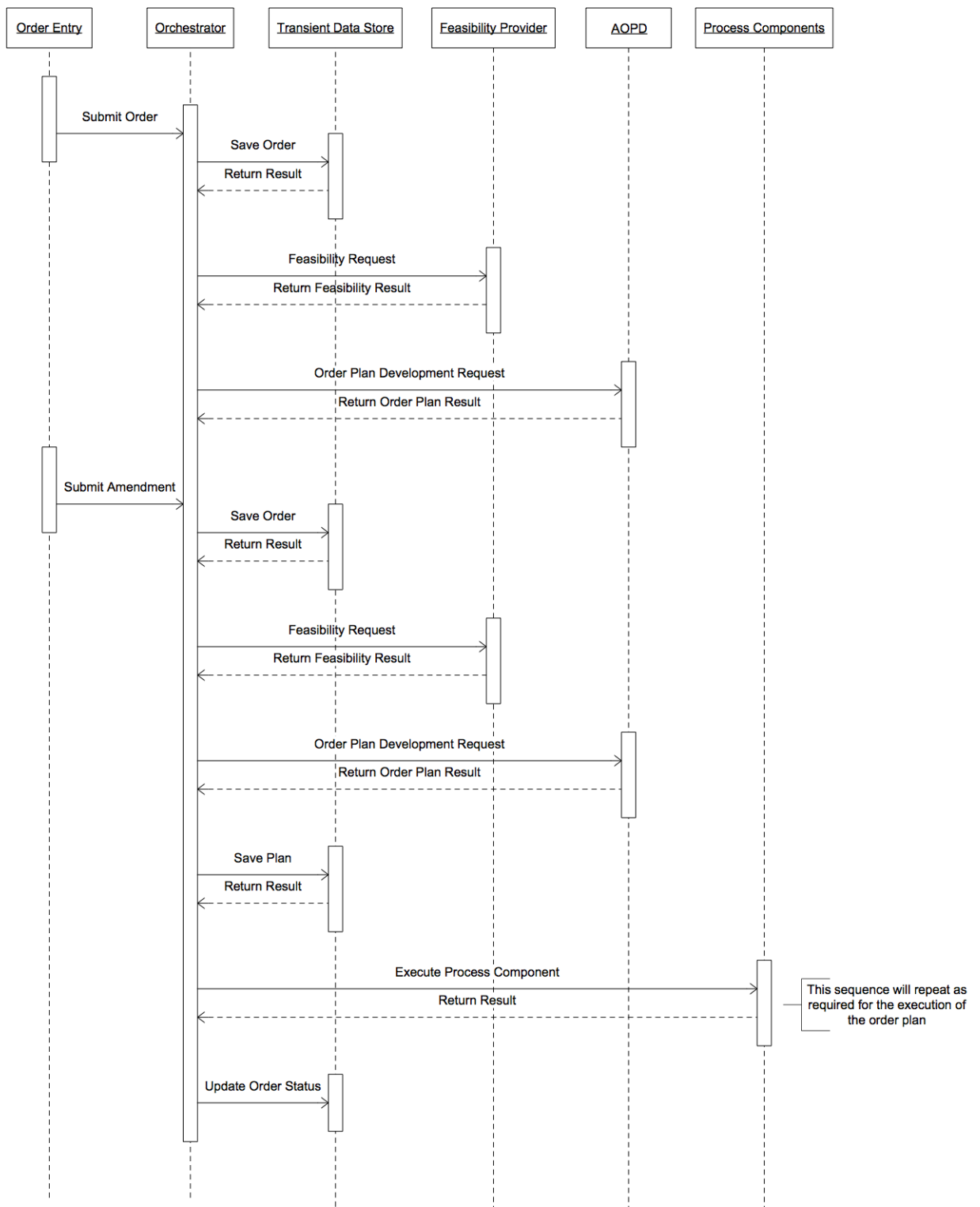
This is the default configuration to generate template plan.

Amend Order Fulfillment

Before Plan Creation

Order amendment is the process of modifying an order after submission. An amendment before plan development occurs prior to the creation of a plan within Orchestrator. The sequence is shown in the following diagram. Note that Validation has been removed for simplicity.

Amend Order Fulfillment – Before Plan Creation Sequence



In this sequence the amendment is shown as occurring after the response from Automated Order Plan Development. Note that the amendment might occur at any point before this event. The key factor is that the plan has not yet been created in Orchestrator.

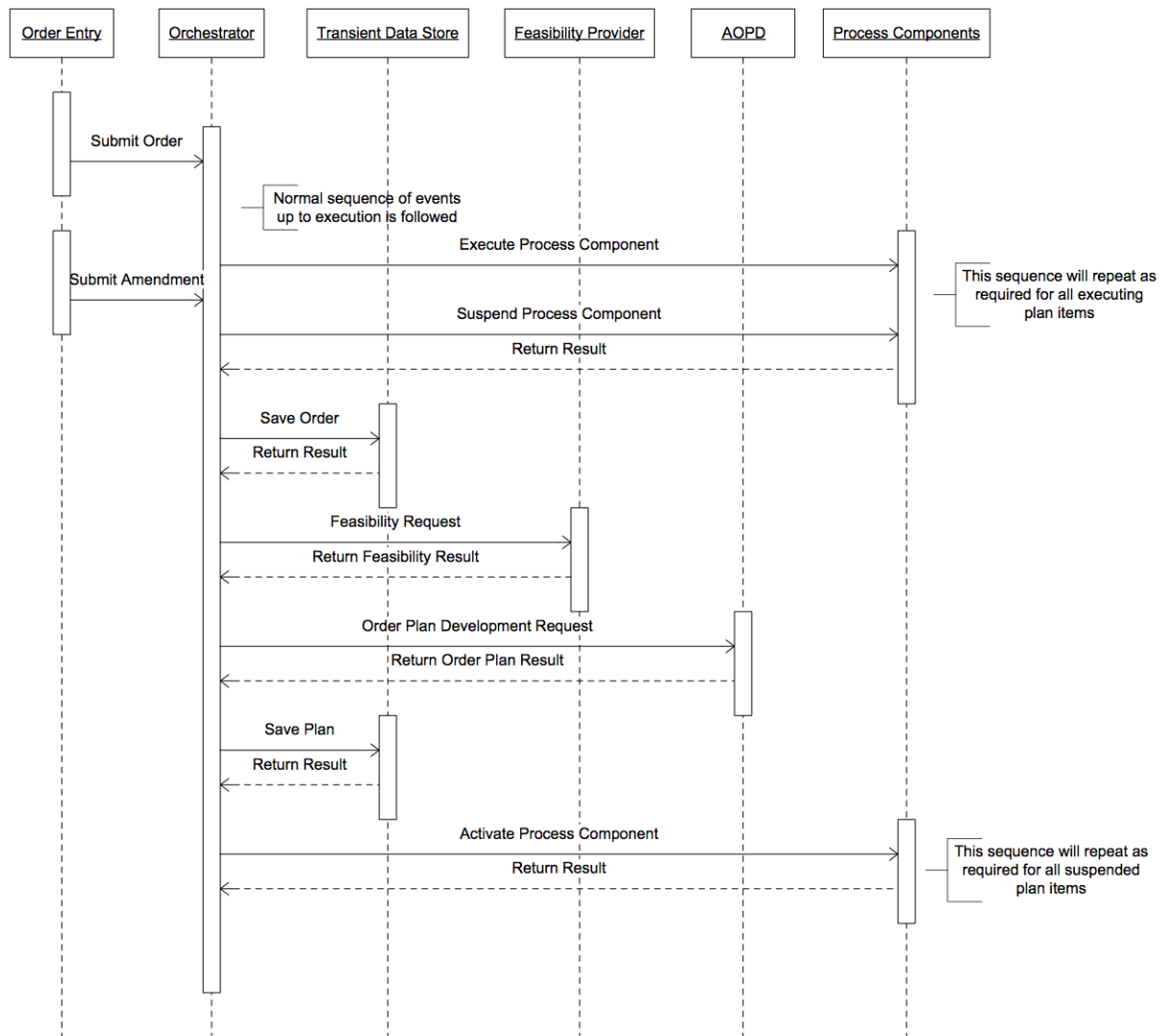
1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status
2. Orchestrator sends a request to Transient Data Store to store the order. The order now has Submitted status.
3. Transient Data Store saves the order and returns a response to Orchestrator. The order now has Feasibility status.
4. Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order.
5. Feasibility Provider aggregates all order feasibility checks and concludes that the order is feasible and sends a response back to Orchestrator. The order status is now Plan Development.
6. Orchestrator sends a request to Automated Order Plan Development to analyze the order and design an execution plan.
7. Automated Order Plan Development sends a response back to Orchestrator with the execution plan definition. Orchestrator then generates a plan based on this definition. The order now has Execution status and the plan now has Start status.
8. The order amendment is submitted from Order Entry through the Orchestrator Submit Order interface.
9. Orchestrator sends a request to Transient Data Store to store the amended order. The order now has Submitted status.
10. Transient Data Store saves the amended order and returns a response to Orchestrator. The order now has Feasibility status.

Processing now continues as in the normal Standard Order Fulfillment case.

Amend Order Fulfillment

Order amendment is the process of modifying an order after submission. An amendment after plan development occurs after the creation of a plan within Orchestrator. Note that Validation has been removed for simplicity.

Amend Order Fulfillment – After Plan Creation Sequence



In this sequence the amendment is shown as occurring during execution.

1. The order is submitted from Order Entry through the Orchestrator Submit Order interface. Note that this might use intermediate service layers. The order now has Start status.
2. The sequence of events follows the Standard Order Fulfillment sequence until the order amendment is submitted.
3. The order amendment is submitted from Order Entry through the Orchestrator Submit Order interface.
4. Orchestrator sends a suspend request to all executing plan items. The plan and order status are now both Suspended.
5. Each executing plan item returns a response indicating it suspended successfully or completed.
6. Orchestrator sends a request to Transient Data Store to store the amended order. The order amendment now has Submitted status. The order and plan are still in Suspended status.
7. Transient Data Store saves the order amendment and returns a response to Orchestrator. The order amendment now has Feasibility status. The order and plan are still in Suspended status.
8. If feasibility is enabled, Orchestrator sends a request to the Feasibility Provider to perform feasibility checking on the order amendment.

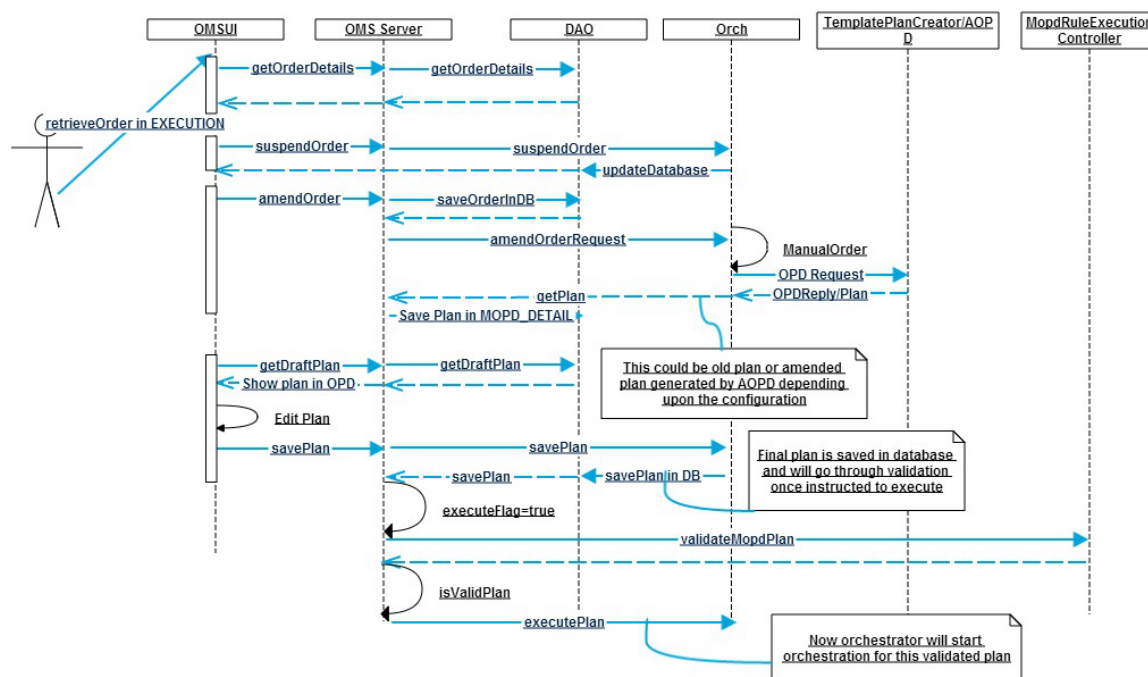
9. Feasibility Provider aggregates all order amendment feasibility checks and concludes that the order amendment is feasible and sends a response back to Orchestrator. The order amendment status is now Plan Development. The order and plan are still in Suspended status.
10. Orchestrator sends a request to Automated Order Plan Development to analyze the order amendment and design an amended execution plan.
11. Automated Order Plan Development sends a response back to Orchestrator with the amended execution plan definition. Orchestrator then updates the plan based on this definition. The order amendment status is now Amending. The order and plan are still in Suspended status.
12. Orchestrator sends a request to Transient Data Store to store the updated plan.
13. Transient Data Store saves the updated plan and returns a response to Orchestrator. The order amendment status is now complete. The order and plan are still in Suspended status.
14. Orchestrator changes the order and plan status to Execution. Orchestrator then sends an activate request to all suspended plan items. The activate message instructs the Process Component to do one of the following:
 - a. Resume execution from the point of suspension
 - b. Cancel execution but do not rollback any previously completed tasks
 - c. Cancel execution and rollback any previously completed tasks
15. Process Components return a response to Orchestrator for each activation.
16. Orchestrator continues invoking Process Components in the correct sequence. This is repeated for each plan item.

Processing now continues as in the normal Standard Order Fulfillment case.

Processing Amended Order with Manual Order Plan Development Enabled

The sequence diagram for processing an amended order with Manual Order Plan Development enables is as follows:

Sequence Diagram for Processing Amended Order with Manual Order Plan Development Enabled



The following steps provide a high level flow for manual order submission in case of amendment:

1. Identify an order which needs to be amended.
2. Suspend the identified order. Order is suspended and database is updated.
3. Submit an amendment request.
4. Orchestrator sends the amend order Request to Automated Order Plan Development, JMS, or third party for draft plan generation.
5. Orchestrator receives the draft plan and stops the further execution until the final plan is not received.
6. Search for the manual order which needs to be edited manually.
7. Retrieve the order details for manual order and gets order details in Order Management Server UI.
8. Traverse to the draft plan which was saved earlier.
9. Draft plan in UI is visible.
10. Provide Instructions to get draft plan in UI. This provides an indication to the system that you want to edit the plan.
11. Edit the plan by adding, modifying, or deleting plan item, milestones, and dependencies in the plan developed by Automated Order Plan Development.
12. Edit the draft plan through the TIBCO Order Management - Long Running UI and provide instructions to save the plan in database.
13. Plan is saved in database.
14. Provide instructions to execute this plan.
15. Plan is validated by using the validation framework provided by the server.
16. If plan is valid then the orchestrator is notified with the final plan and it starts execution of the plan.
17. If plan is not valid then application returns back specifying that plan is not valid and the plan can be corrected.

For amended order, a template plan is generated, which you can edit from the TIBCO Order Management - Long Running UI. This template plan can be generated in following ways:

1. Using the existing plan
2. New plan generated by the Automated Order Plan Development

The following property provides an indication whether the plan is generated by using Automated Order Plan Development or not:

```
<ConfValue description="Template for Mopd amendment plan generation"
isHotDeployable="true" name="Template for Mopd amendment plan generation"
propname="com.tibco.fom.orch.mopd.amendment.templatePlan" readonly="false"
sinceVersion="3.0" visibility="Basic">
  <ConfString default="AOPD" value="AOPD" />
</ConfValue>
```

A property with value FALSE indicates that the existing plan is not used and a plan is generated by using Automated Order Plan Development for amended order. A property with value TRUE indicates that existing plan is used as template plan for editing plan by user.

Property Name	Value	Purpose
com.tibco.fom.orch.mopd.amendment.templatePlan	AOPD	Template plan is generated by using Automated Order Plan Development for editing by user
	existing Plan	Use existing old plan as template plan for this order

Cancel Order

Cancel Order Fulfillment is a special case of Amend Order Fulfillment. The same sequence of events is followed, except at the end of the process the overall order and plan status goes to Cancelled rather than Completed.

Architecture

TIBCO Order Management - Long Running is made of several components. Each component has a particular role.

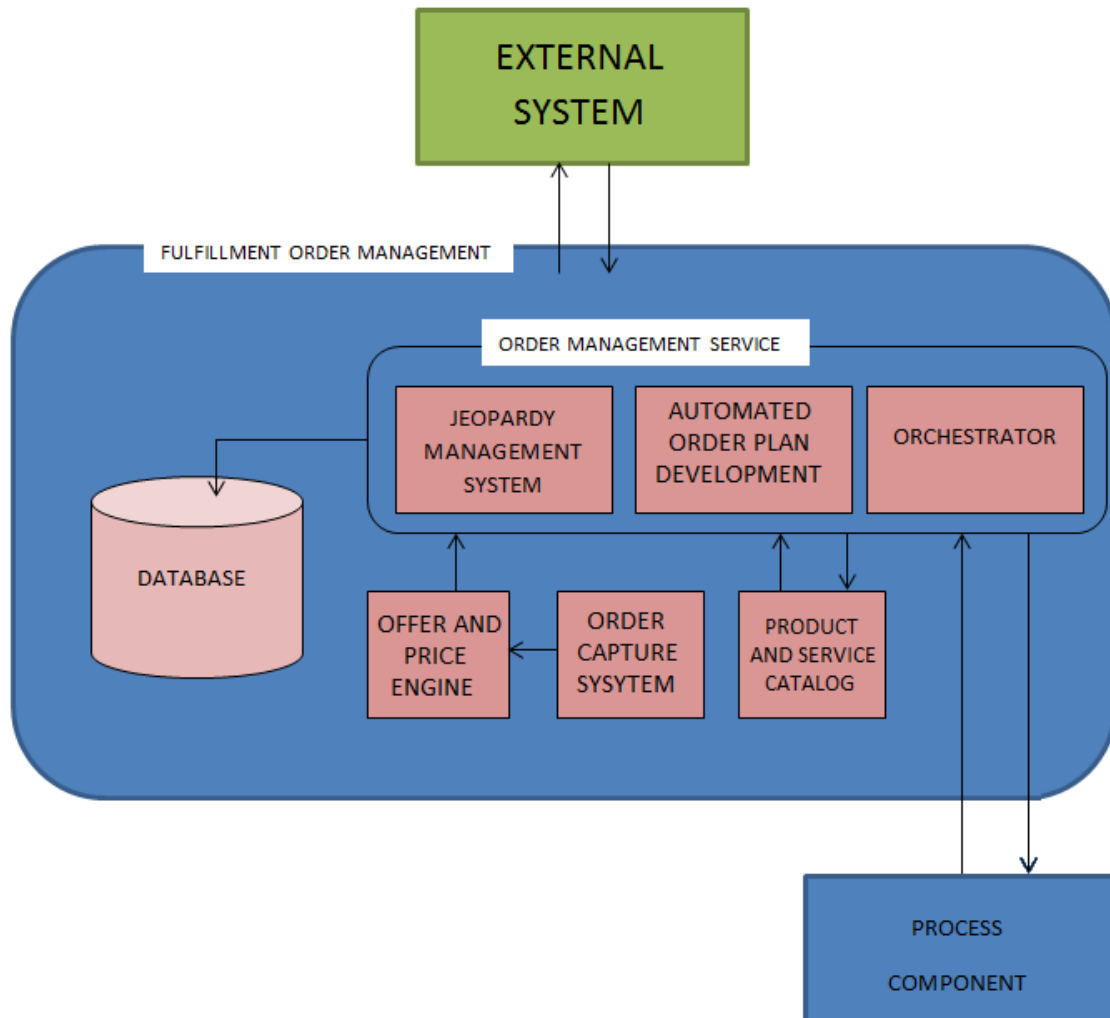
The major components of TIBCO Order Management - Long Running include:

- **Order Manager Server (OMS):** Order Management Server exposes REST HTTP that can be used by external systems to submit orders to the TIBCO Order Management - Long Running.
- **Orchestrator:** Orchestrator takes the order plan from Automatic Order Plan Development and executes those tasks till completion. It invokes micro-level process plan fragments to initiate tasks within the operator's operations ecosystem, enabling appropriate actions in a variety of back-end systems (for example, billing systems, network systems, scheduling systems, and so on). Orchestrator keeps track of status and manages exceptions.
- **Automatic Order Plan Development (AOPD):** Valid orders accepted by the Order Management Server system are decomposed into their individual products, services, and resources. An optimized order plan workflow process is then generated based on those basic building blocks to ensure an accurate order fulfillment. Optimization can take into account both product rules and customer inventory and other data to arrive at the final order plan.
- **Jeopardy Manager System (JeOMS) :** The Jeopardy Management System is a key component of TIBCO Order Management - Long Running. 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, the system notifies the stakeholders as early as possible to take corrective steps.
- **Order Capture System (OCS):** The Order Capture System is a web application component with a UI to create, manage, and submit TIBCO Fulfillment Orchestration Suite orders based on what a subscriber already has. With Order Capture System, you can select subscribers and browse validated products, services, or bundles from the TIBCO Product and Service Catalog. Order Capture System interacts with the runtime model server, which handles data, synchronizes it with all systems, and monitors the life cycle of the shopping cart.

There are additional components, which are explained in more detail in *TIBCO Order Management - Long Running User's Guide*. Here are few of those additional components:

- **Order Management Server User Interface (OMSUI):** Provides operators a GUI to manage and track orders. Order Management System persists order data and allows operators to search, view, track, and trace orders. It allows users to take actions on order/order lines.
- **Router:** Orders from Order Management Server might optionally be routed to an alternate Orchestration engine. In this case, TIBCO Order Management - Long Running Router can read inbound orders and, based on rules, might route to an external orchestration engine.
- **Common Logging:** All TIBCO Order Management - Long Running components report to a common logging component for the ease of maintenance and operations management of the system.

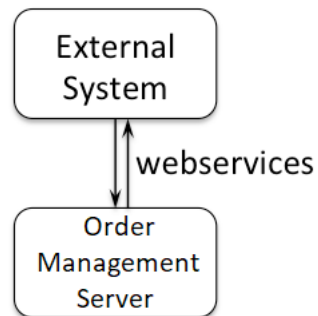
TIBCO Order Management - Long Running Architecture



Order Management Server

Order Management Server provides an input interface for external systems.

Order Management Server



Order Management Server exposes Web service that can be used by external systems to submit orders to the TIBCO Order Management - Long Running.

Order Management Server hosts the following web services:

1. SubmitOrder
2. AmendOrder
3. CancelOrder
4. GetOrderDetails
5. GetOrders
6. GetOrderExecutionPlan
7. PerformBulkOrderAction
8. SyncSubmitOrder
9. WithdrawOrder
10. SuspendOrder
11. ActivateOrder
12. GetEnrichedExecutionPlan
13. FetchAuthenticationToken
14. HandlePlanItemInError

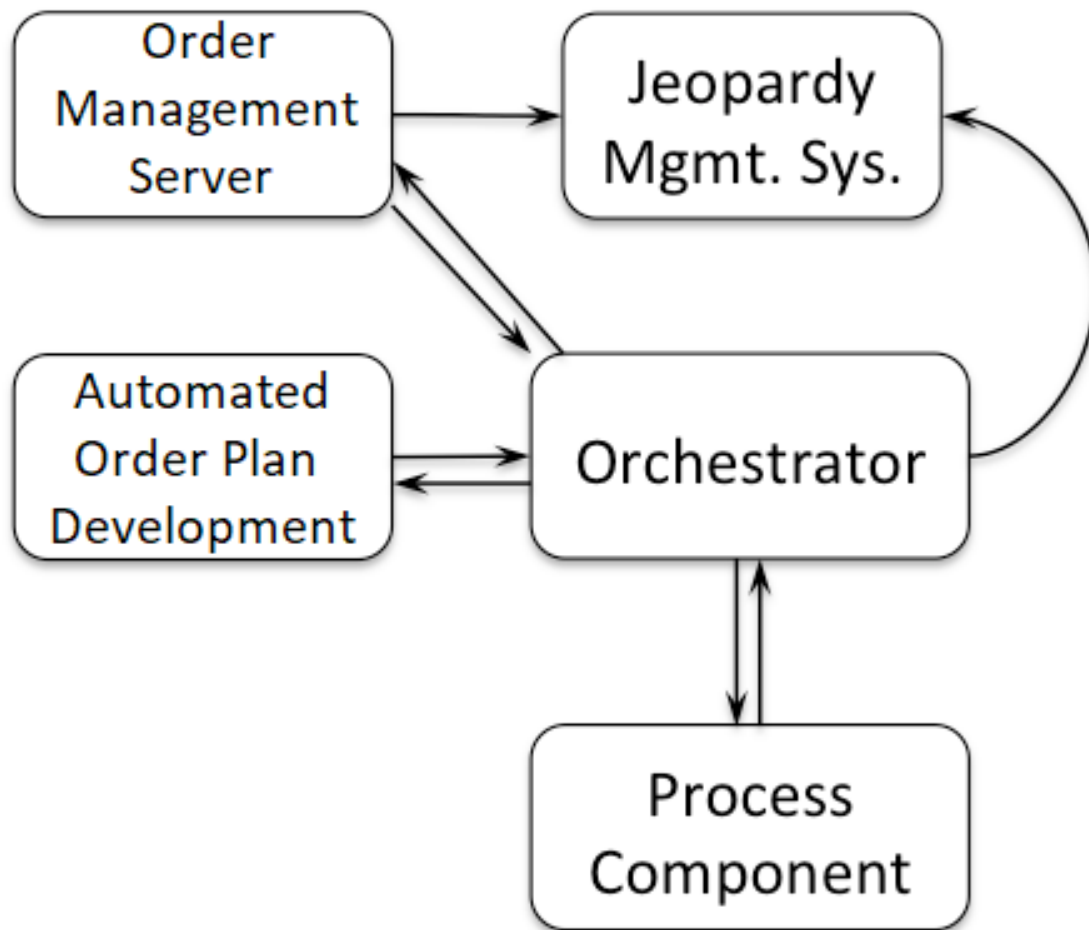
See the *TIBCO Order Management - Long Running Web Services Guide* for more details on the Web Services definitions.

Orchestrator

Orchestrator receives orders from Order Management Server. Then Orchestrator executes a series of tasks in a defined order.

Orchestrator interacts with several other components to store orders, to create plan specifications. Orchestrator is also responsible to communicate with external systems (process components).

Orchestrator



Orchestrator is responsible for the following:

1. Manage the overall order lifecycle.
2. Store the order in cache.
3. Optionally determine order feasibility by sending the order to a Feasibility Provider that returns the result back to Orchestrator.
4. Develop a plan for fulfilling the order by sending the order to Automated Order Plan Development that returns the plan specification.
5. Use this plan specification to create the execution plan for an order.
6. Store the plan in cache.
7. Interpret the execution plan and coordinate order fulfillment by invoking the correct Process Components in the correct order.
8. Update the order status to complete in the cache at the end of the order lifecycle.

When Order Management Server sends an *order*, submitted by an external system, to the Orchestrator, the order might refer to several *products*.

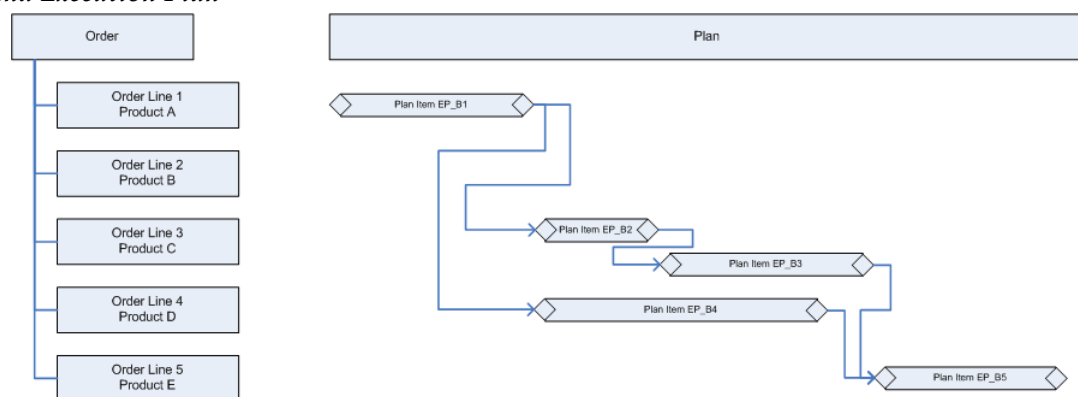
For each ordered *product*, a series of *plan items* must be completed sequentially for that product to be provided. The Product Catalog maintains the link between *product* and *plan item*. Orchestrator receives the requests for order fulfillment. Orchestrator component in turn sends the order to Automated Order Plan Development to analyze the order and the Product Catalog, and determines the *plan of action* to fulfill the order. Orchestrator then uses this plan to reach the goal by invoking the process component associated with

each *plan item* in the defined sequence to fulfill an order. For details, see *TIBCO Order Management - Long Running Administration Guide*.

The actual fulfillment of the product happens by invoking the process component - typically implemented as Fulfillment Provisioning cartridges or the BPM workflow processes - described by the plan fragment assigned to the product in the Product Catalog. The invocation of the process components in a specific sequence and at specific time is known as the *order orchestration*, which is done by the Orchestrator.

Management Orchestrator receives Automated Order Plan Development-generated execution plan for order orchestration. It has one to many inter-dependent plan items, which typically maps to the order lines in the order. See figure [Order and Execution Plan](#).

Order and Execution Plan



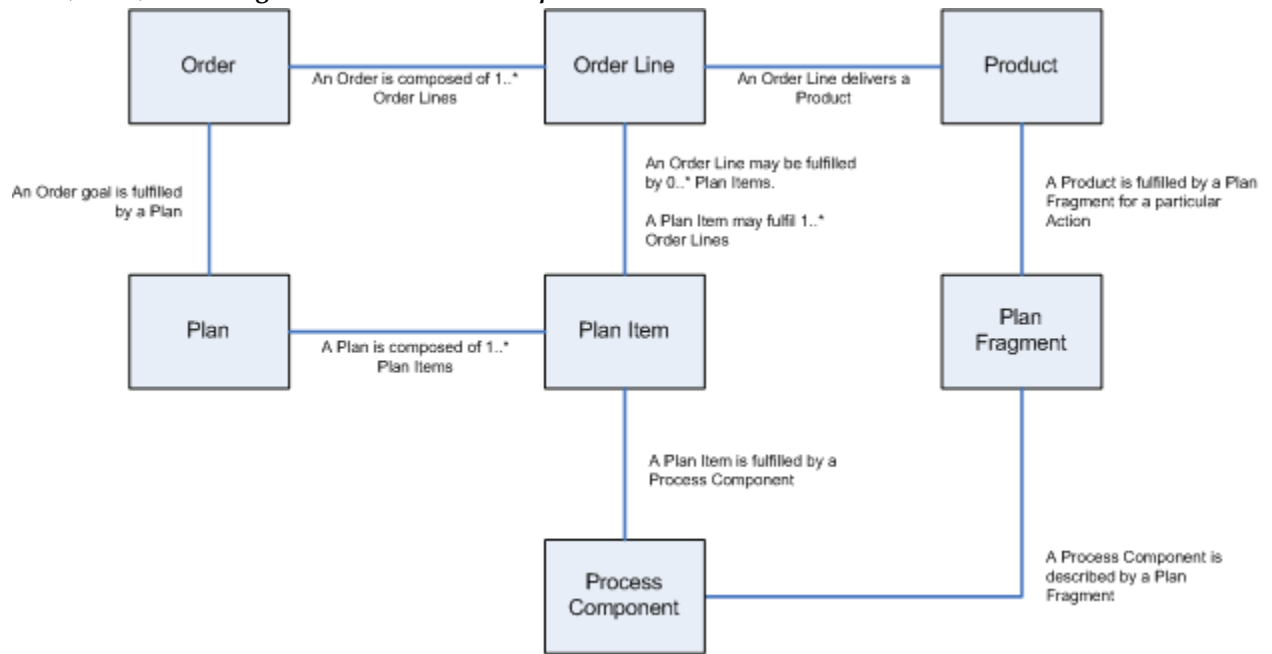
There can be one-to-one, one-to-many, or many-to-one relationships between the order lines and the plan items based on the Product Catalogue modeling.

- In case of Affinity between two products, the two order lines requesting these two products have a single plan item in the execution plan, to fulfill the product products simultaneously.
- In case of a bundled product comprising of sub-products and services, the order line requesting this product have multiple plan items in the execution plan, one corresponding to each sub-product or service.

A plan item contains the process component, which has to be invoked for the fulfillment of a particular product. Order management Orchestrator invokes the process components and starts executing the plan contained in the plan items according to the dependencies between them. The execution plan, and hence the order is considered to be COMPLETE or FULFILLED once all the process components corresponding to the plan items are executed successfully.

The high level relationships between the order and plan entities are shown in the following figure:

Order, Plan, Plan Fragment and Process Component



Plan item, Plan fragment, and Process Component are inter-related concepts.

Here is the brief description of each of these concepts:

- *Plan Item* is one of the steps in a plan that must be executed to reach the goal of fulfilling an order line, and eventually the order. The plan item is configured with the name of the Process Component, which must be invoked to fulfill a product. The name of the Process Component is provided by TIBCO Order Management - Long Running Automated Order Plan Development during plan development, and gathered from the Product Catalogue by using the name of the product.
- *Plan Fragment* is the model definition of a Process Component, which fulfills a particular product. Products are linked to plan fragments in the Product Catalogue. The name of a plan fragment is the same as the name of the Process Component that it describes.
- *Process Component* is the physical implementation of the tasks required to fulfill a product. It is described by a plan fragment and invoked as a plan item step in a plan.

Automated Order Plan Development

In TIBCO Order Management - Long Running context, an order and the corresponding execution plan respectively represent the following:

- What (goal) to fulfill/achieve, and
- How to fulfill/achieve that particular goal.

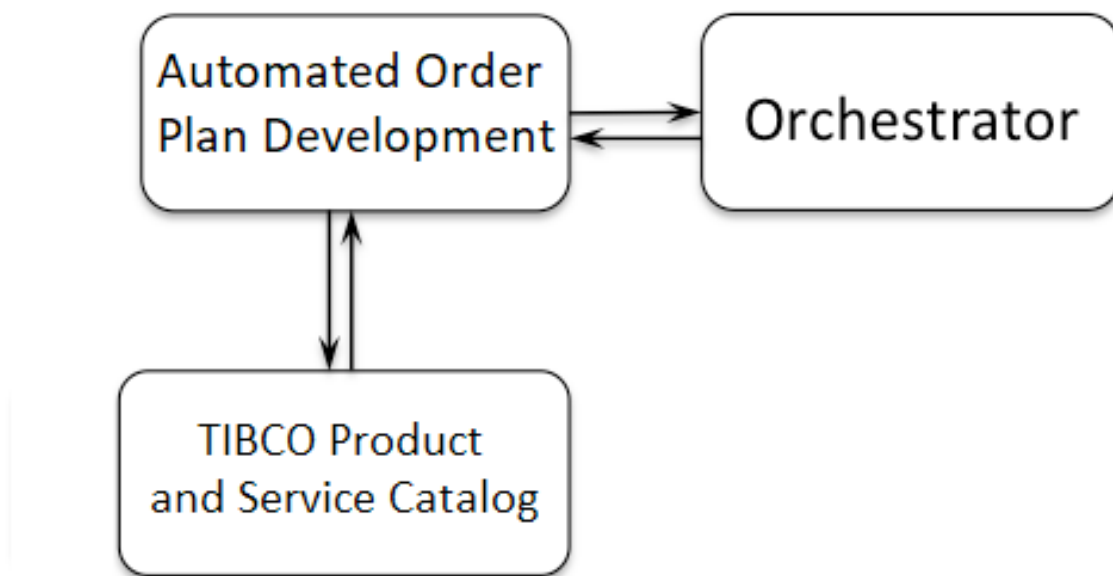
Automated Order Plan Development is the core component of TIBCO Order Management - Long Running, which transforms the What part, i.e. *order* into How - the execution *plan*.

Automated Order Plan Development receives *orders* from Orchestrator. Automated Order Plan Development decomposes an *order* into a *plan*. The *plan* is used to fulfill the corresponding *order*.

Automated Order Plan Development takes into account the specifications of the required *products* and the *products* currently provided to a customer.

Automated Order Plan Development uses a *Product Catalog* to decompose the *orders*. Typically, the *Product Catalog* can be TIBCO Product and Service Catalog.

Automated Order Plan Development



When an *order* is received, its *order lines* are decomposed by using a *Product model*.

A *Product Model* contains Bundles and Products Services. A Product model also contains concepts such as sequencing and dependencies.

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*.

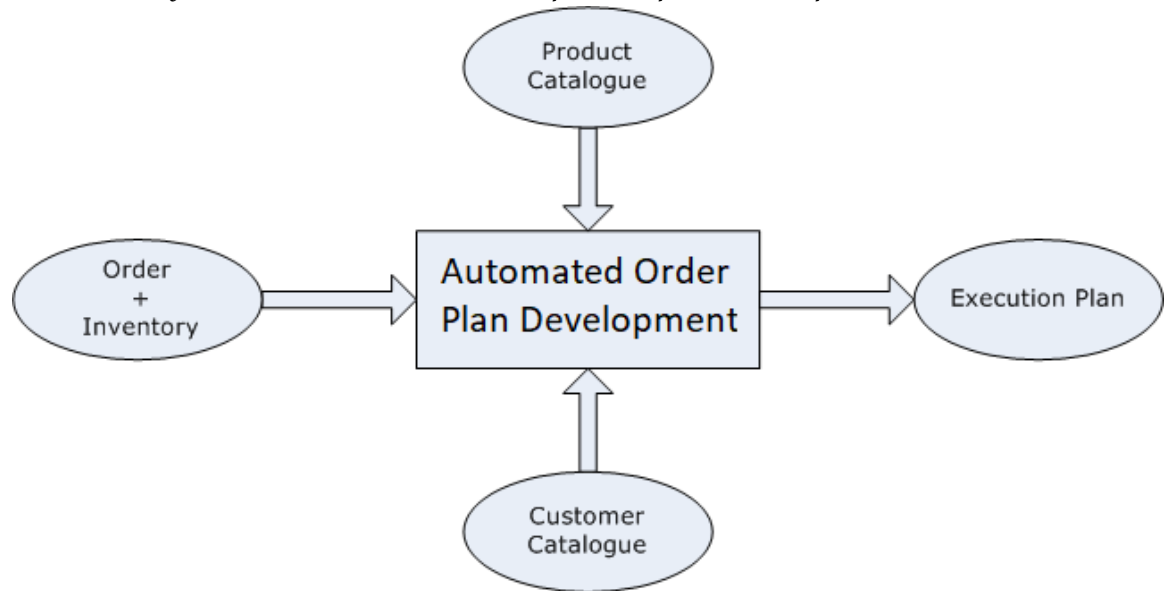
An incoming order to TIBCO Order Management - Long Running consists of one to many order lines, with each line requesting a product or service to be fulfilled. Orchestrator sends the order received from the Order Management Server component to Automated Order Plan Development for the execution plan generation. Automated Order Plan Development component has the active reference of the product and the customer catalog.

Automated Order Plan Development generates the execution plan by applying rules on the incoming order against the product, customer catalogs and the optional inventory for the customer coming along with the order in execution plan generation request. See figure [Plan Generation by Automated Order Plan Development Inputs and Outputs](#)

1. Plan development performance is related to the size of the catalog and order being decomposed. Where possible, the size of both must be reduced to improve performance.
2. Plan Fragments must not be modeled that do not do anything at execution time. Only plan items that do useful work must go into the plan.
3. Milestones and overlapping sequencing must be used instead of empty plan items for dependencies.

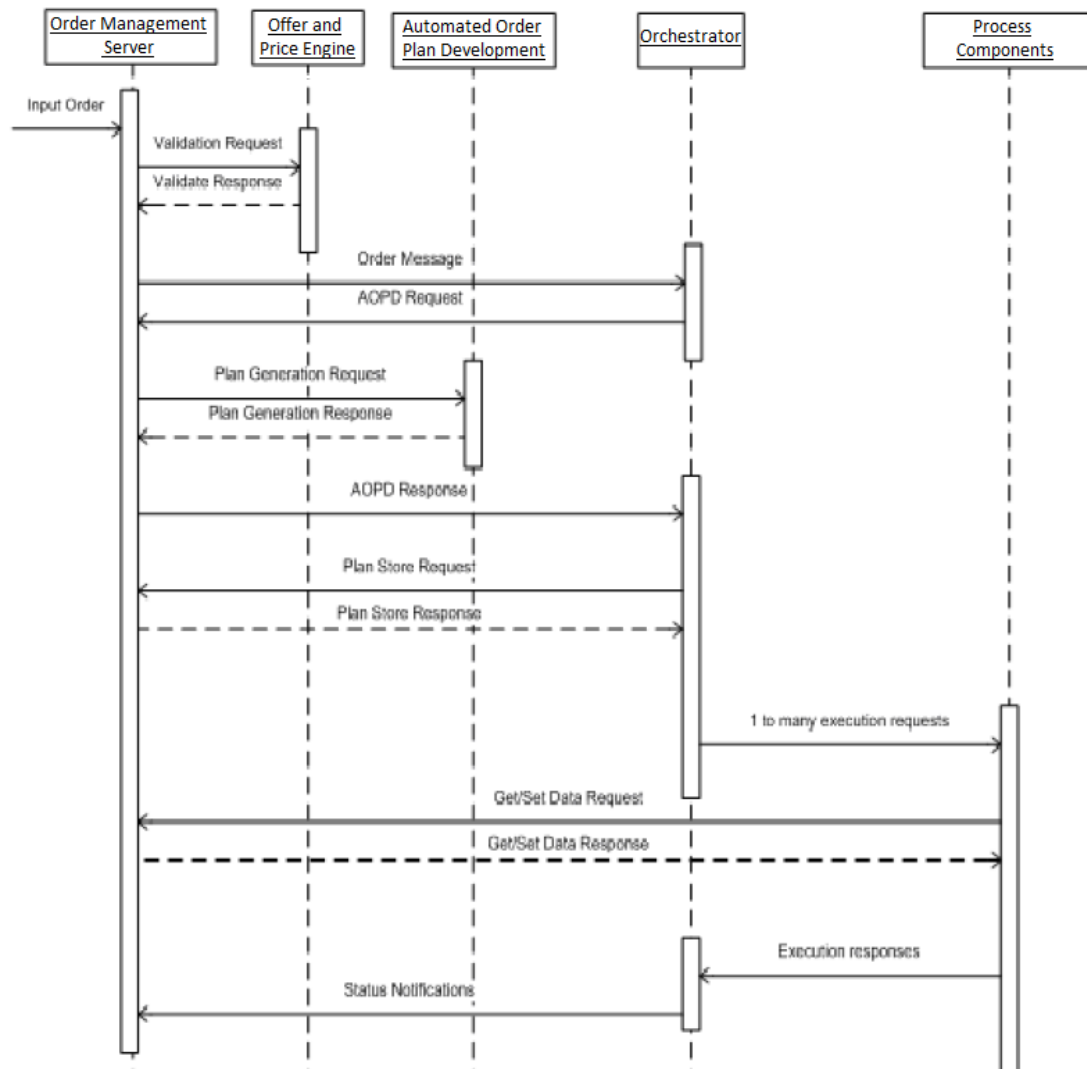
For a product requested in an order line, Automated Order Plan Development creates a plan item and assigns the plan fragment corresponding to the action specified in the order line from the Product Catalog. All such plan items are added into the execution plan. The plan is further optimized by applying rules for features such as Affinity and Single Use. On completion, the generated execution plan is sent back to the TIBCO Order Management - Long Running Orchestrator for the order orchestration process.

Plan Generation by Automated Order Plan Development Inputs and Outputs



The typical order fulfillment flow in TIBCO Order Management - Long Running is represented by the following sequence diagram:

Plan Generation and Execution Sequence



Error Messages and Handling

Automated Order Plan Development provides error handling and returns meaningful responses in case of errors detected during the stages of plan development. During plan generation if any errors are reported, Automated Order Plan Development stops the plan generation immediately and returns an error response. The error handling also takes care of circular dependency in the plan and returns an appropriate response.

Manual Order Plan Development

Manual Order Plan Development (MOPD) gives users the control to modify the Automated Order Plan Development developed plan. In Manual Order Plan Development, plan development happens through TIBCO Order Management - Long Running User Interface under predefined logical boundaries.

1. Order is submitted by using JMS or HTTP.
2. Order Management Server picks up the order request and saves the record into the database.
3. Router Component of Order Management Server routes the order request to Orchestrator by using JMS or can invoke the orchestrator service directly (in Process).
4. Orchestrator sends the order Request to Automated Order Plan Development/JMS/third party for draft plan generation.

5. Orchestrator receives the draft plan and halts the further execution until final plan is not received.
6. User searches for the manual order which is supposed to be edited manually.
7. User retrieves the order details for manual order and gets order details in Order Management Server UI.
8. User traverse to the draft plan (showing Manual Order Plan Development plan similar to execution Plan) which is saved earlier.
9. User can see the draft plan in user interface.
10. User instructs to get draft plan in UI. This indicates that user wants to edit the plan now.
11. User can edit the plan now by adding/modifying/deleting Plan-Item, Milestones and Dependencies in the plan developed by Automated Order Plan Development.
12. User edits the draft plan through the TIBCO Order Management - Long Running UI and instructs to save the plan in database.
13. Plan is saved in database.
14. User instructs to execute this plan.
15. Plan is validated by using the validation framework provided by server.
16. If plan is valid then Orchestrator is notified with the final plan and it starts execution of the plan.
17. If plan is not valid then application returns back specifying that plan is not valid and plan can be corrected.

Server and Client Side Validations

A plan manually modified after being instructed to go into execution by the user does not go for Automated Order Plan Development rule validation. Hence, it is user's responsibility to modify the plan correctly. Some of the plan level validation by default happen at the client side which is covered in the *TIBCO Order Management - Long Running Administration* guide.

Order Capture System

Order Capture System (OCS) is a new component in TIBCO Fulfillment Orchestration Suite to create, manage, and submit TIBCO Fulfillment Orchestration orders.

This component is a web application which you can use to do the following:

- Select subscribers
- Browse validated products, services, or bundles from TIBCO Product and Service Catalog
- Create orders for selected subscribers and submit them to TIBCO Order Management - Long Running

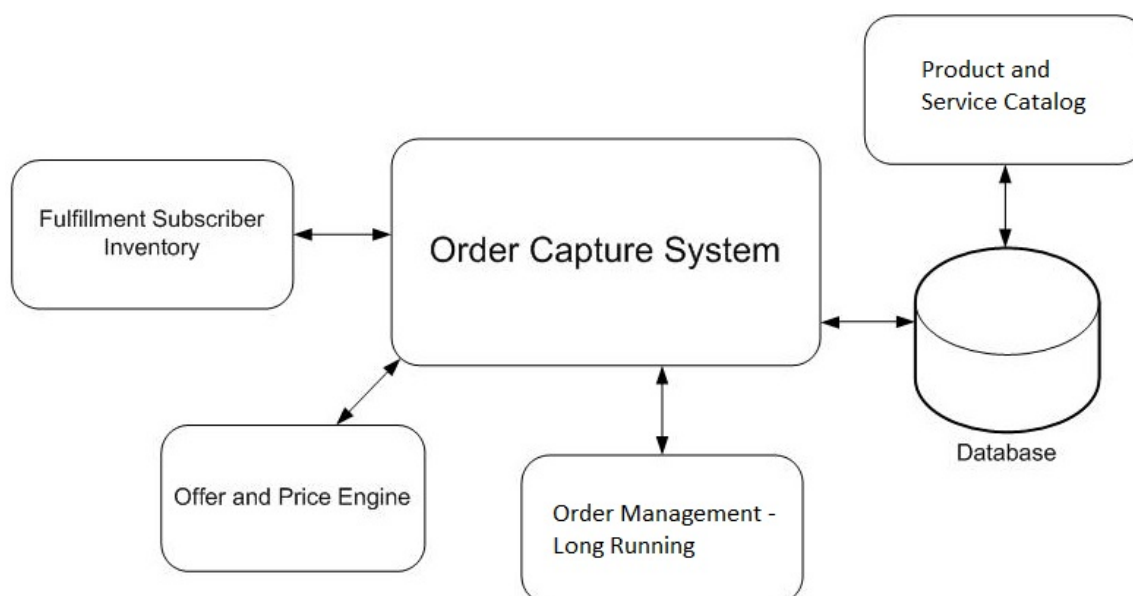
To construct an order, Order Capture System requests information from the following systems:

- TIBCO Product and Service Catalog, which provides product definitions for subscribers
- A subscriber inventory, which provides subscriber details (such as names, addresses, IDs, and segments)
- A subscriber item 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 - Long Running.

The following figure illustrates the interconnection between Order Capture System and the Fulfillment Orchestration sub-systems:

Order Capture System Architecture



Order Capture System is an optional component that is a part of Fulfillment Orchestration and shipped within TIBCO Order Management - Long Running.

Order Capture System Architecture

The Order Capture System application is a web application hosted in a Tomcat server. The Tomcat hosted application accesses external systems to browse the product catalog and create, submit, and amend orders in TIBCO Order Management - Long Running.

The web application is composed of 2 parts:

- A javascript part running in the browser called Order Capture System UI, which accesses the Tomcat hosted application
- A Tomcat hosted java application responsible for accessing the storage and external systems called Order Capture System Server

Order Capture System User Interface

Order Capture System UI is a tool for browsing products, services, or bundles in the TIBCO Product and Service Catalog, creating new orders for selected subscribers, and submitting them to TIBCO Order Management - Long Running.

You can use Order Capture System UI to do the following:

- Select a subscriber.
- View subscriber details.
- Browse the catalog.
- Select items from the list of products.
- Add items (products or bundles) to the shopping cart.
- Review the shopping cart.
- Edit selected items.
- Submit an order to TIBCO Fulfillment Orchestration Suite (such as a new order, an amended order, an updated order, or a ceased order).

Order Capture System External Systems

Order Capture System is a web based application that is used to browse products and bundles, and submit, amend, or cancel orders to TIBCO Fulfillment Orchestration Suite. To construct an order, the server part of Order Capture System connects to the multiple external systems and sends them requests.

The nominal path to construct and submit an order is as follows:

1. Search for and select a subscriber.

When you search for a specific subscriber, Order Capture System connects to a subscriber inventory service to retrieve a list of subscribers matching your search criteria. You can then select a specific subscriber from this list.

2. Configure the products or bundles for the selected subscriber.

This involves three different external systems:

- TIBCO Product and Service Catalog, which provides the definitions of products and bundles
- An eligibility service, which provides the eligible products or bundles for the selected subscriber
- A pricing service, which provides the prices of the products or bundles.



Within TIBCO Order Management - Long Running, both eligibility and pricing are grouped into TIBCO Offer and Price Engine (OPE).

3. After the products and bundles are configured, you can submit the order to TIBCO Order Management - Long Running

Order Capture System uses a database to store data.

Subscriber Inventory	<p>Order Capture System uses this service to select subscribers based on a specific search string. For more information on the WSDL defining this service, see <i>TIBCO Order Management - Long Running Web Services</i> guide. This interface is not implemented in TIBCO Fulfillment Orchestration Suite. Instead, it must be implemented by projects that have access to a CRM.</p> <p>For demonstration purposes, Order Capture System provides a hosted implementation, where you can create demo subscribers and subscriber details. This implementation is on by default and can be toggled on or off from the Order Capture System configuration file. For more information, see the <i>TIBCO Order Management - Long Running Administration</i> guide.</p> <div data-bbox="673 1482 716 1530"> </div> <p>Demo Subscriber Directory is for demonstration purposes only.</p>
TIBCO Product and Service Catalog	<p>TIBCO Product and Service Catalog defines and manages life cycles of commercial and technical offerings. It also contains the metamodel for the services and products you can browse and choose from. Characteristics, such as name and type of attributes of the services and products, are configured by the catalog. Order Capture System accesses the TIBCO Product and Service Catalog segment and product information through an offline export. For more information, see <i>TIBCO Product and Service Catalog User's Guide</i>.</p>

Offer and Price	In the TIBCO Fulfillment Orchestration Suite, this interface is implemented by the TIBCO Offer and Price Engine. This engine determines which products are eligible for the subscriber, based on the current shopping cart and the subscriber's purchased products or services. This engine also determines pricing information based on the product, service IDs, and characteristics you are viewing or selecting.
TIBCO Order Management - Long Running	Order Capture System uses TIBCO Order Management - Long Running to browse products, submit new orders, and cancel or amend existing orders. Order Capture System accesses TIBCO Order Management - Long Running through the Order Management web service. The WSDL used is \$OM_HOME/schemas/wsdl/http/OrderServiceHTTP.wsdl. For more information on how to configure the web service details, see the <i>TIBCO Order Management - Long Running Administration</i> guide.
Database	Order Capture System uses a database to retain information gathered from external systems. For example, it retains characteristic values for products that a subscriber has selected until the order is submitted to TIBCO Order Management - Long Running. After the order is submitted, it is discarded from Order Capture System and the master copy is stored within TIBCO Order Management - Long Running. Therefore, the database can be considered as a temporary cache for Order Capture System. Data within this database can be safely discarded; only uncommitted work is lost.

Process Components

This component can be implemented in a variety of technologies depending on the required functionality. Typically this is TIBCO Fulfillment Provisioning.

All Process Components must adhere to the service contract specified by Orchestrator to be considered a valid Process Component. This means implementation of three types of request events and providing two types of responses. These components must be accessed via standard JMS event interface wherever possible. Individual Process Components must be stand-alone components, which allows for changing the Process Component collection dynamically in real-time without requiring an order management outage.

All external component integrations are through the Process Components component. These integrations is generally either service calls to perform automated tasks or callouts to start a manual workflow.

The integration pattern for automated service calls take the form of Process Components sending out an event to an adapter layer that includes relevant order and order line data as requested from the cache. This adapter layer then transforms the data into the format required by the back-end service and then invoke that service. When it has completed, it sends a response back to Process Components to complete the step in the flow.

Process Components are responsible for the following:

1. Implement the *tasks* required to fulfill a particular *product* on an order. This might be done in any JMS-enabled technology provided the interface specification for a Process Component is satisfied.
2. Accept requests from Orchestrator to start executing a new fulfillment process.
3. Request required information from the cache that is required as part of a fulfillment process.
4. Execute the required business process for fulfilling a particular product that a customer might order. This might take the form of invoking back-end service calls, business process management, or manual tasks as appropriate for the implementing technology.

5. Update information in the cache as part of the fulfillment process if required.
6. Return the execution results to Orchestrator.
7. Suspend execution of a fulfillment process when requested by Orchestrator. Respond to the suspend request by returning to Orchestrator confirmation of a successful suspend or normal completion of the fulfillment process.
8. Resume execution of a suspended fulfillment process when requested by Orchestrator through to completion from the point of suspension.
9. Cancel execution of a suspended fulfillment process when requested by Orchestrator. The tasks following the point of suspension are not executed. Cancellation might require rollback of previously completed tasks in the fulfillment process, or a simple abort of the execution process.

Feasibility Provider

This component can be implemented in a variety of technologies depending on the required functionality provided it meets the interface specification requirements for a Feasibility Provider.

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

It is accessed through a JMS event interface.

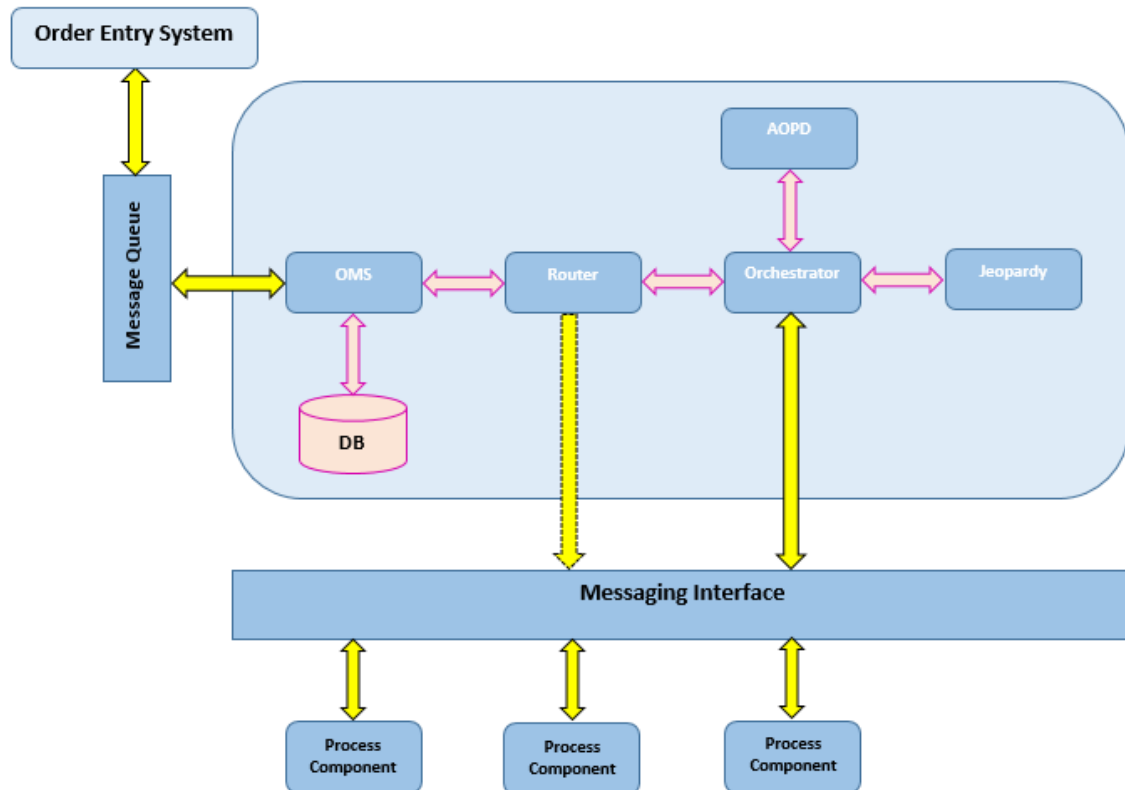
Jeopardy Management

Jeopardy is implemented as a tightly coupled component in TIBCO Order Management - Long Running along with existing features such as Order Management Server, Automated Order Plan Development, and Orchestrator. Notifications to Jeopardy is sent as low level API calls or through in-process communication, which is similar to other component communication. The advantages of Jeopardy Management System are as follows:

- Jeopardy runs in cluster-mode.
- Jeopardy notifications are processed in synchronous, or asynchronous (batch) modes.
- Improved performance of Jeopardy performance due to in-process (low level API calls) communication between this application's components.
- Jeopardy is tightly coupled with orchestrator and it follows all the finite state automata states.
- Orchestrator has complete control over the jeopardy functionality improving its performance.
- In case of any issues, the Orchestrator rolls back any updates to a plan, or plan item, and jeopardy automatically reads the latest changes.
- Instead of saving plan, plan item, and process component information as part of Jeopardy, all the information is now saved as part of state machine. Jeopardy reads and updates the information as part of the state machine itself.

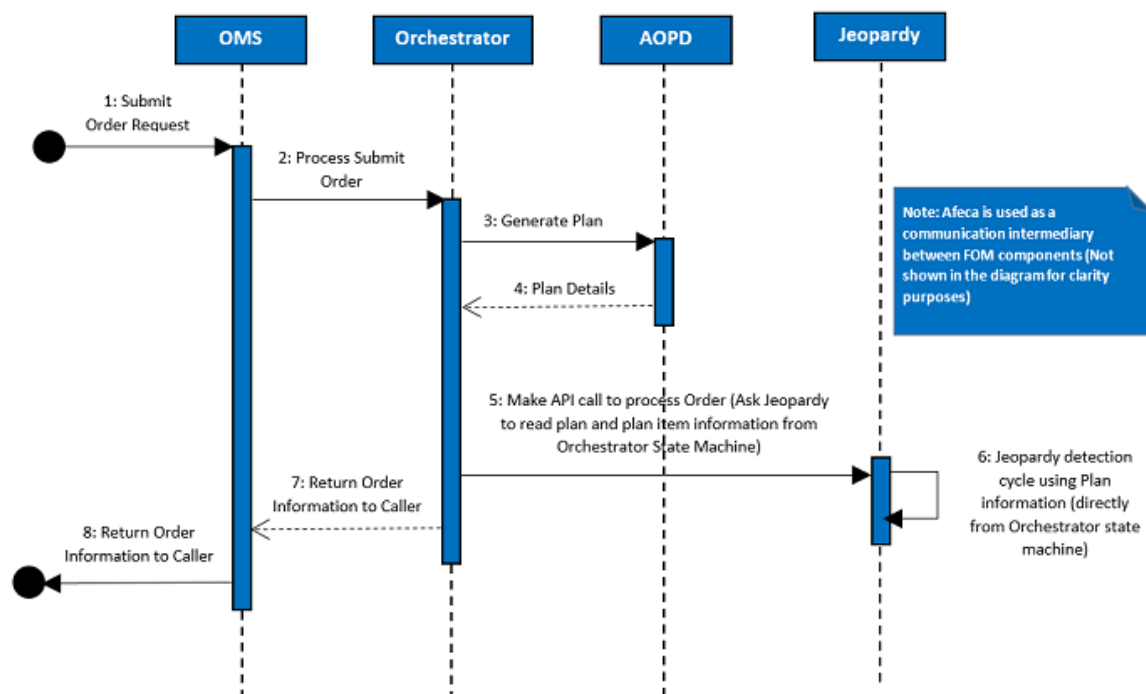
The following diagram is a representation of the architecture of Fulfillment Order Management:

Architecture of TIBCO Order Management - Long Running



The following diagram is a representation of the Plan and Order Execution by using Jeopardy:

Plan and Order Execution

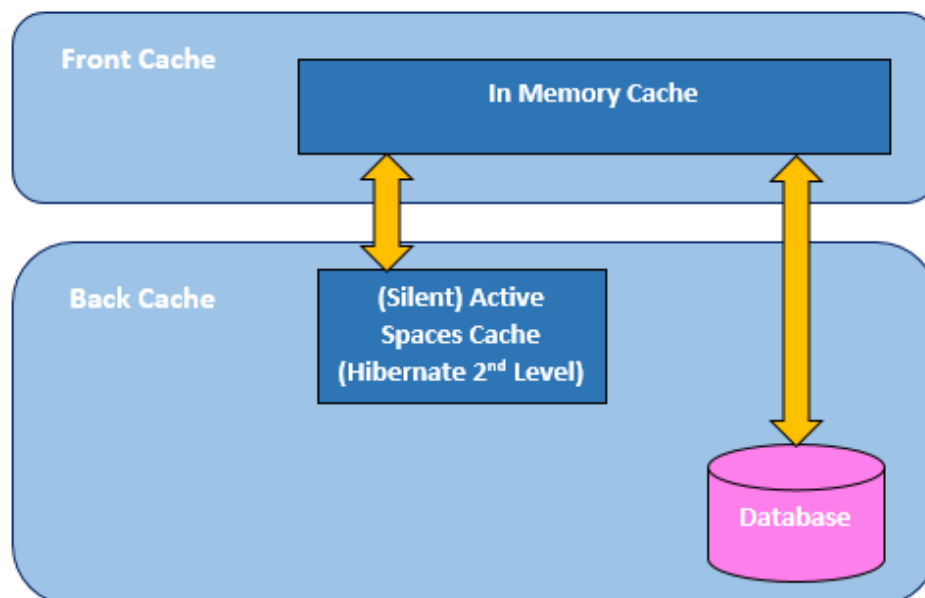


In-memory (Cache) Plan and Process Component Repository

The performance of Jeopardy has been improved by an in-memory implementation of plan and process repository. The plan repository is managed as a component in the Orchestrator finite state machine. If the

server is abruptly terminated, both the plan, and the process component repositories is backed by the database that acts as a secondary level cache to safeguard the changes.

Working of in-memory cache



The in-memory cache provides local cache access and extreme performance. It uses the database as the persistence system to persist information across server startups. The Hibernate feature continues to use active spaces as a silent second level cache for any spillovers from the main memory cache.

The application uses Orchestrator state machine's state as the primary in memory caching system. It also uses an existing Orchestrator caching and spill over mechanism to handle additional caching of data running out of memory. This practically sets no limit on the in-memory cache size.

Data store Compatibility

A new data store (secondary cache) has been added to the system. The application can now be deployed with in-memory cache along with the database as a secondary cache.

The configuration to setup the database as a data store, can be configured in *Configvalues_JEOMS.xml* file as follows:

```

<ConfValue description="Operation Data Store Type" name="Operational Data Store Type"
propname="com.tibco.jm.config.odstoreType" sinceVersion="2.0" visibility="Advanced">
  <ConfString default="cache" value="cache" />
</ConfValue>
  
```



With the collocated architectural changes, the application cannot run in standalone mode with Cache as a data store. This restriction is added to keep the orchestrator component as a master component, to control the jeopardy notification functionalities.

Database: Secondary Cache Option

Database is implemented as a second-level cache for durability purposes. The database is configured as the application second level cache in *Configvalues_JEOMS.xml* file as follows:

```

<ConfValue description="Operation Data Store Type" name="Operational Data Store Type"
propname="com.tibco.jm.config.odstoreType" sinceVersion="2.0" visibility="Advanced">
  <ConfString default="cache" value="cache" />
</ConfValue>
  
```

Cache is the default value and the only configuration supported in collocated mode.



Hibernate uses active spaces as a dialect for silently maintaining in memory spill overs.

Batch Notification and Messages

Following are the notifications sent by Jeopardy Management System:

1. JMS notification: To save jeopardy messages and risk region in the database when a jeopardy condition is detected for a plan or PlanItem.
2. JMS notification: To report an alert in dashboard when a jeopardy condition is detected for a plan or PlanItem.
3. Database notification: To update cache.

In case of existing architecture, notifications are processed and saved into the database. Processing is sequential and not in the batch. Each notification has a dedicated connection from connection pool. With the new architecture, there are options to synchronously process the JMS notifications sequentially, and also to execute the notification in the batch.

Batch Event Processing

Events are consumed by state machine and notifications generated by the state machines that are posted to the Batch process. Events update the cache repositories for any status updates, or event notifications related to jeopardy detection. The following is the sequence of activities involved:

1. State Machine receives the events from JMS.
2. Events are consumed by State machine.
3. State machine generates database notifications and JMS notifications that are posted to Batch Processes.

Router

The Content-based router in Order Management Server allows to route the order to the correct destination based on the contents of the order message.

Content-based routing routes messages are based on the actual content of the message itself, rather than by a destination specified by the message. Content-based routing works by opening a message and applying a set of rules to its content to determine the destination of a message. By freeing the sending application from the have to know anything about where an order should be routed for fulfillment, content-based routing provides a high degree of flexibility to configure multiple type of Orchestration engine. For details, see *TIBCO Order Management - Long Running Administration Guide*.

Key Functionality

TIBCO Order Management - Long Running provides the following functionalities:

- **Order Management Configurator Graphical User Interface (GUI)** - configures the settings for TIBCO Order Management - Long Running system by using GUI mode.
- **Attribute-based Decomposition** - filters execution plan depending on the orderline attributes of the products ordered.
- **Affinity** - allows different plan fragment types to be grouped together on the same order.
- **Order Amendments** - Order Amendment now handled through new Order Management Server and TIBCO Order Management - Long Running Orchestration engine.
- **Offline Catalog** - enables the TIBCO Order Management - Long Running application to fulfill the orders and reduces the dependency on the TIBCO Product and Service Catalog to be Online.
- **Centralized Logging** - provides the logging and reporting capability for analyzing the data and generate meaningful reports.

- **Dependent and Sibling Product** - enables grouping products on requests thereby allowing for sibling products and their children products to be sent to a dependent product. The ability is built in the Product Model to indicate that a product is dependent on its peer or peer's hierarchy.
- **Shared Attributes** - 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.
- **Custom Action Support** - allows you to submit custom actions.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join the 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](#) page.

- *TIBCO® Order Management - Long Running Release Notes*
- *TIBCO® Order Management - Long Running Installation and Configuration Guide*
- *TIBCO® Order Management - Long Running User's Guide*
- *TIBCO® Order Management - Long Running Administration Guide*
- *TIBCO® Order Management - Long Running Getting Started Guide*
- *TIBCO® Order Management - Long Running Best Practices Guide*
- *TIBCO® Order Management - Long Running Concepts and Architecture Guide*
- *TIBCO® Order Management - Long Running Web Services Guide*

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- 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 <https://support.tibco.com>. 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 TIBCO Software 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. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

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

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 © 2010-2022. TIBCO Software Inc. All Rights Reserved.