Oracle® Communications Order and Service Management

Concepts

Release 7.5
F60008-02
June 2024

ORACLE

Oracle Communications Order and Service Management Concepts, Release 7.5

F60008-02

Contents

9 About Runtime Order Management

10 OSM Functional Overview

Glossary

Preface

This guide provides an overview of Oracle Communications Order and Service Management (OSM).

Audience

This guide is intended for:

- Business domain experts who make decisions about the order fulfillment process.
- Order management personnel who need to know how OSM works and how orders are processed.
- Developers who extend OSM to interface with external systems.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Order and Service Management Overview

This chapter provides an overview of Oracle Communications Order and Service Management (OSM).

Overview of OSM

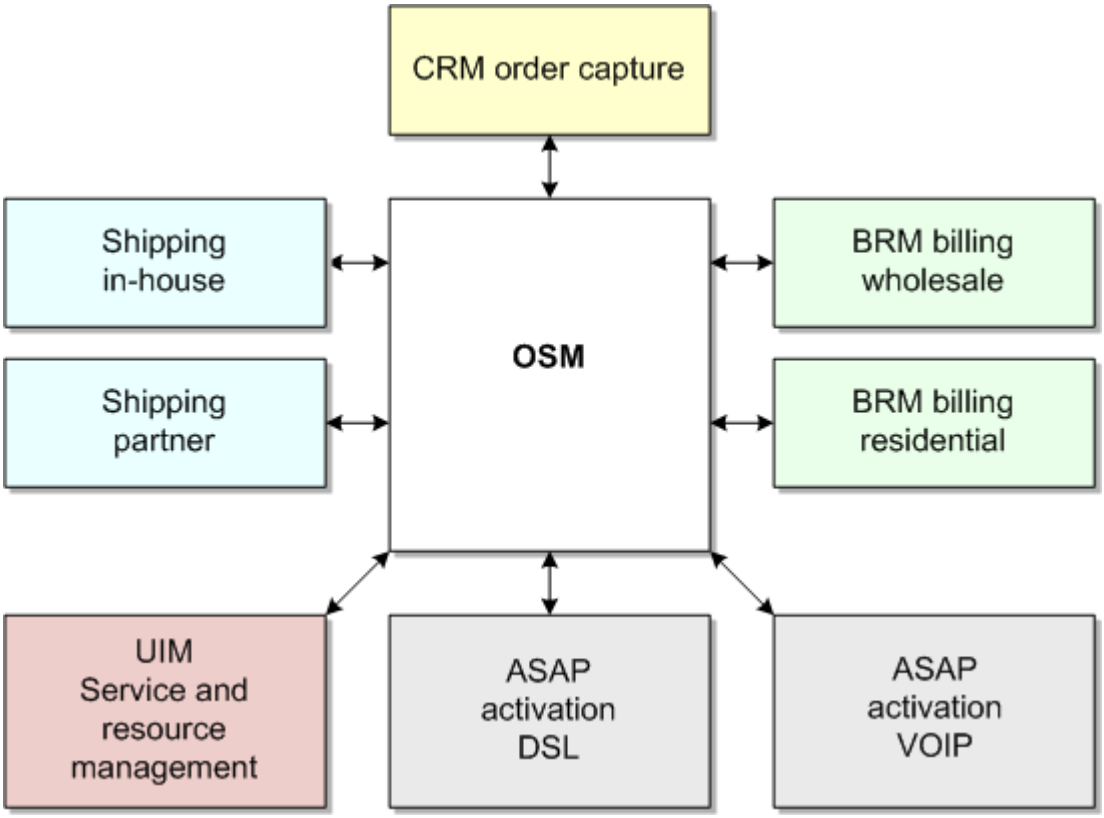
OSM is an order processing system that takes as input an order from a customer relationship management (CRM) system, and manages the fulfillment functions that need to be carried out to complete the order. **Fulfillment functions** include operations such as assigning a phone number, activating a service on the network, shipping a phone, and running billing.

Communications service providers typically register customers and manage their services by capturing orders in a CRM system or on a website. The order received from a customer defines what the customer wants to purchase, such as a phone service. OSM initiates and coordinates the order fulfillment functions required to complete the order. For example, if the customer orders a phone service, OSM can:

- Send billing and customer management requests to a billing system.
- Query a service and resource management (SRM) application to find the network resources required to activate a service and to determine the actions required on each resource; for example, to find out if there is enough circuit capacity for the service and to determine what actions must be performed on the network to allocate the circuit to the customer.
- Send activation commands to Oracle Communications ASAP to activate the service on the network.
- Send a shipping request to a shipping system; for example, to send a phone to the customer.

Figure 1-1 shows how OSM receives orders from a CRM system and then works with multiple external fulfillment systems to handle the order fulfillment requirements. A **fulfillment system** is a system that carries out the actions necessary to complete the order; for example, activate services on the network, or run billing. In this example, OSM coordinates with fulfillment systems that run other Oracle Communications applications; Billing and Revenue Management (BRM), Unified Inventory Management (UIM), and ASAP.

Figure 1-1 OSM and External Fulfillment Systems



While processing an order, OSM can interact with multiple external systems simultaneously, or in a sequence of processes. For example, OSM can interact with a billing system at the same time as working with UIM to assign the resources and determine the actions required to fulfill the service on those resources. After determining the resources and actions in UIM, OSM can send those resources and actions as activation commands to ASAP to activate the service on the network. OSM maintains the status of all of the interactions with external systems, and can return the status of each interaction to the order source system.

Because OSM can support order processing for any type of service or product, OSM does not have a predefined order fulfillment process. Instead, you define the OSM order fulfillment process for each type of order that you process. For example, some orders might require shipping or installation actions, and some might not.

The high-level order fulfillment process is:

1. An order is placed in a CRM system or on a self-service portal such as a page. OSM is not part of the order capture process.

   CRM systems typically create and maintain their own order that tracks the status of the customer's purchase. They use this order to communicate the order status to the customer.

2. The CRM system sends the order to OSM.
3. OSM receives the sales order. OSM validates the order and transforms it into an OSM order in the OSM order format. If you have defined multiple types of orders, for example, for different services, OSM creates the type of order that is required to fulfill the customer's order.

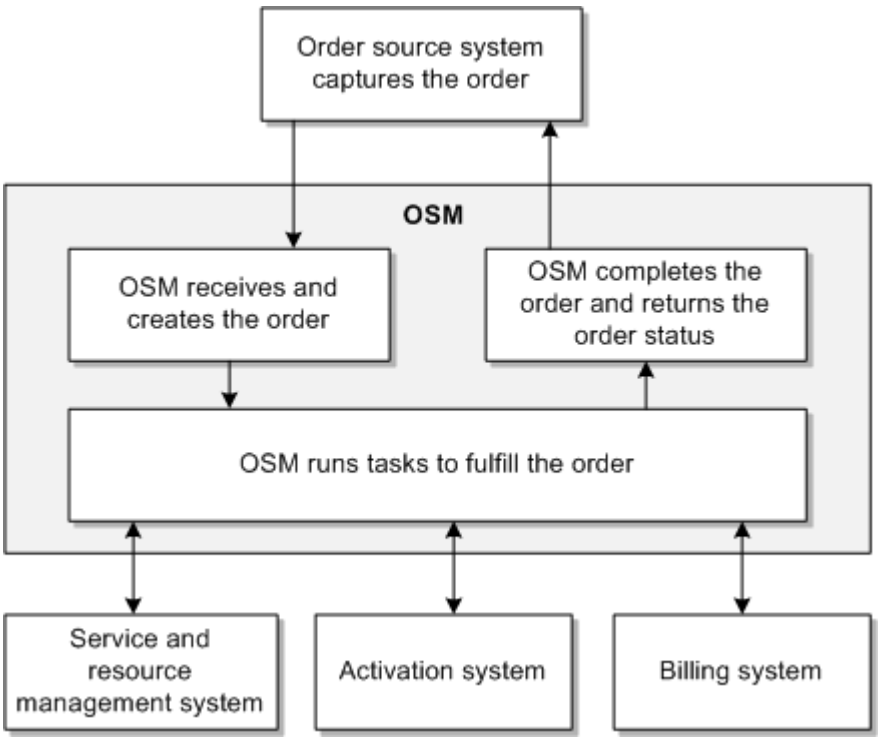
An order in OSM can include:

- The fulfillment functions that need to be completed, such as add or cancel a service
- The requested completion date
- Customer data that is relevant to the order process; for example, the customer's address
- Information about the services being requested; for example, telephone number, bandwidth, or DSLAM port
- Status information; for example, if the order is in progress or completed

4. OSM fulfills the order by running **tasks**. Some tasks are manual; for example, you might want an order processor to manually validate that an equipment installation has been completed. Most tasks are automated; for example, a task that sends an activation command to the network.
5. As tasks are completed, OSM monitors the overall status of the order. You can use the OSM Task web client and the OSM Order Management web client to track the order's progress and manage any problems that occur.
6. When all of the tasks are complete, OSM informs the order source system that the order has been fulfilled, and the services are available to the customer.

Figure 1-2 shows the order fulfillment process.

Figure 1-2 OSM Order Fulfillment Process



For a detailed description of the OSM order process, see "How OSM Processes Orders."

About Order Fulfillment Business Processes

OSM supports the following business processes:

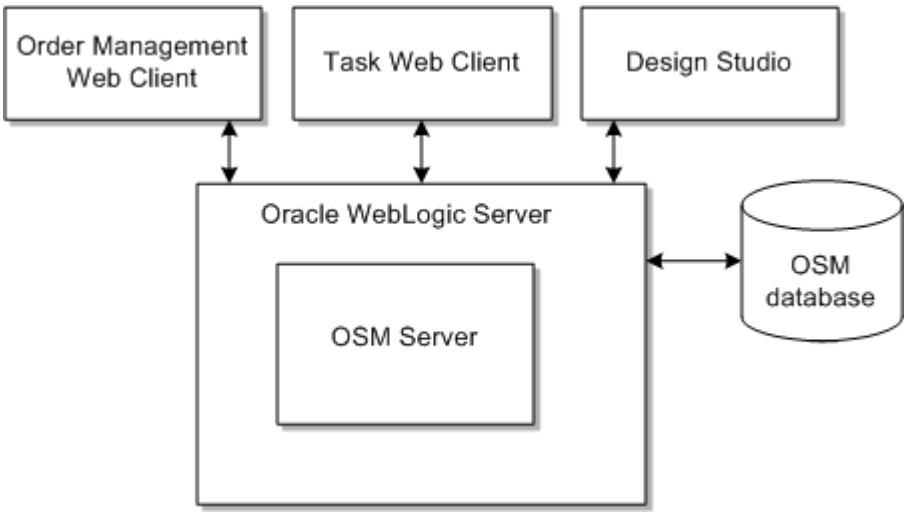
- If an order is in progress and a customer needs to change it, you can resubmit the order to OSM. OSM can roll back and change fulfillment tasks as needed. For example, an order might be in progress that specifies a 5Mbps bandwidth. If the customer decides to change the order to 13Mbps bandwidth, the same order with the new requirement is submitted. The original order is called the **base order**, and the second order is called the **revision order**.
- You can calculate the length of time that an order might take to process and provide customers with an expected delivery date. In some cases, a customer might need a service activated at a date in the future; for example, at the end of the next month. OSM can calculate when to start the order to enable the service at the required date.
- You can configure various ways to report order status, including the status of the entire order and the status of individual fulfillment tasks. You can use **fulfillment states** to combine the various statuses received from multiple external systems into an aggregated status.
- You can create processes to manage errors in order processing. An error in order processing is called **fallout**.
- You can create **workgroups** to manage manual order processing tasks and order fallout. You can use workgroups to assign tasks to specific types of order operators; for example, order operators who interact with customers, or order operators who manage failed orders.

For more information, see "About Order Management Business Processes."

About the OSM System Architecture

The OSM system includes system server components, web-based GUI applications, utilities, and interfaces for communicating with external systems. Figure 1-3 illustrates the OSM system components.

Figure 1-3 OSM System Architecture



An OSM system includes the following client components:

- Use the **Task web client** to monitor and manage the tasks in an order. This application is typically used by order processing personnel to ensure that all the tasks are completed. It is also used by order fallout managers. You can also perform some order management tasks, such as suspending, resuming, and canceling orders.
- Use the **Order Management web client** to display detailed information about the progress of the order. Displaying the order process is useful for developers who are modeling orders and need to see relationships between the specifications they create in Design Studio. You can also perform some order management tasks, such as suspending and resuming orders, canceling orders, and managing fallout. You can open the Order Management web client from within the Task web client.
- Use **Design Studio** to model types of orders, tasks, and other OSM entities that run the order fulfillment process. When you design your order fulfillment process, you also use Design Studio to design entities for other Oracle products; for example, to configure services for UIM, and to design activation processes for ASAP.

  To implement the order fulfillment process that you design, you create **cartridges** in Design Studio and deploy them on OSM server systems. You can deploy multiple cartridges to manage different functional areas; for example, cartridges that deploy OSM fallout management functions, and cartridges that deploy UIM functions.

OSM includes the following server components:

- The **OSM server** manages OSM runtime functionality, including inbound order operations and outbound communications with external systems. The OSM server is deployed on Oracle WebLogic Server.

  You typically configure multiple instances of the OSM server. For example, you might configure one OSM server instance to receive and process incoming sales orders, and one or more OSM server instances to process orders that require interaction with provisioning and activation systems.

  To communicate with external systems, the OSM server uses mostly Java Message Service (JMS) queues. JMS is part of the standard Java platform. In traditional OSM, a JMS queue is a staging area that you configure when you install OSM. In OSM cloud native, JMS queues are added to the project specification for an instance. Systems communicate via JMS queues by publishing messages to them and receiving messages from them.
- Oracle WebLogic Server hosts the OSM server. Oracle WebLogic Server, part of Oracle Fusion Middleware, provides Java JEE services for the hosted components and includes availability, scalability, manageability, clustering, and performance features.
- OSM uses Oracle Database to store orders being processed and orders that have been processed. The OSM database also stores the OSM **metadata** that you create when modeling the order fulfillment process. For example, the order specification that you define in Design Studio is stored as metadata. OSM uses that metadata as a template to create instances of orders at runtime. OSM metadata defines the order model and behavior of the OSM Server, including order templates and tasks.

  OSM uses Java Database Connectivity (JDBC) to carry out communication between server components and the OSM database.

For information about administering an OSM system, see *OSM System Administrator's Guide*.

About Creating an Order Fulfillment Process

Using OSM includes two different types of activities:

- **Design-time activities**. To implement OSM, you define the content of each type of order, and the process that fulfills the order. To do so, you use Design Studio to model types of orders, tasks, and other OSM entities that run the order fulfillment process. As the products, offers, and bundles in your product catalog change, you use Design Studio to make changes to the order fulfillment process. OSM includes sample Design Studio cartridges to use as a starting point, but you must model your own order fulfillment process.
- **Runtime activities**. To manage orders, you can use the Task web client to run manual tasks, and you use the Task web client and the Order Management web client to track the progress of the orders and manage any problems that occur. For example, if an external system reports an error, you can use the Order Management web client to troubleshoot the order. For more information, see "About Runtime Order Management."

OSM fulfills orders to support your specific product offerings. For example, if you sell a DSL service, you model your order process to include the data necessary to activate the DSL service on the network, and to carry out provisioning and activation tasks. If you sell a fixed line telephone service, your order process needs to carry out a different set of fulfillment functions.

To design and implement your order fulfillment process, you do the following:

1. Define your business requirements; for example, the products, bundles, and offers you sell.
2. Plan how to implement the fulfillment requirements for those products and services. For example:
   - Which systems (activation, inventory, billing) does OSM need to communicate with?
   - What data is needed to activate a service?
   - Which tasks need to be performed manually, and which can be performed automatically?
   - How are changes to an order handled?
3. Model the orders and the fulfillment processes in Design Studio and test the order execution.
4. Implement the order fulfillment process in your production system.
5. As your business changes, create new types of orders and implement changes to how orders are fulfilled. For more information, see "Managing Changes in Your Business."

For more information about designing and modeling OSM, see *OSM Modeling Guide* and *Design Studio Concepts*.

2

How OSM Processes Orders

This chapter provides an overview of how Oracle Communications Order and Service Management (OSM) processes orders. Before reading this chapter, read "Order and Service Management Overview ."

About Order Processing

To fulfill an order, OSM initiates actions in external fulfillment systems. For example, if an order needs to implement a service, OSM initiates network activation on an activation system. The fulfillment functions required to fulfill an order are carried out by running tasks. A **task** is an OSM function that initiates work that needs to be done to complete an order.

For example, if an order needs to implement an email service, you could model a task such as Create Email Account. This task would send a message to the network to activate the service, and to receive a notification back from the activation system.

Tasks are run by running a process. A **process** is one or more tasks, run in sequence. You create tasks and processes when you model the order fulfillment process.

To interact with multiple external systems, you can create multiple processes. Some processes need to be completed before other processes, whereas other processes can be run independently. To manage multiple and related processes, you use orchestration. **Orchestration** identifies dependencies between processes and runs them in the required sequence.

OSM uses three basic steps to process an order:

1. **Create the order.** OSM receives the sales order and creates an order in OSM.
2. **Generate the orchestration plan.** The orchestration plan manages how the processes that fulfill the order run.
3. **Run processes and tasks.** The tasks interact with external systems to complete the order. The processes control how the tasks run.

The following procedure describes how OSM processes an order:

1. OSM receives the order from the CRM system. The order specifies the data and fulfillment actions required to fulfill the order. For example, the order might specify that the customer needs a telephone and a telephone number. These requirements are specified in **order line items** in the sales order; for example:
   - Add mobile service
   - Add handset
2. After OSM receives the order, it creates an order in the OSM format. Because you can have multiple types of orders, OSM uses **recognition rules** to determine what type of order to create; for example, for a specific type of service or product offering. You can also use recognition rules to receive orders from order source systems that use different order formats. In that case, you can create multiple order recognition rules that point to the same target order. The order recognition rules would transform the incoming order data into the target order data format.

See "How OSM Creates Orders" for more information.

3. The order includes a default orchestration process. The **orchestration process** starts generating an orchestration plan. Each order has a unique orchestration plan.

> **Note:**
>
> Some orders can be run without orchestration when the order management requirements are simple and relatively static, but most orders require orchestration.

4. An **orchestration plan** specifies how to fulfill an order; for example, the order in which fulfillment actions should be carried out, and which external systems need to be involved. To determine what the fulfillment requirements are, OSM transforms the order line items in the order into **order items**. Order items are individual products, services, and offers that need to be fulfilled as part of an order; for example:

   - Gold Access Internet Bundle: A set of products and features
   - BroadBand Internet Access: A service that defines how the customer will access the Internet; for example, DSL or cable.
   - ADSL Service: The network resources provisioned to deliver the service.

   Order items have properties, including the action that needs to be taken on the order item; such as Add or Delete. For example, order items displayed in the Order Management web client look like this:

   - BroadBand Internet Access [Add]
   - Gold Access Internet Bundle [Add]
   - ADSL Service [Add]

   The orchestration plan considers each order item and defines:

   - Which functions need to be performed on each order item. For example, provision a service, bill a bundle, activate a DSL resource, and so on.
   - Which external system needs to fulfill the order item. For example, a product or bundle order item needs to be fulfilled by the billing system.
   - **Dependencies** between order items. For example, an activation order item cannot be started until a shipping order item is completed.

   To manage different functions, systems, and dependencies, order items are organized into **order components**. Order components organize order items to enable OSM to process them efficiently.

   For example, to manage billing for a BroadBand Bandwidth Bundle, the BroadBand Bandwidth Bundle order item would be included in order components for:

   - The billing function
   - The billing system that handles the BroadBand Bandwidth Bundle
   - The billing system interaction required for a bundle

   Similarly order items that represent resources, such as DSL are included in order components for the activation function, a specific activation system, and an activation interaction.

   This process of organizing order items into order components is called **decomposition**. The final step in decomposition is to create a set of **executable order components** that

OSM can run. The executable order components include the order items that have been decomposed into them. Every executable order component runs a process, which in turn can run subprocesses and tasks. See "About Orchestration" for more information.

5. OSM implements the orchestration plan by running the executable order components, which run processes, which in turn runs tasks.

   Tasks can be automated or manual. Automated tasks run with no intervention, but manual tasks must be run by an order manager by using the Task web client. The Task web client displays a list of tasks that need to be processed. Figure 2-1 shows a list of tasks in the Task web client. In this figure, tasks are displayed for three different orders, as shown by the order IDs (385, 386, and 388).

Figure 2-1 Tasks Displayed in the Task Web Client

Order ID	Order State	Priority	Task	State
388	In Progress	5	CreateVPNSite	Accepted
386	In Progress	5	Ship Modem Self-Install Pkg	Assigned
386	In Progress	5	Assign Port	Received
385	Not Started	5	Add ADSL Siebel	Received

6. As the order is processed, it is assigned order states. Figure 2-1 shows entries for two orders in the In Progress state, and one order in the Not Started state. You use order states to track the progress of the order. Every order has a set of possible states, called a **lifecycle policy**.

   Tasks are also assigned states. When all tasks in an order reach the Completed state, the order is assigned the Completed state. See "About Tasks and Processes" for more information.

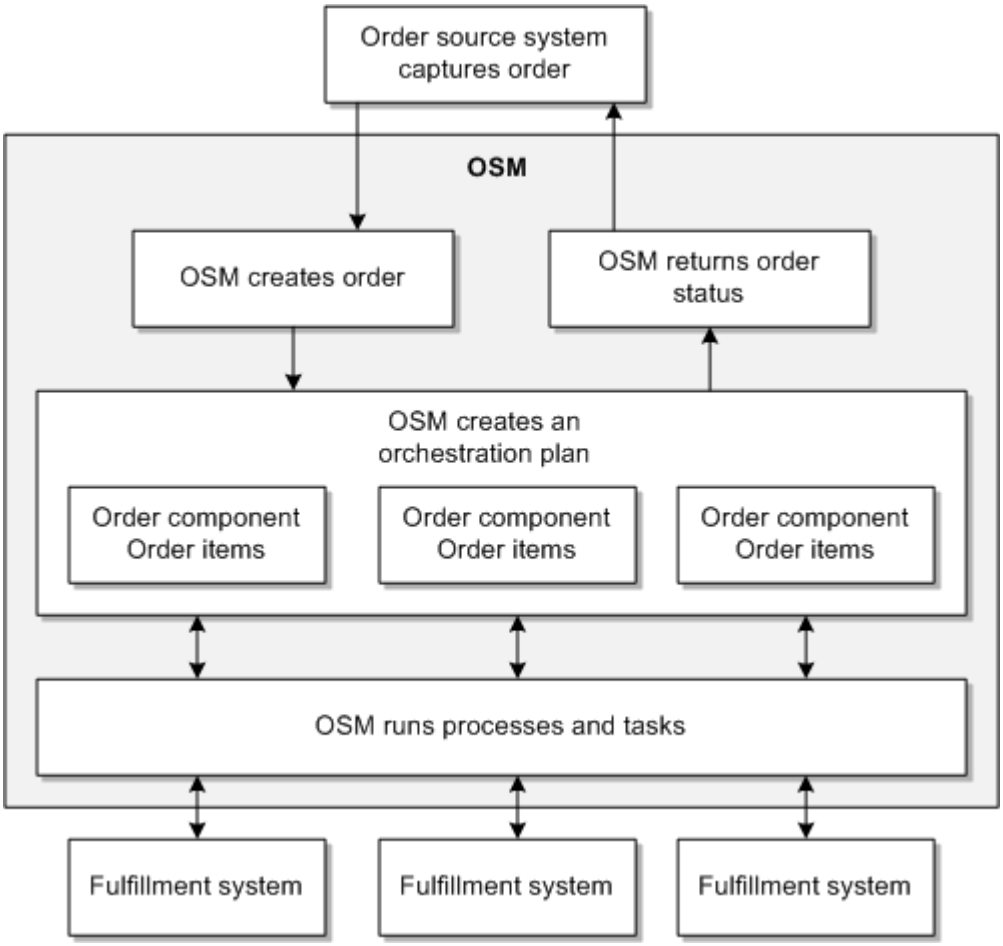
7. As the order progresses, OSM communicates with the originating CRM or order-source system to provide information about the status of the order. You can track the status of tasks, order items, order components, and the order itself. OSM can use **fulfillment states** to aggregate notifications of task completion events to present a real-time, unified view of the order completion process to the originating system and to the OSM web clients.

   You can also use predefined order item **processing states** to aggregate notifications of task events to issue warnings and identify failures.

8. When all order items for the order are complete, OSM closes the order and informs the originating system that all of the fulfillment tasks are complete.

Figure 2-2 shows a high-level view of the order fulfillment process when using orchestration.

Figure 2-2 Order Fulfillment Process by Using Orchestration



About Customer Orders, Service Orders, and Technical Orders

To process an order, OSM typically performs these general types of functions:

- Receive an order from a CRM, and initiate fulfillment actions on external systems.
- Interact with a billing system to handle charging and billing actions on the products, bundles, and offers.
- Interact with a service and resource management system to design the services that need to be implemented, and assign the resources needed.
- Interact with activation, shipping, and work order management systems to implement the services on the network and install equipment at the customer site.

To manage these general functions, you can create an order process that uses three orders. For example:

1. An instance of OSM creates an order, called a **customer order**. The customer order interacts directly with the CRM system and runs tasks that interact with external systems such as a billing system. To provision services, OSM sends an order to another instance of OSM.
2. The second instance of OSM creates an order to communicate with a service and resource management system. This order is called a **service order.** The service order finds the

resources needed for the service, and sends an order to another instance of OSM to process activation and shipping tasks.

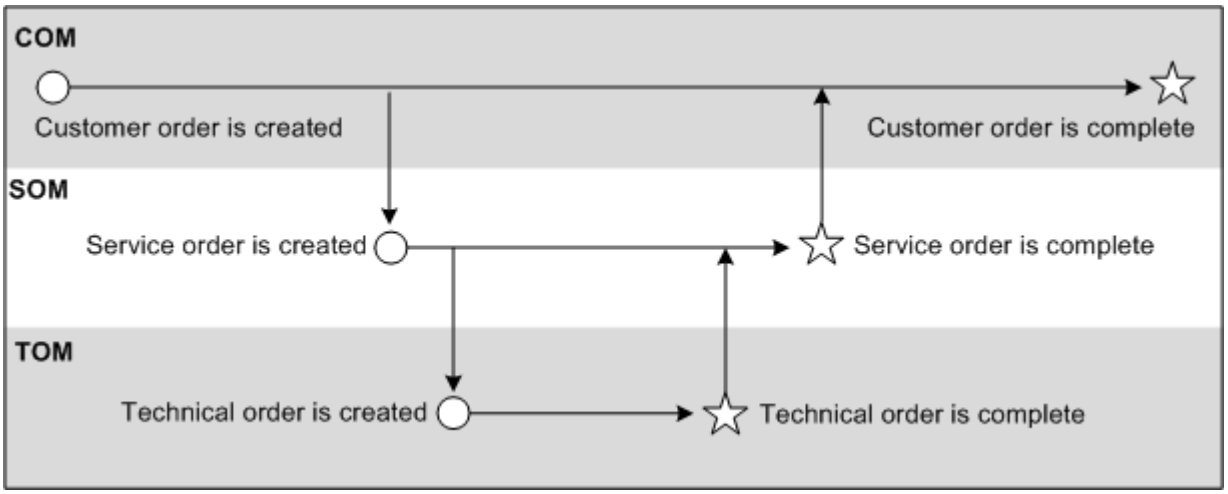
3. The third instance of OSM creates an order to design the service, assigns the resources, and determine the actions required to fulfill the service using those resources. This order is called a **technical order**.
4. When the technical order is complete, the service order can be completed, and, in turn, the original customer order can be completed.

To manage this type of scenario, these three types of orders are processed by OSM running in three different roles:

- Central order management (COM) runs customer orders.
- Service order management (SOM) runs service orders.
- Technical order management (TOM) runs technical orders.

Figure 2-3 shows how COM, SOM, and TOM work together:

Figure 2-3 Correlation Between COM, SOM, and TOM



A sample order fulfillment process that uses COM, SOM, and TOM runs as follows:

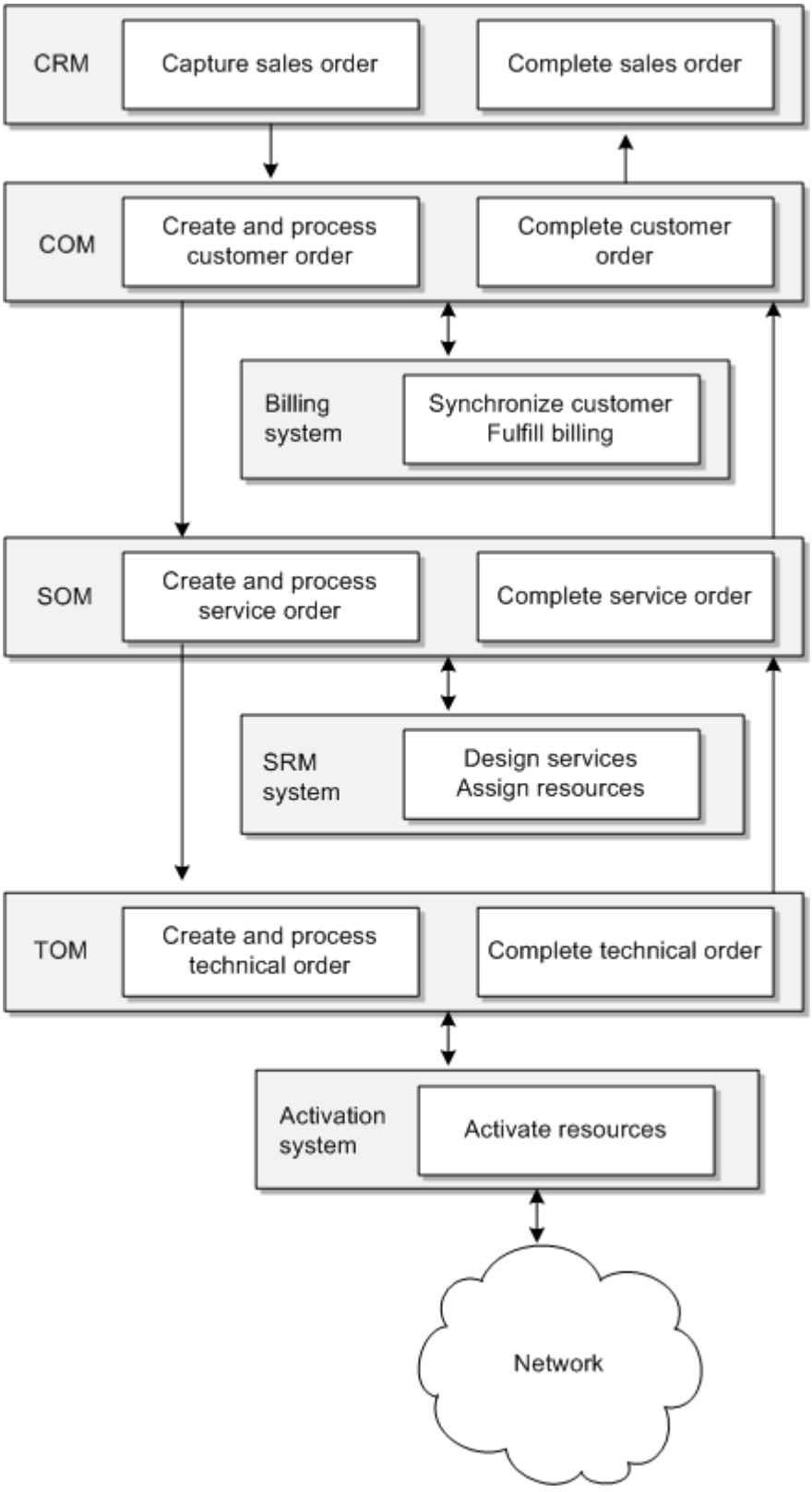
1. OSM in the COM role receives an order from the CRM system and creates a customer order. As described in "About Order Processing," OSM generates an orchestration plan.
2. When the orchestration plan is run, OSM decomposes order items into:
   - An order component that interacts with a billing system to run billing
   - An order component that sends a service order to an instance of OSM in the SOM role to provision the service
3. OSM in the SOM role processes the service order. In this case, OSM uses orchestration again, and OSM decomposes order items into:
   - An order component that interacts with a service and resource management (SRM) system to design the service and assign resources. For example, the service might need a local loop, telephone number, and so on. An SRM system is also known as an inventory system.
   - An order component that sends a technical order to OSM in the TOM role to manage service activation and shipping.

In the SOM role, orchestration is used to ensure that the service design occurs first. The inventory system needs to send data about the network resources and the actions required on those resources back to OSM, so OSM can include that data when processing activation, shipping, and interactions with a partner gateway.

4. OSM in the TOM role processes the technical order and decomposes order items into:
   - An order component that sends activation requests to the network.
   - An order component that sends requests to a shipping system.
   - An order component that sends requests to a partner gateway to create a local loop.
5. When the activation and shipping tasks are complete, OSM in the TOM role sets the status of the technical order to Completed and informs OSM in the SOM role that the activation order is complete. This is done by using order item processing states, order lifecycle policy states, and, potentially, fulfillment states.
6. OSM in the SOM role sets the status of the service order to Completed and sends a message to the originating OSM instance in the COM role.
7. While the service order and technical order were running, the customer order processes the billing and shipping functions. Upon being informed that the provisioning and activation orders have been completed, OSM completes the original order.

Figure 2-4 shows how an order is fulfilled by using three orders, sent to three OSM instances in the COM, SOM, and TOM roles.

Figure 2-4 Order Fulfillment Using COM, SOM, and TOM



About COM, SOM, and TOM

As described in "About Customer Orders, Service Orders, and Technical Orders," OSM can run in three roles, COM, SOM, and TOM, to process customer orders, service orders, and technical orders.

OSM in the COM role typically interacts with a billing system to perform such tasks as synchronizing customer accounts between the order source system and the billing system, and initiating billing activities in billing systems. OSM in the COM role also typically identifies the services that are associated with the products, bundles, and offers, and sends that data to OSM in the SOM role in a service order.

> **Note:**
>
> OSM in the COM role can also interact with workforce management (WFM) and supply chain management (SCM) systems to ship products to customers. However, shipping tasks may require knowledge of the services and resources being activated and shipped; for example, the service design process might determine which type of modem to ship. Therefore such shipping tasks should typically be delegated to OSM instances running in the SOM or TOM role.

OSM in the SOM role works with service and resource management (SRM) systems to design services, assign the resources required to fulfill the services, and define how those resources need to be configured to fulfill the services. This process is called **design and assign**.

To design and assign services, OSM in the SOM role uses the data received in the service order. OSM in the SOM role sends that data to an SRM system to design the service and assign resources. As part of the service fulfillment design in Design Studio, you model predefined service configurations in your SRM system, such as UIM.

The design and assign process works as follows:

1. OSM sends the SRM system a request to design a service and assign resources. The request specifies the type of service, for example, broadband Internet, the requested service attributes, such as upload and download speed, and relevant data, such as the location of the customer.
2. Given the customer requirements, the SRM system determines which predefined service configuration is appropriate, and based on that, finds the network resources that are available. For example, if Broadband Internet Service maps to DSL service, the SRM system knows that the DSL service design requires a port and a local loop. The SRM system finds an available local loop at the customer's location and assigns it to the customer's service.
3. The SRM system returns the resources, resource-facing services, and their associated actions to OSM. The SRM system also changes the status of the resources in the inventory.

By using the design and assign process in a service order, the incoming sales order does not need to include any information about the existing installed network resources, such as local loops, ports, and so on. The incoming order needs to describe only the type of service, the desired attributes such as bandwidth, and any information that affects the choice of resources, such as the customer's location.

The design and assign process completes the transformation from a **customer-facing service** (CFS) to a **resource-facing service** (RFS). A CFS is a representation of the service that the customer purchased. An RFS is how the service is implemented on the network.

For example, a customer might purchase a product offering named Gold Broadband Service. The CFS is Broadband Internet Service. How that service is implemented on the network is the RFS, in this case DSL Service. Therefore, the CFS Broadband Internet Service is resolved to RFS DSL Service. However, the customer's requirements might be such that DSL is not possible, but a cable broadband access is possible. In that case, the CFS Broadband Internet Service is resolved to the RFS Cable Internet Service.

Because the resource-facing services are pre-configured in the SRM, the SRM can design the resource-facing service and assign resources based only on the requirements of the customer-facing service.

After receiving the required data from the SRM system, the OSM SOM instance sends a technical order to OSM in the TOM role. In the TOM role, OSM processes the technical order and orchestrates the activation, shipping, and installation tasks. The systems typically involved in these activities are WFM, SCM, and activation systems. Partner gateway (PGW) systems for third party service providers or trading partners can also be involved at the TOM level.

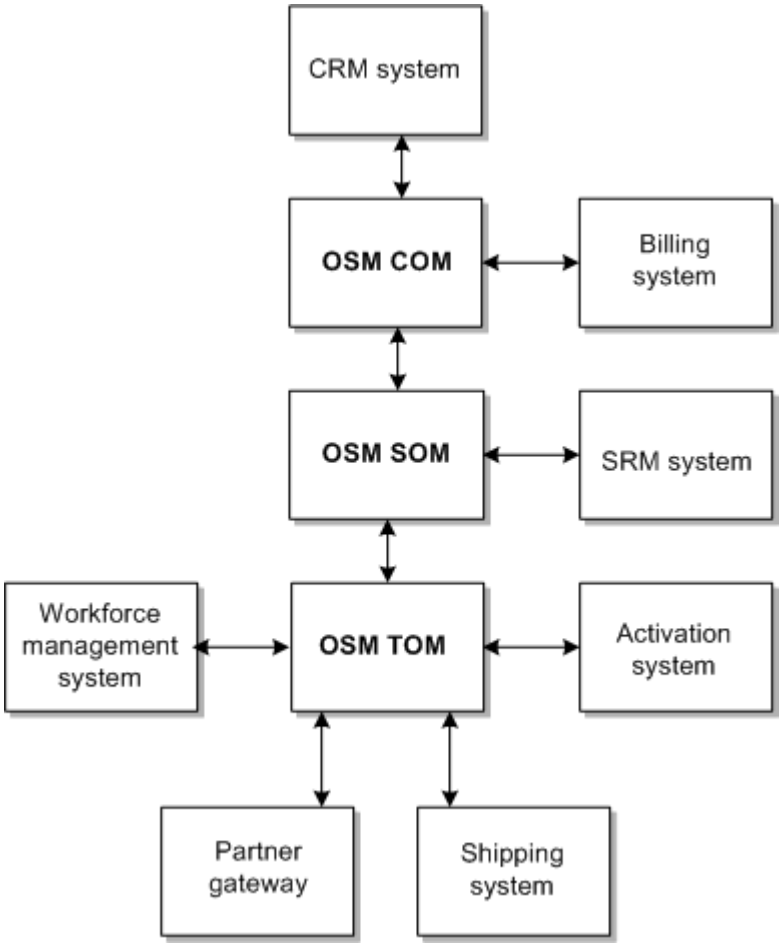
After completing the tasks in the technical order, OSM in the TOM role communicates the order status to OSM in the SOM role, which in turn communicates its order status to OSM in the COM role. OSM in the COM role can then complete the original customer order.

> **Note:**
>
> OSM can also send the status of individual order items to the order source system while the order is being processed.

By using COM, SOM, and TOM, OSM is able to take as input the products, bundles, and offers that the customer purchases, and resolve those into customer-facing services and, ultimately, the resource-facing services that need to be implemented on the network.

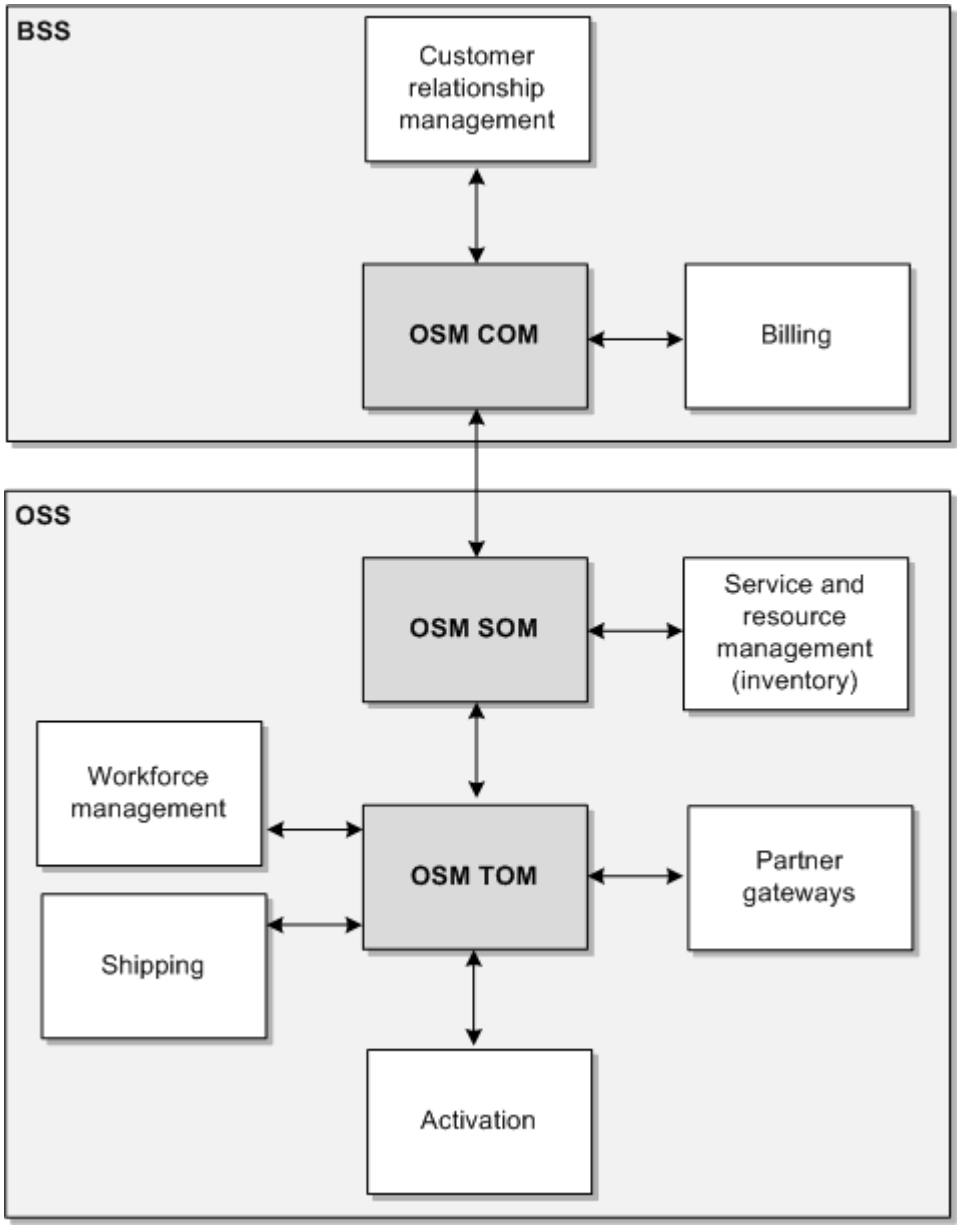
Figure 2-5 shows a fulfillment topology that uses COM, SOM, and TOM. A **fulfillment topology** consists of all of the fulfillment systems required to fulfill orders, and the relationships between the fulfillment systems.

Figure 2-5 Fulfillment Topology Using COM, SOM, and TOM



A typical OSM fulfillment topology includes business support systems (BSS) fulfillment systems and operations support systems (OSS) fulfillment systems. Figure 2-6 shows how OSM in the COM, SOM, and TOM roles function with BSS and OSS systems.

Figure 2-6 Roles of SOM, COM, and TOM in BSS and OSS Fulfillment Systems



3
How OSM Creates Orders

This chapter provides an overview of how Oracle Communications Order and Service Management (OSM) receives and creates orders. Before reading this chapter, read "Order and Service Management Overview " and "How OSM Processes Orders."

How OSM Receives and Creates Orders

To enable order processing, you need to configure your order-source systems to send orders to OSM. For example, if you use Siebel as a customer relationship management (CRM) system, you configure Siebel to send a sales order to OSM. A single OSM instance can receive orders from multiple order-source systems.

OSM receives and creates orders as follows:

1. The order is captured in a CRM system; for example, as a Siebel sales order.
2. The CRM system sends the sales order to OSM by using the OSM CreateOrder web service operation. The OSM Web Service API is the primary API for external clients that you can use to communicate with OSM (see *OSM Developer's Guide* for more information).
3. OSM receives the incoming sales order as an XML message in a Java Message Service (JMS) queue managed by Oracle WebLogic Server. Incoming orders are processed directly by the OSM server from the queue.
4. Because a single instance of OSM can process multiple types of orders, OSM needs to determine which type of order to create. For example, if you have created different order specifications for different services, OSM needs to find out which service needs to be fulfilled to know which order specification to use when creating the order.

   To identify the order specification, OSM uses **recognition rules**. Recognition rules read the data in the incoming order; for example, the type of service, the fulfillment mode (the action to carry out, such as add a service or cancel a service), or the type of customer can all be used to identify the correct order specification. See "About the Data in an OSM Order."

   In most cases, you use one generic order type to create all orders. You can use orchestration to define the order fulfillment process for different service domains and other processing requirements. Oracle recommends having one standard order type that accepts all incoming orders with other order types for only very specific uses, such as a fallout management order type that can extract information about a failed order.

   You typically model a different order specification when the structure or order data is different from any existing order type, or when there are specific and different fulfillment requirements.
5. After the order specification has been identified, OSM uses validation rules to perform **order validation**. For example, you can use validations rules to:
   - Ensure that all mandatory fields are populated.
   - Ensure that valid characters (numeric or alphanumeric) are used as needed.

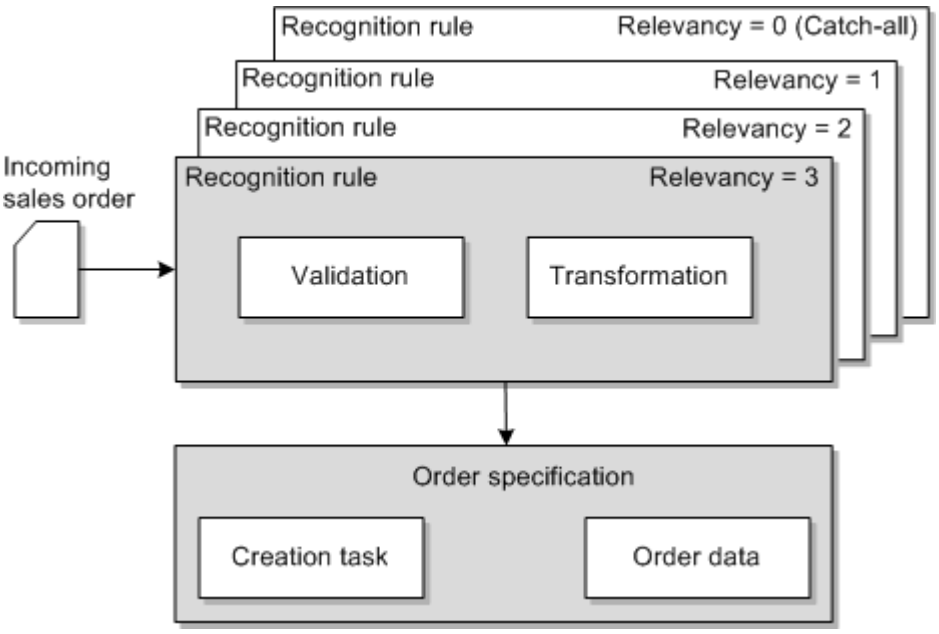
6. Following order validation, OSM uses transformation rules to perform order transformation. **Order transformation** does the following:
   - Normalizes data; for example, ensure that telephone number formats are correct. If there are multiple different CRM systems sending orders with different data structures, you can transform each order into the target order format.
   - Adds data needed for order processing. This data is part of the creation task. The **creation task** is used by OSM in the order process; for example, when an order is canceled, the order is returned to the creation task. The creation task is required in all orders.
   - Sets the order priority. The **order priority** defines the processing priority of the order in relation to other orders in the system.
   - Sets the order reference number. The **order reference number** is used as an identifier to external systems; for example, between the COM order that generated a child SOM order, or between two sibling SOM orders within an OSM SOM instance.
7. OSM creates the order and runs the default orchestration process to generate the orchestration plan. Every order includes a **default process** that is run when the order is created.

> **Note:**
>
> If an order does not require orchestration, it can run a process directly, not an orchestration process. Most orders require orchestration.

8. OSM runs the orchestration plan to process the order. See "About Orchestration."

Figure 3-1 shows how OSM receives and creates orders. To identify the order specification, you create multiple recognition rules. Each rule is assigned a relevancy. OSM uses the highest relevancy first, until an order specification is found. You can create a catch-all recognition rule (relevancy = 0) to allow OSM to accept any order, even if it cannot process it. This allows you to troubleshoot how orders are received.

Figure 3-1 Receiving and Creating an Order



About the Data in an Incoming Order

Incoming sales orders typically include the following data:

- The **order header** information, which contains information that is applicable to the entire order; for example, the sales order number, the order action (Add, Cancel, Delete, and so forth), and customer account information.
- The order line items, which contain information about the products and services on the order.

Figure 3-2 shows part of a sales order, received from a CRM system.

Figure 3-2 Example of a Sales Order

Orders that are submitted to OSM typically have a specific purpose that is defined as an **order action**. This information is usually included in the order header to indicate if the order adds or cancels products and services for a customer. For example, the following line from an incoming sales order specifies that the order adds services:

```
<im:typeOrder>Add</im:typeOrder>
```

The order action is used in an orchestration plan as the fulfillment mode. **Fulfillment modes** enable OSM to generate an orchestration plan based on whether a service needs to be added, changed, canceled, or qualified, such as validating that the network has enough capacity to offer the purchased products.

Whereas the order header typically contains data that pertains to the entire order, the order line items describe the individual fulfillment actions that need to be carried out. Figure 3-3 shows

one of the order line items in its expanded form. Included in this order line item are the requested delivery date and the action to take (Add).

Figure 3-3 Example Order Line Item in a Sales Order

```
<!-- Speed By Demand - PRODUCT -->
<im:salesOrderLine>
   <im:lineId>16</im:lineId>
   <im:parentLineReference>
      <im:parentLineId>14</im:parentLineId>
      <im:hierarchyName>default</im:hierarchyName>
   </im:parentLineReference>
   <im:rootParentLineId>13</im:rootParentLineId>
   <im:promotionalSalesOrderLineReference>1
   <im:serviceId></im:serviceId>
   <im:requestedDeliveryDate>2016-12-31T12:00:00</im:requestedDeliveryDate>
   <im:serviceActionCode>Add</im:serviceActionCode>
   <im:serviceAddress>
   <im:itemReference>
</im:salesOrderLine>
<!-- END Speed By Demand - PRODUCT -->
```

Order line items include details about the services that the order must fulfill. They can include:

- The offers, bundles, and products being ordered by the customer, the services these products ultimately resolve to, and the resources that deliver them.
- Information about the services; for example, speed, storage size, and requested service date

Each order line item in an incoming sales order that OSM receives specifies an action to perform. Order line item actions are typically one of the following:

- Add a product, service, or resource.
- Change an existing product, service, or resource
- Delete a product, service, or resource
- Update attributes of a product, service, or resource
- Cancel an existing product, service, or resource
- Move a product, service, or resource
- Suspend or resume a product, service, or resource

An order can contain a mix of actions for different products or services. For example, an existing customer might request to add some new products, change some existing products, and remove other products. These can all be included on the same order.

Incoming order item lines also include data that is used later in the service fulfillment process, but not needed by the initial customer order that OSM creates. For example, a customer's street address might not be needed until a technical order is processed to assign a local loop.

About the Data in an OSM Order

When a sales order is captured in an order-source system, it includes data such as the customer's name and contact information, customer billing information, the products that the customer is ordering, and the requested date of delivery. A subset of that information is included in the sales order that is sent to OSM; for example, the customer information and the order line items that specify the offers, bundles, and products, and the actions that must be performed on them.

When you model the data in an order specification, you specify the mandatory and optional data that OSM uses to fulfill the service. For example, in an order for a telephone service, the order must include a telephone number. The data an order can contain is called **order data**, and is defined in the **order template**. You can define data in data dictionaries and then import these elements into order templates. Modeling data in data dictionaries enables you to reuse the same data definitions across a solution.

The metadata that you model in the order template defines the data that the order can include at runtime. For example, a runtime order can include the following data:

- Information about the order. For example:
  - The type of order, such as a request for a new service or a change to an existing service.
  - Order creation date.
  - Expected completion date.
- Information about the customer; for example, name and address.
- Information about the services being requested; for example, upload speed, download speed, and quality of service.
- The order components, order items, processes, tasks, and dependencies that are required to fulfill the order.
- Status information. For example:
  - If the order is still in flight, or if it has completed.
  - State of the tasks that need to be performed.
- Tracking information; for example, remarks, notifications, and order history.

The data in customer orders, service orders, and technical orders is typically different for each type of order:

- Customer orders include information about the customer, such as their location, the product offerings that the customer purchased, and the product requirements, such as download speed.
- Service orders include information about the customer-facing services that need to be provisioned, including the technical requirements such as bandwidth and quality of service, and the customer's location.
- Technical orders include information about the resources and resource-facing services that need to be activated, and the equipment that needs to the shipped or installed. Resources and resource facing services are identified by the SRM system from customer-facing services that OSM SOM sends to the SRM system.

Data Used for Processing an Order

In addition to data such as the customer's address and phone number, an OSM order includes information that defines how the order is run. This includes:

- **Control data.** Control data provides information about order items, order components, and dependencies required to generate the orchestration plan. This includes status and requested delivery dates for its order items and components. You can also track order and order item fulfillment states and order item processing states in control data.
- **Behaviors.** You can use behaviors to manipulate data and to control how data is displayed in the Task web client. For example, you can specify the minimum and maximum times that a data element can be used in an order. See "About Using Behaviors to Customize the Task Web Client."
- **Notifications.** You can use notifications to alert users and external systems to events that occur in the order as it processes or to tell users that an action must be carried out. See "About Notifications."
- **Lifecycle policy**. The lifecycle policy defines the states that an order can have; for example, In Progress and Suspended, and the rules governing transitions between states. See "About the Order Lifecycle Policy."

About the Order Lifecycle Policy

Every order specification you create must be associated with an order lifecycle policy. The lifecycle policy defines the states that an order can be in, (such as In Progress or Canceled), the rules governing the transitions between those states, and who is authorized to initiate those transitions. For example, you can specify that an order can be transitioned to the Suspended state only when it is in the In Progress state, and only by OSM users of a designated role.

OSM allows any number of order lifecycle policies to be configured. You can create a custom policy for each order type or one general policy that is applied to many order types. The default order life cycle contains the minimum set of order state and transaction combinations assigned to all roles defined in the system.

Customizing an order lifecycle policy enables you to control the following:

- You can specify conditions that need to be met before an order can transition from one state to another. A common example is specifying the **point of no return** for revision orders, which controls the transition from the In Progress state to the Submit Amendment and Process Amendment states, effectively causing OSM to reject any further revision orders for the base order.
- You can specify a grace period that allows the order to complete processing tasks before performing a transition to another order state.
- You can specify the roles that are allowed to perform a transaction. **Transactions** typically transition an OSM order from one order lifecycle state to another. For example, the suspend order transaction causes an OSM order to transition from the In Progress state to the Suspended state.
- You can specify the error message and severity to use when a transition condition is not met.

In addition, you can configure an order to publish common events at specified order lifecycle milestones, such as when an order is first created in OSM, when the order state has changed,

or when the order begins amending. These events can be used to trigger notifications. See "About Notifications."

4

About Orchestration

This chapter provides an overview of how Oracle Communications Order and Service Management (OSM) uses orchestration to manage the order fulfillment process. Before reading this chapter, read "Order and Service Management Overview " and "How OSM Processes Orders."

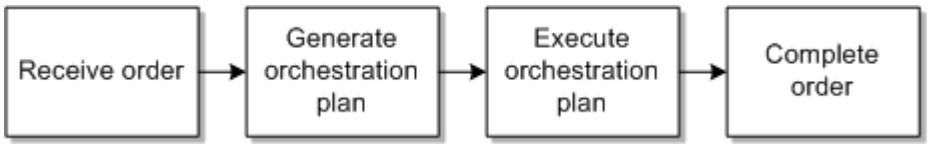
How OSM Generates and Runs an Orchestration Plan

A single order can fulfill multiple products and services. To process an order, OSM needs to run processes and tasks for a variety of functional areas while interacting with multiple systems. To manage all of the orchestration entities and processes, OSM generates an orchestration plan. The **orchestration plan** decomposes order items into order components, and establishes dependencies between order components and between order items.

Each order has a unique orchestration plan, based on the customer's requirements and the tasks required to complete the order. OSM implements the orchestration plan by running the order's default orchestration process. The **orchestration process** begins the process of selecting the order components to run. As order components run their processes and tasks, the orchestration plan manages their dependencies.

Figure 4-1 shows a high-level view of the process flow for generating and running an orchestration plan:

Figure 4-1 Generating and Running an Orchestration Plan



The following process flow describes how OSM typically processes an order.

1. OSM receives the order from the CRM system and uses recognition rules to find the order specification to use when creating the order. Each order specification that you create identifies a default process to run.
2. OSM starts generating an orchestration plan by running the order's default orchestration process.

   The orchestration plan specifies an orchestration sequence. The orchestration sequence specifies the orchestration stages that need to be followed. The stages define the order components that order items need to be organized into; for example, function order components, target system order components, and granularity order components.
3. OSM converts order line items in the incoming order into OSM order items and their order item properties. For example, the order line item shown below becomes the Firewall order item with the Add order line action property.

   ```
   <im:serviceActionCode>Add</im:serviceActionCode>
   <im:name>Firewall</im:name>
   ```

This order items displays as **Firewall [Add]** in the Order Management web client.

4. In addition to creating the order items required for the order, OSM uses the incoming order data to determine:
   - The **fulfillment mode** (Deliver, Cancel, and so on). Fulfillment modes enable you to design different order fulfillment flows depending on whether the order adds a service, qualifies a service, or deletes a service.
   - The **product, customer-facing service, resource-facing service, or resource specification**. A product specification is a group of related products that share common attributes, such as the same **service domain**. For example, the products Broadband Light, Broadband Medium, and Broadband Ultimate would all belong to the ServiceBroadBand product specification. When processing an order, you can process order items that belong to the ServiceBroadBand product specification differently from order items that belong to the ServiceMobile product specification.
   - The **product, customer-facing service, resource-facing service, or resource specification order item action**. You might have an order with a fulfillment mode of Add that designates products that the customer wants to perform actions on such as add, alter, or move.
5. Based on the fulfillment mode and the order item's product, customer-facing service, resource-facing service, or resource specification, OSM assigns each order item to a **fulfillment pattern**.

   In general, order items for a given product specification and a given fulfillment mode need similar fulfillment actions. You create fulfillment patterns that organize order items by the combination of fulfillment mode and product, customer-facing service, resource-facing service, or resource specification. For example, you can define a fulfillment pattern that includes order items for orders that deliver a broadband service. In this case, the fulfillment mode is Deliver, and the product specification is ServiceBroadBand.
6. The fulfillment pattern initiates the first level of decomposition, by decomposing order items into the **function order components** identified in the fulfillment pattern. For example, order items are organized into Billing, Shipping, and Provisioning order components.
7. After decomposing order items into function order components, OSM decomposes the order items in each function component into **target system order components**. For example, the order items that were decomposed into billing function order components might be further decomposed into order components for a wholesale billing system and a retail billing system.
8. After decomposing order items into target system order components, OSM decomposes the order items in each target system order component into **granularity order components**. This is typically the final stage of decomposition, although additional stages can be defined.

   Granularity order components are usually needed when a single fulfillment system must process commands in a specific way. For example, you might need to fulfill billing requirements for mobile and fixed services. You can use different order components to process the billing requirements for those services separately.

   See "About Decomposition" for more information.
9. After order items have been decomposed as much as required, OSM runs executable order components. **Executable order components** run processes that in turn run the tasks to complete the order. The tasks that the executable order component runs are the tasks that complete the order items that the order component contains. For example, an order component that includes activation order items runs a process that runs activation tasks. An order component that includes the following order items can run a single provisioning process (and its subprocesses) to complete all of the order items:

- Fixed Line Service [Add]
- Fixed Call Waiting [Add]
- Fixed Caller ID [Add]

OSM uses the dependencies defined in the orchestration plan to process order components in the correct order. A dependency requires a waiting order item and a blocking order item. The blocking order item is the order item that must be completed before the waiting order item is started. See "About Dependencies."

10. As the order progresses, OSM can communicate with the originating order-source system to provide information about the status of the order. In addition, OSM tracks the status of each order item and order component. You can configure how the order status is reported by modeling fulfillment states and processing states. See "About Order Status" for more information.
11. When the last task in the order completes, the order transitions to the Completed state.

About Decomposition

To process order items that are fulfilled for different functions, and by different target fulfillment systems, OSM organizes order items into **order components**, a process known as **decomposition**.

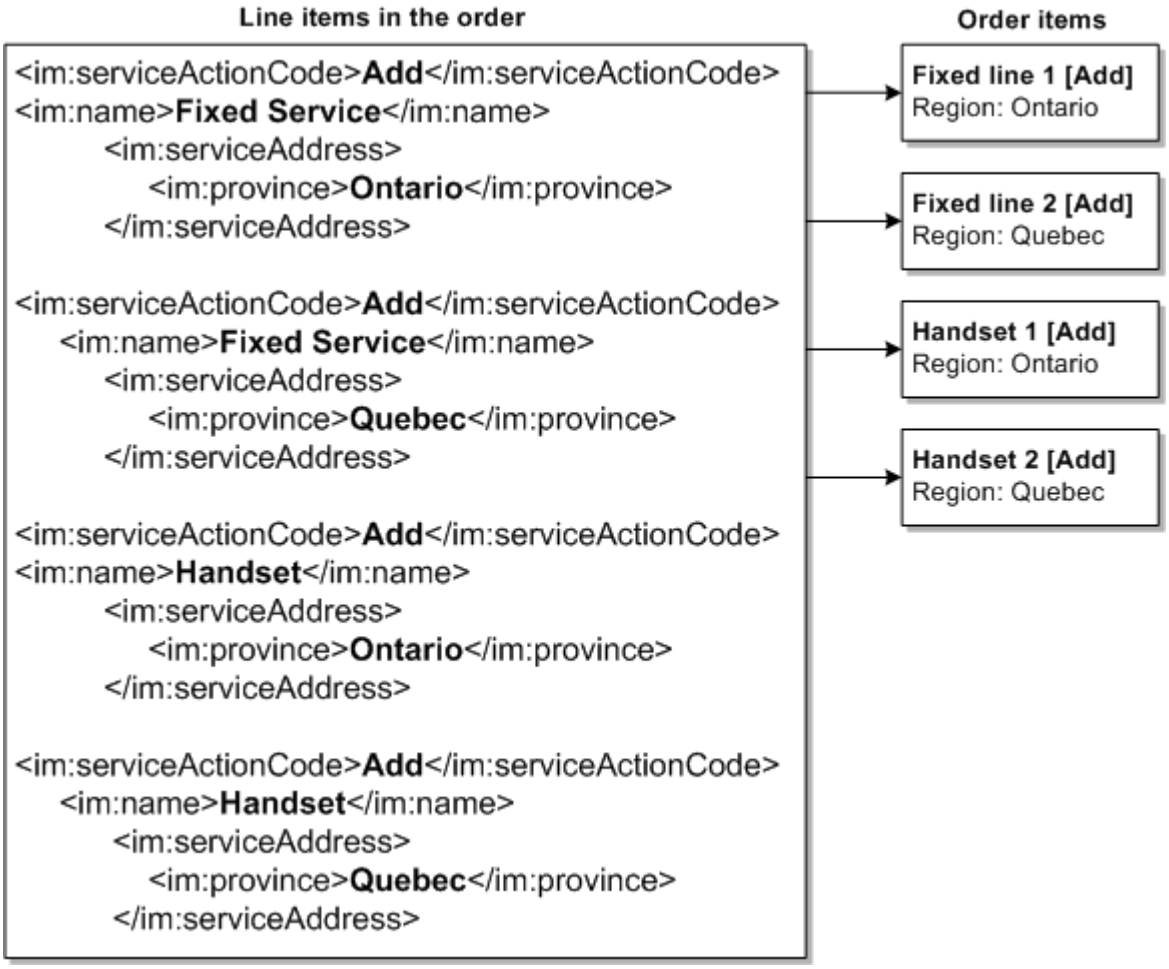
Decomposition occurs in stages. Order items are organized into types of order components from general to specific:

- **Function order components** organize order items by functions; such as billing, provisioning, and activation.
- **Target system order components** organize order items by fulfillment systems; for example, an activation system for DSL and an activation system for VoIP.
- **Granularity order components** separate order components that are fulfilled on the same fulfillment system.

The following examples show how decomposition works.

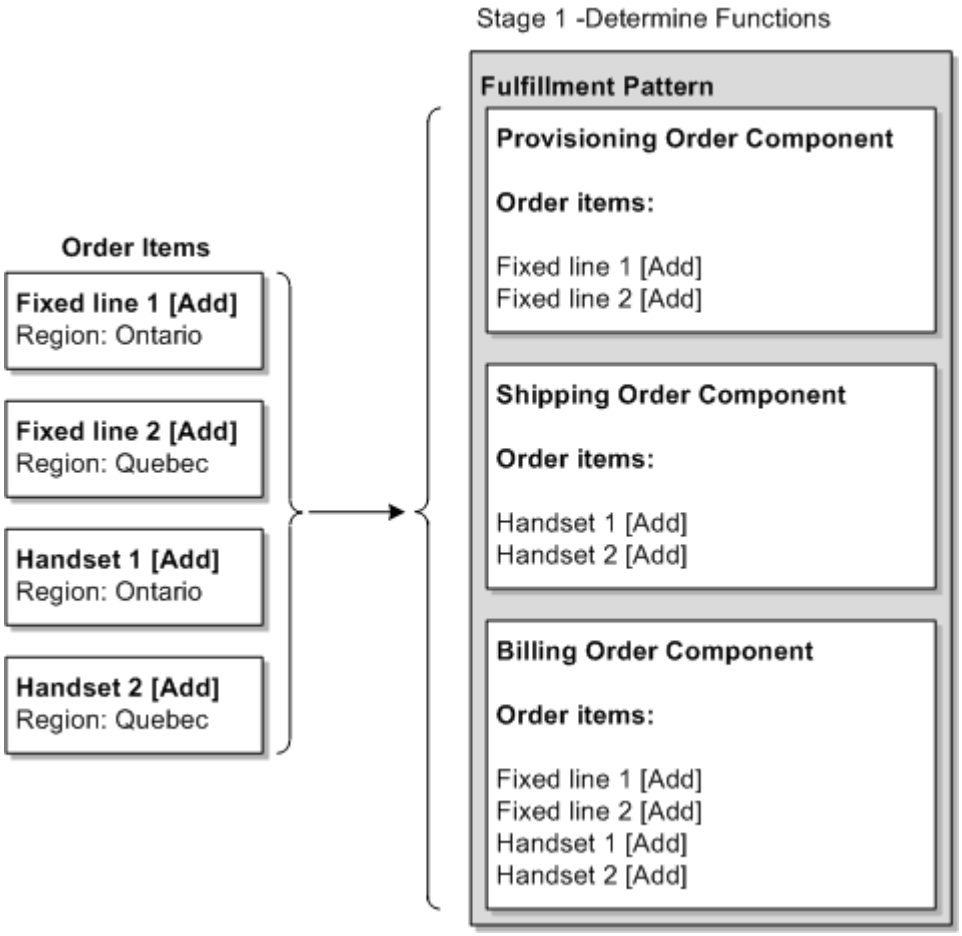
Before decomposing order items into order components, OSM creates order items from order line items in the incoming order. Figure 4-2 shows how order line items for two fixed services and handsets are derived from the order. In addition to separate order items for adding services and shipping handsets, there are different regions defined for each service and handset (Ontario and Quebec).

Figure 4-2 Order Line Items and Order Items



The first decomposition stage (Determine Functions) organizes order items according to function. Figure 4-3 shows how the order items derived in Figure 4-2 are organized into three function order components: Provisioning, Shipping, and Billing. The fixed-line services require provisioning, the handsets require shipping, and all order items require billing, so all order items are included in the Billing order component.

Figure 4-3 Function Order Components

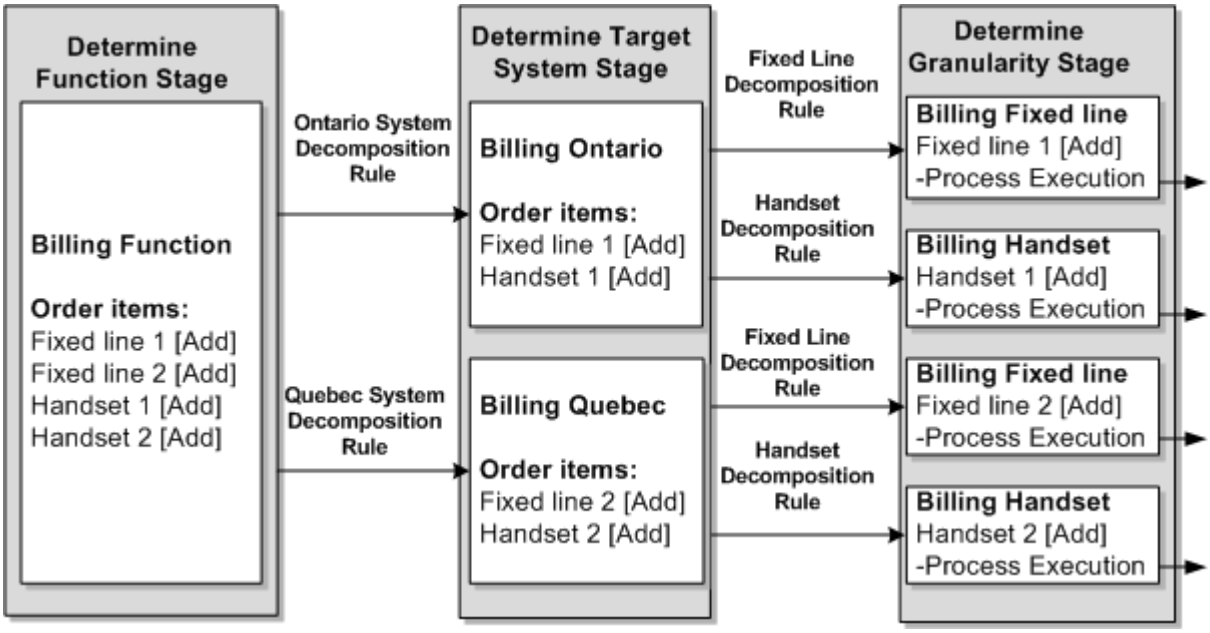


After decomposing order items into function order components, OSM decomposes the order items in each function component into target system order components, and then into granularity order components. These stages are called Determine Target System and Determine Granularity.

Figure 4-4 shows how the order component for the billing function is composed further into two levels of decomposition:

- Order items for the Ontario and Quebec regions are decomposed into target system order components. This sends the billing fulfillment process to the correct region, Ontario or Quebec.
- For each region, the fixed-line service must be billed separately from the handset. Therefore, order components for each target system are further decomposed into granularity components.

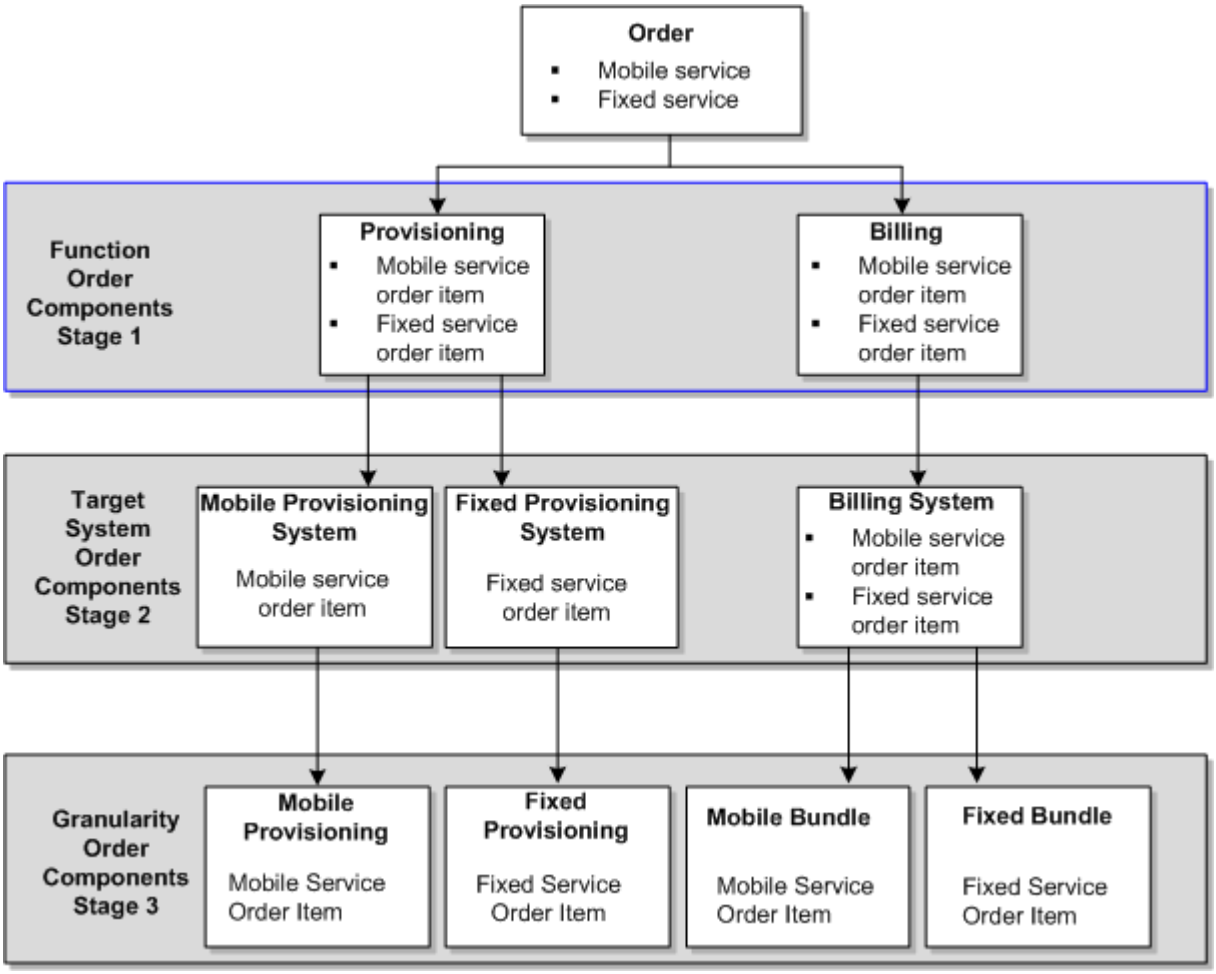
Figure 4-4 Function Order Component Decomposed into Target and Granularity Order Components



As shown in Figure 4-4, you configure decomposition by modeling **decomposition rules**.

Figure 4-5 shows how order items are organized across all three orchestration stages.

Figure 4-5 Order Items Decomposed into Order Components



About Dependencies

An orchestration plan is based on two main factors: decomposition, which organizes order items into order components, and dependencies, which dictate when the executable order components are allowed to run.

Some services might require that some fulfillment tasks are completed before others. For example, you need to complete provisioning order items before you can process activation order items. Dependencies are relationships in which a condition related to one order item must be satisfied before another item can be processed successfully. For example, a piece of equipment must be shipped to a location before the action to install it at that location can be taken.

Dependencies can be between order components in the same order (intra-order dependencies) or between order components in different orders (inter-order dependencies). Inter-order dependencies are particularly common in situations that involve amendments or follow-on orders. For example, the order items in a follow-on order for VoIP provisioning might depend on the processing of the order items in the original order for DSL provisioning. See "About Follow-on Orders."

Dependencies are configured in Design Studio and determined for an order when OSM generates its orchestration plan.

You can model the following types of dependencies. In each of the dependencies below, you can also model a delay after the condition is met. For example, you may want the installation order items to start two days after the shipping order items have completed.

- A **Completion** dependency means that the dependent order item requires that another order item complete before it can begin. For example, the Provisioning order component for the VoIP Service order item cannot begin until the Provisioning order component for the High Speed Internet Service order item is complete.
- A **Data Change** dependency indicates that an order item has a dependency on data in another order item. For example, the status of the Provisioning order component for the VoIP Service order item must be set to Designed before the Ship Order order component for the same order item can begin.
- An **Order Item** dependency indicates that an order item is dependent on the completion of another order item in another order. Because orders can contain many order items, an order component of an order can contain many Order Item dependencies. Order Item dependencies support follow-on orders. Follow-on orders always have at least one order item that has a dependency on an order item in a base order. The line item in the base order must be completed before the line item in the follow-on order can begin processing.

Orders can have combinations of these types of dependencies. For example, an Installation order component may have a Data dependency on the status of a Ship Order order component as well as Order Item dependencies among its order items and order items in other orders.

Although dependencies exist logically between order items, they are managed by order components. In other words, if any item in a component has a dependency, the component as a whole cannot be started until the dependency is resolved. In the Order Management web client, order items include dependency IDs to indicate items whose dependencies are managed together. See *Order Management Web Client User's Guide* for more information.

Dependencies and Order Revision

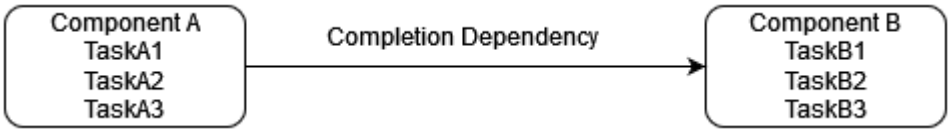
You can manage dependencies during amendment processing. For example, when you submit a revision order, OSM fulfills the order components based on the order of dependencies in the revised orchestration plan. OSM handles the dependency during the revision as follows:

Completion Dependency

For completion dependencies between the order components, if the completion dependency between predecessor and successor components is not removed in the revised order's orchestration plan, then the redoing of the successor component will wait until the compensation of the predecessor component is complete. The undoing of the predecessor component will also wait until the compensation of the successor component is complete.

The following image shows the completion dependency between component A and component B. For example, component B is dependent on component A. The redoing of TaskB1 in component B must wait until the redoing of TaskA1, TaskA2 and TaskA3 in component A is complete.

Figure 4-6 Completion Dependency Between Component A and Component B



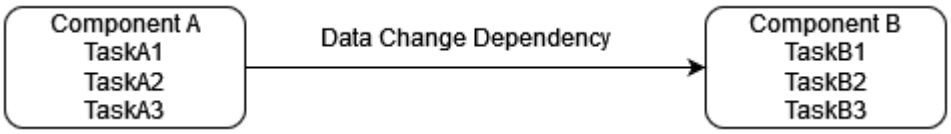
Data Change Dependency

For data change dependencies between the components, the redoing of the order components is run in the sequence of the dependencies of the revised orchestration plan. The undoing of the predecessor component will wait until the compensation of the successor component is complete. If a cartridge is built with its target OSM version set to 7.5.0 or newer, additional logic comes into play. The base order resolved data change dependencies are reevaluated during the amendment compensation processing.

When the data change dependency is met on reevaluation during compensation processing, that is, in the redo mode, the redoing of the successor component will start without waiting for the compensation of predecessor component completion. For cartridges built with an older version of OSM, the redoing of a successor component will wait until the redoing of the predecessor component is complete.

The following figure shows the data change dependency between component A and component B. If a cartridge is built with its target OSM version set to 7.5.0 or newer, the data change dependency is reevaluated during the redoing of the TaskA1, TaskA2 and TaskA3 in the component A until it is met. Once it is met, the redoing of TaskB1 in component B will start without waiting for compensation of the component A to be completed. If a cartridge is built with its target OSM version set to older than 7.5.0, the redoing of TaskB1 in component B must wait until the redoing of TaskA1, TaskA2 and TaskA3 in component A is completed.

Figure 4-7 Data Change Dependency Between Component A and Component B

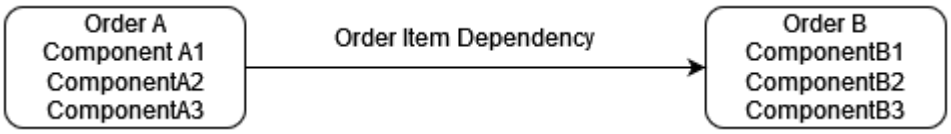


Order Item Dependency

For order item dependency between orders, given that the scope of compensation is always a single order, compensation of the succeeding order's components and tasks will happen independently from the predecessor order. On cancelling an order, OSM rejects cancellation of a predecessor, when there is an in-progress successor order.

The following figure shows the order item dependency between Order A and Order B. The component B1 of Order B is dependent on component A1 of Order A.

Figure 4-8 Order Item Dependency Between Order A and Order B

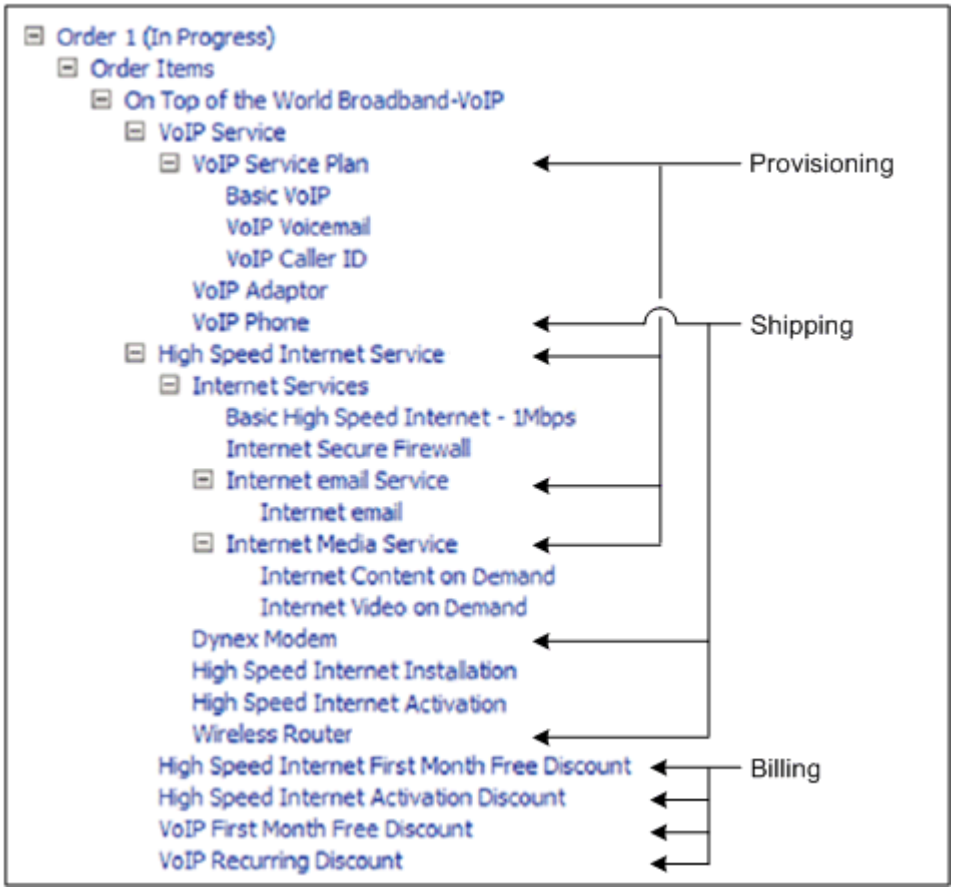


About Order Items and Order Components

As described in "How OSM Generates and Runs an Orchestration Plan," orchestration is the process of decomposing order items into order components, which then run based on their dependencies.

Because order items are hierarchical, a single parent order item can include child order items that can be decomposed into multiple function order components. Figure 4-9 shows the order items that can be decomposed into Provisioning, Shipping, and Billing order items.

Figure 4-9 Hierarchical Order Items



The offers, bundles, and products would first be decomposed into function order components for Provisioning, Shipping, and Billing. Therefore, even though all the child order items share a parent order item, they do not necessarily share function order components.

The provisioning order items might be further decomposed into system order components with the following order component IDs:

- Provisioning.Voip
- Provisioning.Email
- Provisioning.Media

The provisioning order items for VoIP might be decomposed into granularity order components with the following order component IDs:

- Provisioning.Voip.WholeOrder
- Provisioning.Voip.Bundle
- Provisioning.Voip.Offer

At runtime, OSM creates an order component ID based on the sum of all order components used in the decomposition. For example, the order component **Billing.BillingSystem.FixedBundle** represents:

- A Billing function order component
- A BillingSystem target system component
- A FixedBundle granularity component

The **Billing.BillingSystem.MobileBundle** order component represents:

- A Billing function order component
- A BillingSystem target system component
- A MobileBundle granularity component

Orchestration and COM, SOM, and TOM

As described in "About Customer Orders, Service Orders, and Technical Orders," you can use OSM in the COM, SOM, and TOM roles to run customer orders, service orders, and technical orders in a single service fulfillment process. Each type of order uses orchestration, but each type of order uses different order items.

In the COM role, OSM receives a sales order from the order-source system and creates a customer order. The order items in a customer order typically describe the products, bundles, and offers that customers have purchased; for example, Add Gold Broadband Bundle or Add Triple-Play Plus Offer. Services are generally identified generically, such as Add Email Service or Add Video Service.

Because a customer order works with order items that describe the products and bundles that customers purchase, OSM in the COM role typically runs executable order components that work directly with billing systems. (A customer who purchases a product needs to be billed for it.)

To manage provisioning order items, such as Email Service [Add] or Video Service [Add], a customer order runs provisioning order components that send data to OSM in the SOM role.

In the SOM role, the order items in the sales order describe customer-facing services; for example Broadband Internet Service [Add] or Video Service [Cancel]. The data in the order contains the provisioning data required to transform the customer-facing services into resource-facing services. For example, the provisioning data might include the customer's location, the bandwidth requirements, and so on.

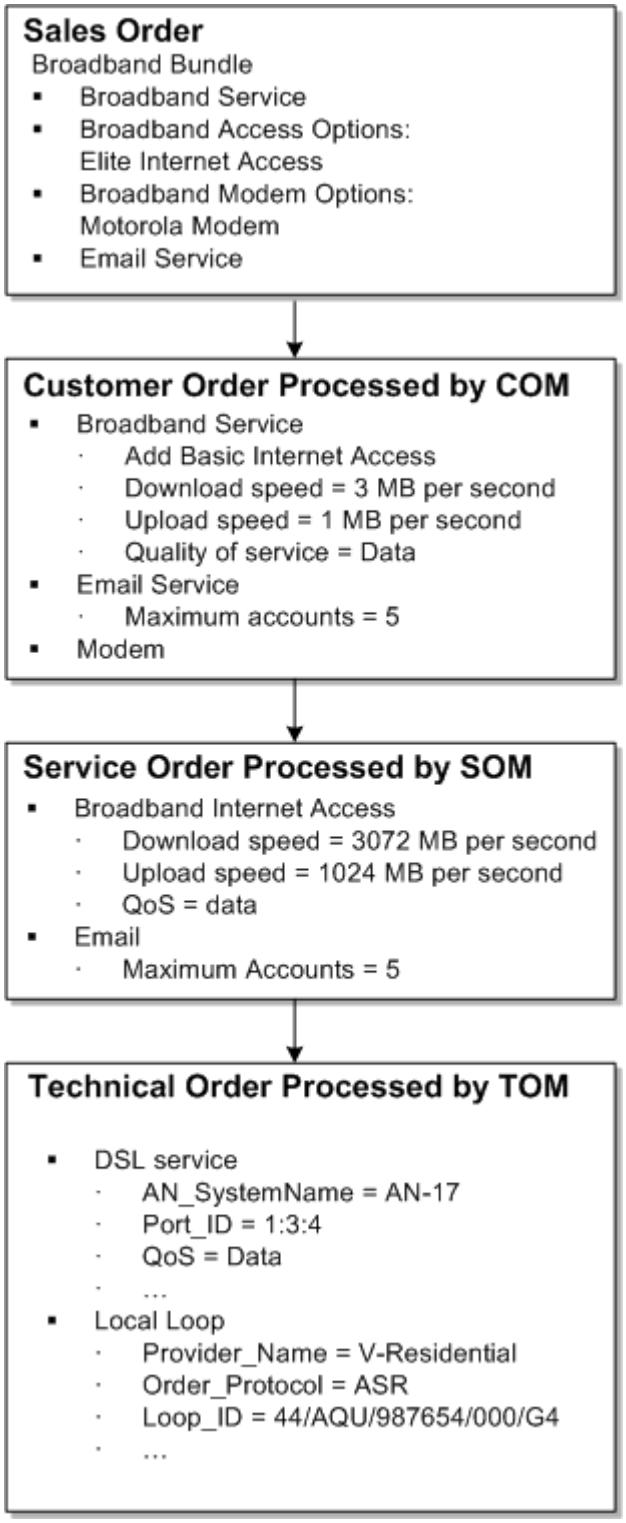
The executable order components in a service order interact with a service and resource management (SRM) system such as Oracle Communications Unified Inventory Management (UIM). These order components send the provisioning data that describes the service requirements to the SRM. The SRM system returns the data required to activate the services. The executable order components in the service order then send the data to OSM in the TOM role to create a technical order to activate services and ship equipment.

OSM in the TOM role processes a technical order to activate the resource-facing services and interact directly with shipping and workforce management systems. The order items describe resources and resource-facing services such as DSL Service [Add] or Firewall [Add].

The executable order components in a technical order interact with the activation and shipping systems. They send the service requirements, and receive confirmation that the services were activated and equipment shipped.

Figure 4-10 shows examples of the order items that are processed in COM, SOM, and TOM.

Figure 4-10 Order Items in COM, SOM, and TOM



About Order Item Transformation

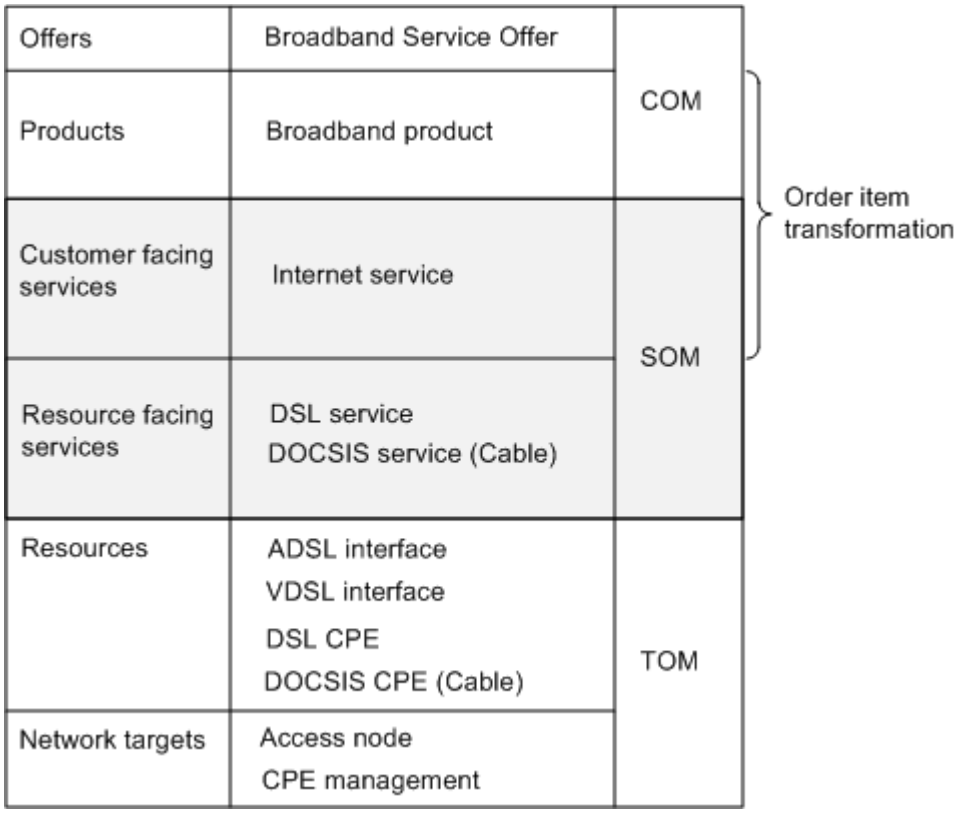
As described in "Orchestration and COM, SOM, and TOM," OSM in the COM role processes a customer order to transform order items for products into order items for customer-facing

services. You can use **order item transformation** to enable OSM to automatically identify the order items for customer-facing services. To configure order item transformation, you configure the following:

- The order items to transform from.
- The order items to transform to.
- The rules that govern which customer-facing service needs to be implemented.

Figure 4-11 shows how OSM in the COM role runs a customer order to calculate a service order. In this example, OSM can transform the Broadband product order items into the Internet service order items.

Figure 4-11 Order Items Transformed by OSM SOM



About the Design Studio Conceptual Model

The Design Studio conceptual model functionality helps you model an order fulfillment process by using the known aspects of your business requirements. Instead of using Design Studio to manually create entities for Oracle Communications ASAP, UIM, and OSM, you provide Design Studio with information about your products, services, and resources, and Design Studio creates entities required for the service fulfillment process.

The conceptual model takes as input such entities as:

- Customer-facing services
- Resource-facing services

- The actions required to fulfill services
- Resources on your network
- Products in your product catalog
- Customer locations

Conceptual model entities represent abstractions of services that are converted into application model entities. This conversion process is called **realization**. The conversion starts with an abstract conceptual model entity and creates a real application model entity. You include the realized application model entities in application projects, and then deploy the application projects that contain the realized entities to runtime environments.

The application entities that are realized are such entities as:

- UIM service configurations. These are the service configurations that OSM needs to design a service.
- UIM inventory resources. These are used by OSM to assign resources to services.

You can use the output from a conceptual model as a starting point to model your OSM runtime solution.

See *Design Studio Concepts* for more information about conceptual model projects. See *OSM Modeling Guide* for more information about using conceptual model entities to map order items to fulfillment patterns.

5

About Tasks and Processes

This chapter provides an overview of Oracle Communications Order and Service Management (OSM) tasks and processes. Before reading this chapter, read "Order and Service Management Overview " and "How OSM Processes Orders."

About Tasks and Processes

A **task** is an activity that must be carried out to complete an order; for example, if an order needs to verify that an ADSL service was activated, you could model a task named Verify ADSL Service. Tasks can be manual or automated.

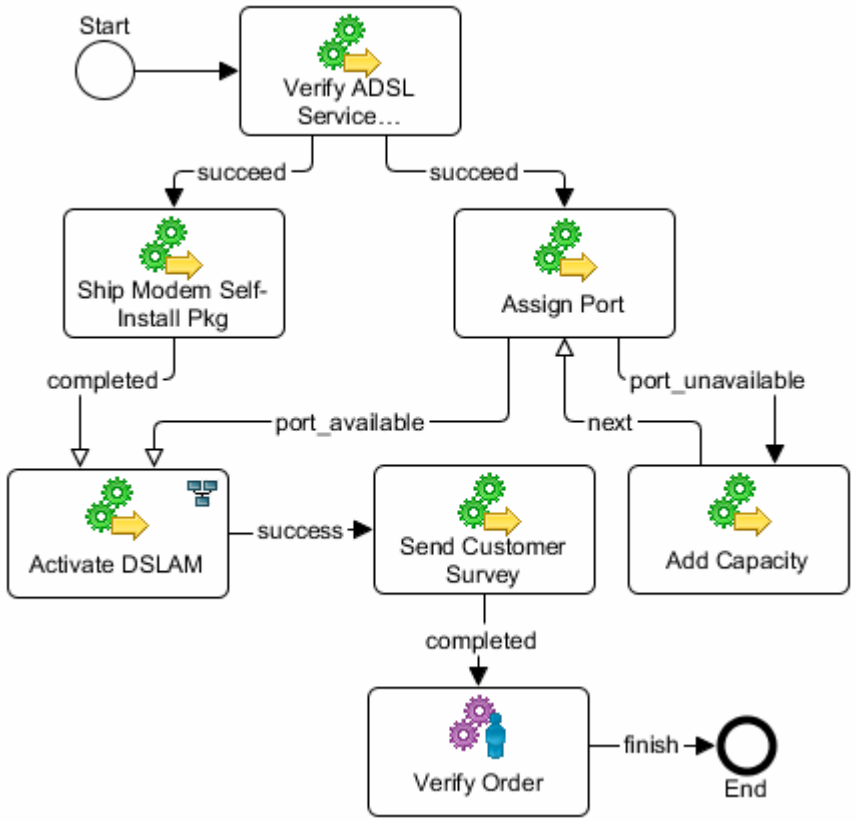
- **Manual tasks** must be processed by order management personnel, by using the Task web client. Manual tasks typically include tasks that cannot be automated, or tasks that require decision-making, when there are multiple choices for how to proceed with order processing.
- **Automated tasks** run automatically with no manual intervention. Automated tasks typically communicate with external systems by using the external system's API. For example, you could define an automated task called Verify Address. An automation plug-in can be configured to send order data to a third-party address verification system whenever an order reaches the Verify Address task. The third party returns an address confirmation to OSM, completing the task.

A **process** is a sequence of tasks and subprocesses that run consecutively or concurrently to fulfill all or part of an order. Processes enable you to break down the work required to run and fulfill an order into functional tasks, which can be distributed to various systems and order managers to be completed in a controlled manner.

In processes, you can control how the tasks are run. For example, you could create a rule that evaluates data and branches the process appropriately. Any number of processes can be defined in an order process, consisting of any number or combination of manual and automated tasks. You can also run subprocesses from a process. **Subprocesses** are processes that are launched from another process, as opposed to being launched from an order.

Figure 5-1 shows a process and its tasks, as shown in Design Studio:

Figure 5-1 Example of an OSM Process



This process manages the fulfillment of a request for an ADSL service:

1. The first task, **Verify ADSL Service**, is an automated task that verifies that the ADSL service exists. For example, the task might run a web service operation that reads a database to determine if the service is available at the specified address.
2. After verifying that the service is available, the process branches to two tasks that are independent and can run in parallel:
   a. The Ship Modem Self-Install Pkg task sends a shipping order to the hardware provider.
   b. The Assign Port task looks up a port in the inventory system and assigns it.

      If the port is available, the next task is Activate DSLAM. However, if the port is not available, the process transitions to the Add Capacity task, and then back to the Assign Port task.
3. After the Assign Port task is finished, the Activate DSLAM task can run. This task contains an OSM integration with a third-party activation system to activate the DSLAM.

   The Assign Port task is dependent on the completion of both the Ship Modem Self-Install Pkg task and the Activate DSLAM task. Therefore, even if the Ship Modem Self-Install Pkg task completes, the Activate DSLAM task cannot start until the Assign Port task is finished.
4. When the activation is complete, the next two tasks send the customer survey and require that an OSM user verifies the order to make sure it is complete. After these two tasks are completed, the order is complete.

Any of the tasks in this process can be configured as automated tasks. For example, the Assign Port task can be an automated task if there is an integration with the inventory system,

and the inventory system is able to respond to an automation plug-in sender requesting a port number with a response that assigns the port number for the service.

When you create tasks in Design Studio, you define the data required by the task. A task typically contains a subset of the order data received on an incoming order. For example, the task for shipping a modem might require a customer name, phone number, and address but not the required internet bandwidth.

About Manual Tasks

Manual tasks are assigned to personnel who complete the work for these tasks in the OSM Task web client. Personnel can manage tasks by adding comments to the order, attaching documents, displaying the history of the order, and manually entering and saving order data required to complete the task.

To run manual tasks by using the OSM Task web client, an order manager works from a list of manual tasks called a **worklist**. To complete a task, an order manager typically enters data or reviews the data, and then indicates that the task is complete.

Figure 5-2 shows tasks displayed in the Task web client worklist. In this example, Assign Port and Ship Modem are manual tasks.

Figure 5-2 Tasks Displayed in the Task Web Client

Order ID	Source	Order State	Type	Priority	Process	Task	Execution Mode	State
412	Add ADSL	In Progress	Add ADSL	5	Add ADSL v3	Assign Port	Do	Received
412	Add ADSL	In Progress	Add ADSL	5	Add ADSL v3	Ship Modem	Do	Received

Task **states** track the progress of a task. For example, when a task begins, it is in the Received state until such time as an operator accepts the task. The task remains in the Accepted state until the operator complete the task.

When you model manual tasks in Design Studio and define the data required for the task, you can do the following:

- **Assign Roles**: You can assign roles to tasks which, when associated with OSM WebLogic user accounts as workgroups, limit who can receive, accept, and work on new tasks in the Task web client.
- **Define Notifications**: You can configure a task to trigger notification messages that appear in the Task web client to inform order managers of the progress of the order. Notifications can also trigger automation plug-ins that send status updates or jeopardy warnings to external systems or users.
- **Configure Behaviors**: You can use **behaviors** to manipulate data and to control how data is displayed in the Task web client. For example:
  - You can specify the maximum allowed number of characters for text string data.
  - You can specify the contents of a list displayed in the Task web client.

About Automated Tasks

Automated tasks require no manual intervention. Automated tasks handle internal interactions with external fulfillment system, such as billing systems, shipping systems, activation systems,

and other fulfillment systems. OSM processes typically include more automated tasks than manual tasks.

To create an automated task, you do the following:

- Model a task entity in Design Studio. An automated task entity includes many of the same elements as a manual task; for example, data required for the task, notifications to send, and task states.
- Create one or more automation plug-ins. Automation plug-ins can perform custom logic, send a message to an external system, or update OSM with data received from an external system. For example, you could define a task called Verify Address. An automation plug-in can be configured to send order data to a third-party address verification system whenever an order reaches the Verify Address task. Another plug-in can be used to receive the address verification. Most plug-ins use XQuery to find, filter, and transform data.

OSM uses the automation framework to run and manage plug-ins. The automation framework provides the primary interface for outbound and inbound operations that interact with external systems for automated order fulfillment. The automation framework also provides internal data processing for automated tasks within a process workflow. A plug-in can access task data and perform OSM functions such as completing a task.

An **activation task** is a type of automated task, designed specifically to interact with the Oracle Communications ASAP product and the Oracle Communications IP Service Activator product to activate services on your network.

Activation tasks include many of the same properties as other automated tasks; for example, you can assign permissions, define the task data, and configure notifications that trigger automation plug-ins. However, you also configure activation-specific data elements, such as how to map data sent to and received from ASAP or IP Service Activator.

About Task States

A **task state** determines the condition of a task in a process. Every task in OSM has a set of possible states that reflect the life cycle of the task. The required tasks are:

- **Received:** The task has been received by a workgroup and is waiting to be accepted.
- **Assigned:** The task has been assigned to a specific OSM user. Tasks that are in the assigned state cannot be worked on by other users.
- **Accepted:** An order manager has accepted the task and is working on it. Tasks that are in the Accepted state cannot be worked on by other users unless the state is returned to the Received state.
- **Completed:** The task has been completed by a user or an automation plug-in. A task that has been completed no longer appears in a user's worklist.

6

About Order Management Business Processes

This chapter provides an overview of how Oracle Communications Order and Service Management (OSM) works with business processes such as managing changes to orders and managing order delivery dates. Before reading this chapter, read "Order and Service Management Overview " and "How OSM Processes Orders."

About OSM and Order Management Business Processes

When you design your order fulfillment process, you can configure various entities in different ways to support your business processes. For example:

- You can configure orders and tasks to support changes to in-flight orders. See "About Making Changes to In-flight Orders."
- You can create dependencies between orders in case an in-flight order cannot be changed. See "About Follow-on Orders."
- You can configure OSM to be able to calculate when an order will be completed, and if necessary, when it should be started. See "About Determining Order Completion Dates."
- You can track the status of orders, tasks, and order items. See "About Order Status."
- You can configure notifications to alert order management personnel about orders that need attention. See "About Notifications."
- You can configure ways to deal with order and task failures. See "About Managing Order and Task Fallout."
- You can configure OSM in the COM, SOM, and TOM roles to minimize changes to the order fulfillment process when changes occur in your product offerings and network resources. See "Managing Changes in Your Business."

About Making Changes to In-flight Orders

If an order is in flight and a customer needs to change it, you can resubmit the order to OSM. OSM can roll back and change fulfillment actions as needed.

OSM manages changes to orders as follows:

1. An order is submitted to OSM. OSM begins processing the order.
2. The order is resubmitted to OSM with some of the data requirements changed. For example, the bandwidth requirement might change from 5Mbps to 13Mbps. You can also resubmit an order to correct a failed order.
3. OSM checks to see if the order is amendable. You specify whether an order is amendable when you model the order specification.
4. If the order is amendable, OSM looks for the original order by checking in-flight orders for a matching value in an order key. You configure the order key when you model the order specification. For example, you can specify to use the sales order number as the order

key. In that case, when OSM processes an order, it looks for an existing order that has the same sales order number and amends that order.

If an existing order is found, OSM needs to manage both the original order, called the **base order**, and the new version of the order, called the **revision order**.

5. OSM performs further checks on the base order to determine if the order is allowed to be amended. OSM does the following:
   - OSM checks to see if the base order is in a state that can be amended. Orders in the Not Started, Completed, or Aborted state cannot be amended. You can customize the allowed transitions to the amending order state by configuring the order lifecycle policy.
   - OSM checks to see if the base order has not passed the point of no return. The **point of no return** is the point in the processing of an order item after which order amendments are either impossible or too expensive to allow. See "About Point of No Return."
6. After determining if the base order requires changes, OSM begins the process of **compensation**. Compensation compares the requirements in a revision order to the requirements in the base order and determines the changes that need to be made. OSM creates a **compensation plan** to define the actions that need to be carried out to amend the base order.

   When you define data in OSM, you can flag data that might need to be changed as **significant**. OSM uses the significance of the data to determine if compensation is needed. Data significance allows you to optimize amendment processing in a way that compensation is considered only for changes to data that is marked as significant.
7. OSM handles the base order and the revision order as follows:
   - For the base order, OSM generates a new orchestration plan that includes the order components and their dependencies.
   - For the revision order, OSM transitions it to the Completed state because its only purpose was to revise the base order.
8. OSM processes the changes according to the compensation plan it calculated and recalculates the compensation plan needed after every change. The revised orchestration plan changes how order components are processed:
   - Order components with data that has changed as a result of the revision are redone.
   - Order components that have been processed but are no longer required in the revision are undone.
   - Order components that are inserted as new requirements are fulfilled.

   As order components are run, OSM runs tasks as needed. As with order components, tasks can be undone, redone, and so on.
9. All processing not related to compensation is suspended for an orchestration plan until compensation is complete. After compensation is complete, the order is restored from the Amending state to an In Progress state and normal processing continues.

About Submitting Multiple Revisions of an Order

In some cases, multiple revisions to a single order are submitted. Each revision is expected to be a new revision of the in-flight order, not a cumulative comparison of previous revisions. The latest amendment is assumed to be the most complete revision containing all of the changes from earlier revisions.

You can use versioning in revision orders to recognize the order of the revisions as OSM receives them. For example:

- If revisions are received out of sequence, OSM ensures that the latest revision is used. If a later revision is received while the current revision is being compensated, OSM completes the compensation for the current revision before processing the latest version. If a version is received that is earlier than the current revision being processed, the earlier version is ignored.
- If several revisions are received, OSM discards interim revisions and applies the latest revision because it represents the latest customer instructions for the order and is a complete copy of the base order.

About Point of No Return

In some cases, there may be a point in the order process after which it becomes impossible or undesirable to make changes to an order. This is called a point of no return.

There are two types of point of no return in OSM.

- A **hard point of no return** indicates that amendments to the order are either impossible or undesirable. In the case of a hard point of no return, a revision order is not possible. Instead, you can create a **follow-on order**.

> **Note:**
>
> A follow-on order is not a change to an in-flight order but is an alternative when revising the in-flight order is not possible. Follow-on orders are used to make changes to items on an order that have not yet been completed but are past the point of no return. OSM manages follow-on orders to ensure they do not run until the order items upon which they depend are completed. See "About Follow-on Orders."

- A **soft point of no return** indicates that order amendment processing is still possible, but there are consequences for the customer. For example, you can specify to bill a customer for an extra charge if the order is revised after the soft point of no return has been reached.

You can define multiple point-of-no-return milestones in an order's fulfillment flow. For example:

- For a fixed-line service, a point of no return after provisioning.
- For a broadband service, a point of no return after billing.

All soft and hard points of no return depend on the order lifecycle policy conditions that control whether orders can transition from the In Progress state to the Amending state.

About Follow-on Orders

A follow-on order is an order that can be started only after the completion of an order item in another order. You can configure follow-on orders for various reasons:

- If an order has passed its point of no return where a revision order is no longer possible, you can configure a follow-on order to modify an order after the order completes.
- In some cases, your order management process might be more efficient or faster if you use follow-on orders to manage fulfillment functions on different systems, implement load

balancing, and so on. Using follow-on orders provides another method of controlling when an order runs.

To configure a follow-on order, you create inter-order dependencies. When using inter-order dependencies, the blocking order item is in the base order, and the waiting order item is in the follow-on order. A typical scenario is:

1. A customer has ordered a broadband service.
2. The next day, while the order is still in-flight but past the point of no return, the customer requests a change to the service bandwidth.
3. Because a revision to the base order cannot be submitted, the customer service representative creates a follow-on order.
4. The follow-on order is submitted to OSM; however, it does not begin processing until the blocking order item in the base order has completed.

About Determining Order Completion Dates

An order received by OSM includes a **requested delivery date**. This is the date that the customer wants the order to be completed. OSM can calculate the expected completion date based on the time OSM expects to take to complete all of the tasks in the order.

If the requested delivery date is the same as or earlier than the expected completion date, OSM can start processing the order immediately. However, in some cases, the start date of an order should be delayed. For example, a customer might request that a new VoIP service be added at the beginning of the next month, when the customer's current service expires. In cases where the requested delivery date is later than the expected completion date, you can specify to start the order in the future.

In addition, there might be groups of order items within an order that need to be fulfilled at different times. For example, an order might contain three services, such as internet, IPTV, and VoIP. The internet and IPTV services might have an immediate requested delivery date, but the VoIP service might only be required at the end of the month, after the customer's current phone service plan has expired. In this case, you can enable OSM to calculate an order start time that would allow the service to be activated at the requested delivery date at the end of the month.

To calculate when an order should start so that it can meet a requested delivery date, OSM calculates the **expected duration** of the order. To enable OSM to calculate the expected duration, you assign processing duration values, for example:

- You can assign processing duration values to order components.
- You can assign processing duration values to tasks. These values can be used for calculating the processing duration for order components.
- You can assign requested start dates to order items.

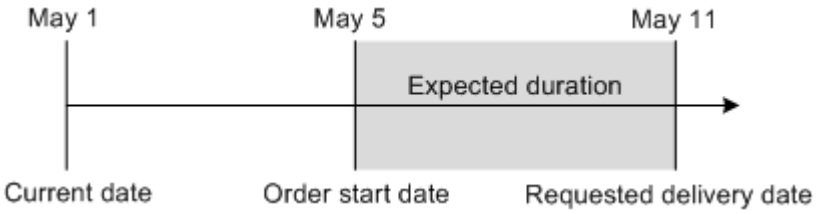
To calculate the expected duration for the order, OSM uses the expected duration of the order components and tasks in the order. Dependencies between order components are included in the calculation.

After the expected duration for the order has been calculated, OSM assigns an expected completion date to the order, based on the current date and the expected duration. For example, if the current date is May 1, and the expected duration is 5 days, the expected completion date is May 6.

If the expected completion date is later than the requested delivery date, OSM can set the order start date to a date in the future. For example, if the current date is May 1, the expected

duration is 5 days, and the requested delivery date is May 11, OSM sets the order start date to May 6. Figure 6-1 shows how a start date is determined for a future order.

Figure 6-1 Future Order Start Date



To track order completion dates, you can see the following fields in the Order Management web client:

- **Order Creation Date**: The date when the order is created in OSM.
- **Expected Order Start Date**: The date when the order is expected to start being processed.
- **Expected Order Completion Date**: The date when the order is expected to be completed.
- **Requested Order Delivery Date**: The date by which the customer requests the order to be delivered.
- **Expected Order Duration**: The amount of time the order is expected to take to complete processing.

About Order Status

Order management personnel can use the Order Management web client and the Task web client to track the status of an order, order components, order items, and individual tasks.

The customer service representative (CSR) who runs the customer relationship manager also needs to keep the customer informed about the order status. You can use **fulfillment states** or **processing states** to maintain a complete and detailed view of order item status and (with fulfillment states) order status. For example, a CSR might need to know if shipping has been completed for an order, but activation has not been completed.

OSM can send requests to many different systems for the same order item or order. Each request may have many responses that provide statuses.

Processing states are a predefined set of states that an order item can enter. OSM aggregates the values returned from external systems in each order component for the order item to determine the overall processing state of the affected order item. You can apply processing states based on values in response messages from external systems that OSM receives in automated task automation plug-ins or based on direct operator input in manual tasks. In addition, because order items can be arranged hierarchically, when a child order item processing state changes, OSM also evaluates whether the parent order item should change, all the way up the hierarchy.

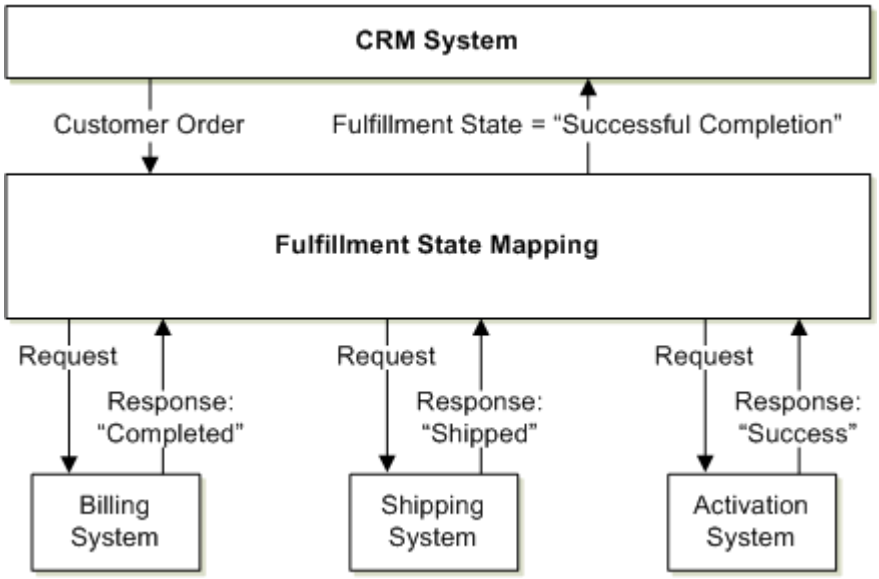
Fulfillment states allow you to send meaningful status values to an upstream system, which would not be able to parse all of the individual statuses returned by the external systems. You can use fulfillment states to manage system complexity in the following ways:

- Normalize status names. For example, to indicate success, one system might return OK, while another returns Complete. You can map both of these responses to the same value, for example Successful.
- Create statuses based on multiple statuses. For example, if one system returns the status Complete, and another system returns the status In Progress, you can create a status named Partially Complete.
- Define composite status values for order items, order components, and the order itself. For order items, you can change the status of a parent order item when the status of a child order item changes.

In addition to using fulfillment states to provide order item and order processing state information to upstream systems, you can also use them to restrict processing of order amendments from the upstream system. This functionality is provided by point-of-no-return processing, which uses fulfillment states. See "About Point of No Return" for more information.

Figure 6-2 depicts a fulfillment state scenario.

Figure 6-2 Fulfillment State in an Order



About Notifications

You can use notifications to alert users and external systems to events that occur in the order as it processes or to tell users that an action must be carried out.

There are two types of notifications:

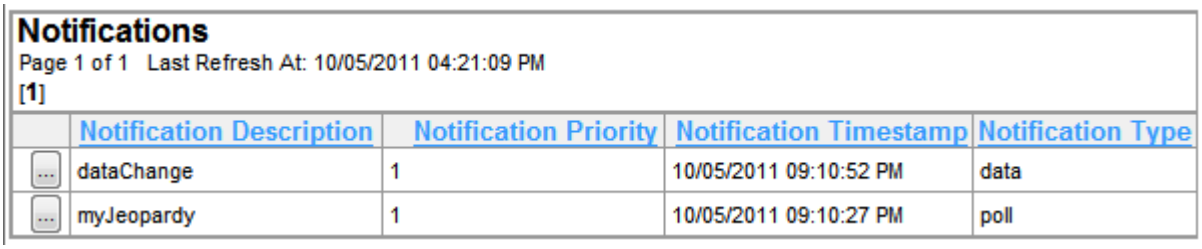
- Use **jeopardy notifications** when you want to alert users that an order or a task might have a problem. To trigger jeopardy notifications, OSM checks order and task conditions at specified intervals. If an action has not occurred as expected, OSM sends a notification. Jeopardy notifications are displayed in the Task web client Notifications page.
- Use **event notifications** to alert users of changes to an order based on its progress. Event notifications can be triggered by:
  - A change in task state

- A change in order status
- A change to order data

You typically configure notifications by using automation plug-ins, which send notification data to end users or systems.

Figure 6-3 shows notifications displayed in the Task web client. You can specify which workgroups can see the notifications.

Figure 6-3 Notifications Displayed in the Task Web Client



About Managing Fallout Exception

During the order processing journey, an individual order interacts with many external systems, and often the exchange involves more than a single request and response. Order fulfillment is a complex process and failures are not uncommon. When order processing becomes disrupted from its regular path, then a fallout exception management framework can provide a coordinated set of capabilities from the identification of fallout through to its resolution.

OSM's simplified fallout exception management framework changes the internal state of the order in case of fallout, thus eliminating the dependency on external systems. This approach acts as a net to stop further automation tasks on that particular order that is in fallout. This streamlined approach also ensures a smoother order management experience, minimizing disruptions.

Simplified fallout exception management framework includes:

- Automation plugin APIs that allow automation code to mark an order as having experienced fallout.
- For TMF cartridges using an OSM extended specification, custom states indicating fallout.
- User-friendly interface for easily identifying orders affected by fallout.
- Specialized fallout resolution actions accessible to the Fallout Specialist through the Task Web client user interface.

The simplified fallout exception management is available for both Freeform cartridges as well as TMF cartridges. However, for TMF cartridges, you must only use the simplified fallout exception management framework.

> **Note:**
>
> Simplified fallout exception management is supported for OSM cloud native deployments only.

Fallout Exception Scenarios

Fallout exception scenarios can be described as situations in which failures may either be temporary or require human intervention to determine whether to proceed or declare a failure. Temporary errors can often be resolved over time. Ideally, these scenarios should not disrupt the order processing flow, and work can resume once the necessary remedial action is taken by the user.

Typical fallout exception scenarios include:

- Failure in a network or system resource such as a network connectivity or a database failure.
- Complex exceptions returned from downstream systems such as inadequate inventory.
- Simple errors from downstream systems such as bad data.
- Other internal processing errors.

Example: A Fallout Exception Scenario

This section describes an example of a fallout exception scenario for better understanding.

In the inventory management system, the available stock of product X is inaccurately recorded. Consequently, more pre-orders are accepted than the actual available stock. As the fulfillment process begins, the system detects this discrepancy between the pre-orders and the available inventory.

The desired behavior of the overall solution is:

- **Accurate Inventory Management**: The overall solution should ensure precise recording and management of available stock for product X. This entails real-time tracking and updates to prevent overcommitting to pre-orders beyond the actual available stock.
- **Order Discrepancy Detection**: The system should be capable of detecting any discrepancies between pre-orders and the available inventory as soon as the fulfillment process commences, promptly identifying potential issues.
- **Fallout Exception Generation**: In cases of inventory shortfalls, the solution should automatically generate a fallout exception for each affected order, clearly marking them for attention and resolution.
- **Prevention of Overselling**: The system should halt further processing of affected orders to prevent the sale of products that are not in stock, thereby avoiding customer dissatisfaction and order-related problems.
- **Visibility and User Interface**: Users should have a user-friendly interface where they can view all orders in the fallout status, allowing for efficient monitoring and management.

The OSM fallout exception mechanism plays a crucial role within the overall scenario. It serves as the safety net for exceptional situations that cannot be addressed through standard automated processes. Here is how it contributes:

- **Alerting the System**: Automation would receive details of the inventory shortfall from the inventory system. In response, the automated code would raise a fallout exception with details about the issue.
- **Error Resolution**: OSM fallout serves as a critical component in addressing issues arising from inaccurate inventory recording. It takes on the role of resolving the fallout exceptions by identifying them as inventory shortfalls as designed by a cartridge developer.
- **Preventing Further Processing**: OSM provides a dashboard for fallout orders which can be monitored by the fallout specialist to identify all fallouts that have occurred.

- **Retry Operation**: After the product X is restocked, the OSM Task Web Client offers the fallout resolution actions. In this example, the Retry operation would retry the task and as the product is available now, it will continue to process the remaining steps of order fulfillment process.
- **Order History Tracking**: Detailed order history records are maintained to show that a retry operation was performed, providing transparency in the process.
- **Persistent Exception Records**: Resolved fallout exception objects remain in the system, offering an ongoing reference point for tracking and audit purposes.

By generating a fallout exception in response to the inventory issue, OSM fallout exception management plays a pivotal role in preventing customer disappointment. The intervention of a Fallout Specialist ensures that each affected customer receives a fair and satisfactory resolution, maintaining the service provider's reputation for delivering excellent customer service even when faced with unexpected challenges. This approach highlights the significance of effective order management and the responsiveness of the system to maintain customer satisfaction.

Fallout Exception Lifecycle

The lifecycle of a fallout exception includes:

- Fallout exception is raised
- Task processing is stopped
- Fallout is remediated through actions in the Task Web client application
- Fallout exception is cleared
- Normal task processing continues

Fallout Reporting
OSM provides a fallout reporting feature within the Automation API in conjunction with the OSM fallout feature. This is intended to allow plug-ins to report the occurrence of fallout at the discretion of the OSM Automation user (plug-in).

- There will be a 'Fallout Context' interface for automation plugins to report 'fallout exceptions''. Cartridge developers need to be aware of this interface at the time of cartridge development. Fallout should be reported by a plugin when the cartridge developer cannot handle the error any further. See the sections about fallout management in *OSM Modeling Guide* and *OSM Developer's Guide*.
- Automation triggers a fallout exception when it encounters an error that it cannot handle.
- In case of TMF orders, TMF order state transitions to *OrderState*.**fallout**.
- Simplified fallout exception is visible in the Fallout Orders dashboard and the Task Web Client UI shows fallout order details.

Fallout Orders Dashboard
The Order Operations and Fallout Management user interface provides users with multiple capabilities and is equipped with robust filtering mechanisms that provide detailed error insights and track overdue orders. Access a comprehensive graphical overview page to get a holistic view of existing fallouts. Dive deep into error details to gain more insight into a specific case. Filtering based on OrderId and customer name makes it easy for users to filter and gain insights.

Selecting the **Actions** option in the Order Operations and Fallout Management user interface navigates the user to the Process History page in the Task Web client user interface. From

here, users can navigate to the worklist where the fallout exception actions are available in the list of shortcut menu options for an order.

Remedial Actions for Fallout Exceptions
In the Worklist page in the Task Web client user interface, the shortcut menu options include a "Fallout Exception Actions" submenu. The following options are available in the submenu:

- **Manually Complete Task**: Users can access the task editor to manually input data and complete tasks, which means fallout exception is resolved automatically.
- **Retry Task**: This leads to the fallout exception being tagged with the action code "retry." The task is then retried and the fallout exception is cleared. If the retry is successful, the task completes. If the retry runs into problems, the cartridge code can raise a new fallout exception.
- **Fail Order**: This results in the entire OSM order failing and triggers an update to the final TMF state in case of TMF orders. The fallout exception is resolved automatically, marked with the action code "fail."
- **Abort Order**: Abort tells OSM to simply stop working on this order without triggering any further work.

For more detailed information on the actions that can be performed on Fallout Orders, refer to the *OSM Task Web Client User's Guide*.

About Managing Order and Task Fallout

Order fallout occurs when an order fails during processing and cannot continue processing. Task fallout occurs when a task fails during process and can only continue processing in a fallout execution mode. **Fallout management** is the ability to resolve fallout and enable an order or task to continue processing normally. You can model automated fallout management, which corrects errors by compensation or by running automation plug-ins in a fallout execution mode, or you can model manual fallout management, which supports manual intervention to correct errors.

For any order specification, you can define order fallout definitions. Fallout definitions enable you to identify specific order data that can cause a fallout scenario. For example, it might be common for a task that activates a port to return an error that the port is already in use. The fallout definition can identify the port ID as the data that needs correcting. This allows OSM to undo the resource assignment task in the inventory system, so the task can be redone and the port ID corrected. The order can then resume processing with the corrected data.

OSM can manage fallout that occurs both internally during OSM processing, such as errors in internal data, and as the result of an error returned by an external fulfillment system. The most common fallout scenarios are:

- Failure in a downstream system; for example, a failure in an activation system.

  Scenarios include:

  - The data was received, but was missing or incorrect and could not be processed. When a downstream system detects missing or incorrect data received from OSM, it returns an error, which in turn fails the order.
  - An internal error unrelated to the data occurred.
- Failure in a network or system resource; for example, a network connectivity failure or a database failure.
- Failure when an order is created in OSM; for example, recognition or validation errors. If OSM receives a corrupted order from an external system, it accepts the order but immediately places it in Failed state.

- Failure in runtime OSM processing; for example, an unresolved dependency that prevents an order from being processed.

You can track the progress of order items including fallout scenarios that impact order items in the Order Management web client by enabling tasks to associate status values included in response message from external fulfillment systems to OSM order item processing states. You can classify these response messages into order item processing states that fall into the normal, warning, or failure categories that you can track in the Order Management web client.

When a task receives a failure message from an external system, you can configure the task to run in a fallout execution mode such as Do in Fallout in contrast to a normal execution mode such as Do. For example, you may configure an automated task to run special automation plug-ins that only run when the task transitions to a fallout execution mode.

You can retry or resolve a failed task in the Task web client, or you can retry or resolve all failed tasks on an order or within an order component in the Order Management web client. Retrying a task returns the task to a normal execution mode in the received state. Resolving a task returns the task to a normal execution mode in the state it had been in when the failure occurred.

Managing Changes in Your Business

As a service provider, you need to manage changes in your service fulfillment implementation, for example, changes to product offerings, service configurations, and network resources.

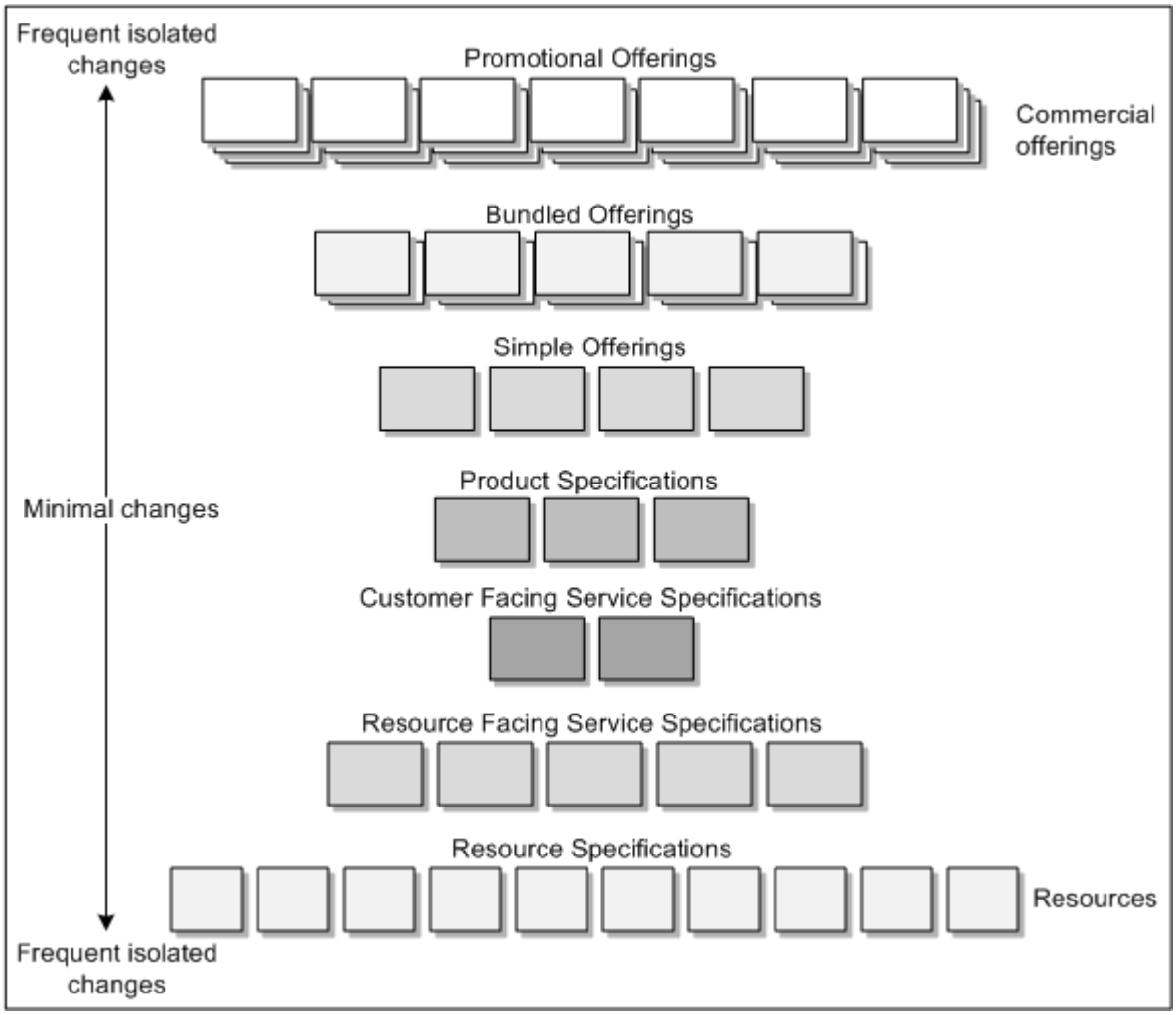
By running OSM in the COM, SOM, and TOM roles, you can decouple your product offerings from your network resources, and minimize the changes required in order processing. For example:

- Changes to product offerings do not change how resource-facing services are configured, and do not change how technical orders are processed. A change in how a product is packaged has no effect on how OSM manages the activation of the services in the offer.
- Changes to how you implement your services on the network typically do not change your product offerings, unless you add resources for a new type of service. For example, you might make equipment-level changes to your network that require changes to technical orders. However, those changes do not affect how a customer order is processed.

Figure 6-4 shows how frequent changes occur at both ends of the process; to commercial offerings and to network resources. Changes occur less frequently in the middle of the process, where services are designed according to predefined models, and resources are assigned to the services.

In addition, whereas there are many commercial offerings and many network resources, there are considerably fewer service designs. A single service design can be used by many different commercial offerings and resources.

Figure 6-4 How Changes Occur in an Order Fulfillment Process



Decoupling commercial offerings from network resources is important because system changes typically occur at both ends of the order fulfillment process. For example, when managing product offerings, you might introduce a new pricing for an existing service. When managing network resources, you might add more circuit capacity. These changes should have only minimal changes to an isolated part of the order fulfillment process.

The following are some typical scenarios that require changes to the order fulfillment process:

- **Change product offerings.** Examples include: introduce a new pricing for an existing service, introduce a new product in an existing product family, introduce a new bundle of existing services. In this case, changes are usually required only in customer orders. No changes need to be made in the SOM and TOM processing.
- **Upgrade network elements to a new vendor.** This requires new activation tasks to be able to send commands to the new network elements. Because there are no changes to the services offered, no changes need to be made to the order management process or to customer-facing services. The COM, SOM, and TOM processes remain the same. No changes are needed in product offerings or in the product specifications defined in OSM.
- **Introduce a variant to an existing technology.** For example, you might introduce a VDSL option when ADSL is already available. In this case, you probably need to make

changes to the definitions of resource-facing services, add resources to inventory, change some service activation tasks carried out by technical order management, and create new activation tasks. COM and SOM remain the same. No changes are needed in product offerings or in the product specifications defined in OSM.

7

About REST APIs and System Interaction (Cloud Native Only)

This chapter provides conceptual information about REST API support.

> **Note:**
>
> TMF REST APIs and System Interactions are supported for cloud native deployments only.

Overview of REST API Support via System Interaction

JMS has long been the dominant mechanism for the exchange of messages between OSM and the external applications involved in order fulfillment. With REST APIs becoming increasingly common, OSM cloud native now provides capabilities for automation plugins to easily integrate with systems that use REST. In the TMForum ODA ecosystem for instance, there are TMF REST APIs describing all interactions traditionally involved in order fulfillment such as Partner Ordering, Shipping, Billing, Inventory, and Activation. A specific application instance may expose one of the TMF REST APIs or may offer a custom REST API - both are equally supported in OSM.

OSM cloud native offers a logical way to manage these interactions by modeling the API contract as an OpenAPI document, which is the System Interaction.

Terminology

The following table lists out common terminology that you need to be familiar with:

Table 7-1 Terminology

Name	Type	Description
Target System	Object	The logical name for an external system.
System Interaction	Object	This is an auxiliary resource in OSM that guides behaviour when processing an order fulfillment step that requires interaction with a Target System.

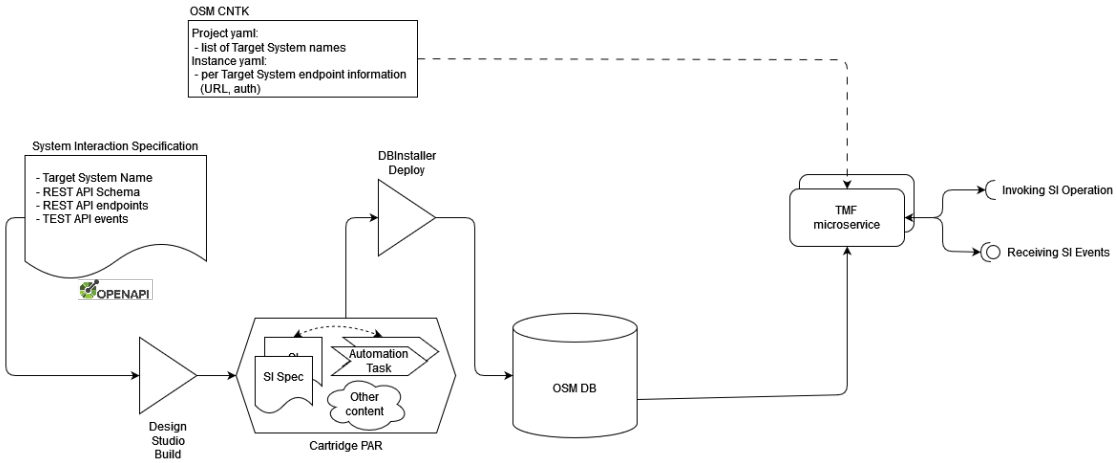
Table 7-1 (Cont.) Terminology

Name	Type	Description
System Interaction API	API	This is the API presented by the external target system. The order fulfillment step for OSM needs to communicate with this API. The interaction might be as a client (to invoke operations) and as a server (to process asynchronous event responses). In UML, the client portion of this is termed a "Required Interface" and in TMForum Component, a "Dependent API".
System Interaction Specification	Specification	A System Interaction expressed as an OpenAPI, its API and its semantics.

System Interaction Specifications

When an OpenAPI document describing the REST APIs of an external system is imported into OSM, it is referred to as a System Interaction Specification. A System Interaction specification would generally be created for each functional interaction with an external system. The specifications are carried in multiple OSM cartridges - either TMF or Freeform. When the cartridges are deployed, the system interaction specification is persisted to the OSM database schema. From here, the System Interaction specification is accessed by the OSM Gateway microservice container. OSM Gateway uses this information for routing incoming and outgoing messages between OSM and the external system as well as handling JSON-XML payload transformation according to the schema.

Figure 7-1 System Interaction Specifications



Expectations

This section highlights the expectations for the cartridge developer, administrator, and the OSM gateway.

External Application

- You must provide an OpenAPI to OSM cartridge developers that captures the semantics of the REST APIs for the system, Operations, HTTP headers, path parameters, HTTP response codes, schema, server url and version and so on.
- It ensures that only a single version is referenced inside the OpenAPI document. The **info:version** and version contained within the **server:url** must match.
- A condition of any integration using System Interaction, is that external systems must honor the HTTP header used by OSM for correlation. External system async responses must echo back the HTTP header **X-Correlation-ID** that was sent on the request either as a HTTP header **X-Correlation-ID** or in event payload field **correlationId**.
- Every operation in the OpenAPI document must contain an **operationId** field with a unique value to identifying the operation.
- For an external system that emits events and notifications, the OpenAPI document that is provided to OSM cartridge developers must contain the path that OSM cloud native is expected to expose via its listener.

Cartridge Developer

- Within each Order Component editor in Design Studio that needs to interact with a given Target System, import the OpenAPI specification for that Target System and provide a logical Target System name.
- For each automation task in that Order Component, designates the task supports "System Interaction" (rather than the default "Direct JMS").
- Writes sender automation plugin(s) to generate the payload (in XML) compliant with the API schema in the SI specification and identifies the REST operation to invoke for each.
- Writes sender automation plugins(s) to set HTTP headers and path parameters if necessary.
- Writes receiver automation plugin(s) to process the synchronous responses (HTTP codes and defined payloads) as XML.
- Optionally, writes receiver automation plugin(s) to process event payloads as XML.
- Provides a list of logical Target System names to the deployer or administrator.

Deployer (Administrator)

- Ensures that all Target System names needed by a cartridge are listed in the project specification.
- For each Target System logical name, identifies the actual target system for this particular instance.
- For each Target System logical name, creates or updates the required configuration in the instance specification. At a minimum, this is the URL of the external system and will very often include some authentication details linked from a secret.
- Ensures all required secrets are created as referenced in the target system configuration in the instance specification.
- Creates or upgrades the OSM cloud native instance.
- Ensures all external applications that act as a System Interaction Target System are properly configured with the OSM cloud native access URL and credentials.

OSM Gateway

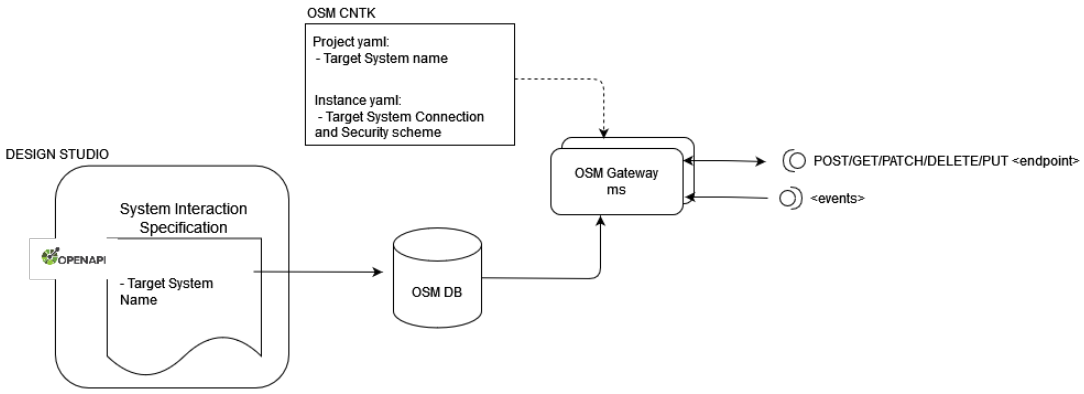
- Validates payload generated by sender automation plugin against SI schema, converts to JSON and uses this as payload to invoke the identified operation on the external system.
- Handles transient communication errors with delays and retries.
- Passes on hard failures and unresolvable transient failures back to the automation plugin.
- Validates synchronous REST response against the SI spec, converts payload to XML and passes it on to the receiver automation plugin.
- If automation plugins have registered for event notifications then OSM Gateway listens for incoming events, validates payloads against the SI specification, converts to XML and passes them on to the external receiver automation plugin.

Target System

The System Interaction Specification must define a Target System. This should be a logical name for the target system and should describe the function of the target system rather than its specific location. For example, a logical name of "Wireless-Activation" would be appropriate, as opposed to "Test-ASAP", as the latter pins it down to a specific system. This is important to allow flexibility between cartridge design and solution deployment design. The cartridge developer can freely reference a logical target system, leaving it up to the deployment scripts and configuration to allocate an actual target for that logical system in the form of a specific URL, authentication and so on. This target system is done via a two part configuration:

- **Design Configuration:** In Design Studio, OSM Order Components have a System Interaction tab which provides for entry of the target system name.
- **Deployment Configuration:** The logical target system name referenced in a cartridge must be defined in the actual CNTK deployment scripts. The CNTK holds the logical target system name in the project specification while the connection details like URL and authentication are defined in the instance specification. At runtime, OSM Gateway will resolve the logical target system name for a given System Interaction Specification by reading deployment artifacts created by the CNTK during instance creation.

Figure 7-2 Deployment Configuration



About Modeling Fulfilment Using System Interaction Specifications

The System Interaction feature is available for both Freeform and TMF cartridges. Refer to the "Modeling External REST Interactions using System Interaction" in the *OSM Modeling Guide* for further details.

8

About TMF Orders (Cloud Native Only)

This chapter provides conceptual information about TMF ordering support.

> **Note:**
>
> TMF orders are supported for OSM cloud native deployments only.

Introduction

The TM Forum defines and maintains a set of specifications intended to represent key objects and operations in the OSS and BSS space of a communications provider. The traditional order management roles that OSM plays are covered by two of these TMF specifications:

- **TMF622 Product Ordering Management**: This specification defines the Product Order entity and describes operations that can be performed on it. It is intended to cover the COM order fulfillment layer. Its input is an order containing ordered products, bundles, discounts, and so on. Fulfillment of the product order involves handling all operations that act on the ordered products, like partner ordering and shipping. Fulfillment includes transforming the product order into one or more service orders (operations on Customer Facing Services) directed towards a downstream fulfillment system.
- **TMF641 Service Ordering Management**: This specification defines the Service Order entity and describes operations that can be performed on it. It is intended to cover the SOM order fulfillment layer. Its input is an order containing service entities with associated actions. Fulfillment typically involves interacting with service inventory and network activation, and may also include workforce management and shipping.

OSM cloud native includes a framework that can transparently provide the capabilities defined in either TMF specification. At a high level, this includes exposing operational endpoints, automated TMF state calculation and reporting, along with new design time tooling that automatically translates the TMF specification schema components into Complex Data Type (CDT) structures in an OSM cartridge. This new TMF framework helps solution designers create and maintain a cartridge that provides COM or SOM order fulfillment while letting OSM transparently provide compliance with the TMF specification.

About Standards

TMForum

- TMForum created the TMF 630 **REST API Design Guidelines** to help applications extend the APIs defined by TMF. The OSM extensions align with the documented guidance.
- TMForum provides a means for vendors to "verify the successful implementation of Open APIs in commercial products and real-world deployments" by way of a TMF compliance toolkit. Oracle achieved a successful status on the OSM implementation of both the TMF 622 (v4.0.0) and TMF 641 (v4.1.0) specifications.

For more information, go to: http://www.tmforum.org.

OpenAPI Initiative

The OpenAPI Initiative documents patterns that should be used when extending the schema object. OSM 622 and 641 extensions in this area align with the documented guidance. Parsing technology is used to validate that the structure and syntax of imported TMF specifications at design time aligns with OpenAPI Initiative v3.0.1.

For more information, go to: https://www.openapis.org/.

Terminology

The following types of terms are introduced and contextualized:

- Object: This is an object that lives in OSM and is managed by OSM.
- API: This is the interface OSM presents in support of the Object.
- Specification: This is the description of the Object, the API, and its semantics; OSM uses this to model, operate, and present the Object.

The following table describes the terminology.

Name	Type	Description
Hosted Order	Object	This is the main resource of OSM - each instance represents an order and OSM executes order fulfillment for it, and presents it to the outside world.
Hosted Order API	API	This is how the hosted order is presented to the outside world. TMForum examples for this are TMF622 and TMF641. In UML, this is termed as "Provided Interface" and in TMFC (TMForum Component), as "Exposed API".
Hosted Order Specification	Specification	The Hosted Order expressed as an OpenAPI, its API and its semantics.
TMF cartridge	set of OSM cartridges	A TMF Cartridge is built around exactly one Hosted Order Specification, and provides the fulfillment logic for that order type.
Freeform cartridge	set of OSM cartridges	Freeform cartridges include all cartridges prior to 7.5.0 as well as non-TMF cartridges in 7.5.0.
Hosted Order Specification	Specification	The Hosted Order expressed as an OpenAPI, its API and its semantics. Used in a TMF cartridge.
System Interaction Specification	Specification	The REST API of an external system, expressed as an OpenAPI. Used in any type of cartridge - TMF or freeform.

Overview of TMF in OSM

When one of the two supported TMF ordering specifications is incorporated into the OSM TMF support framework, it is referred to as a Hosted Order Specification. The hosted order specification is carried in one or more TMF cartridges.

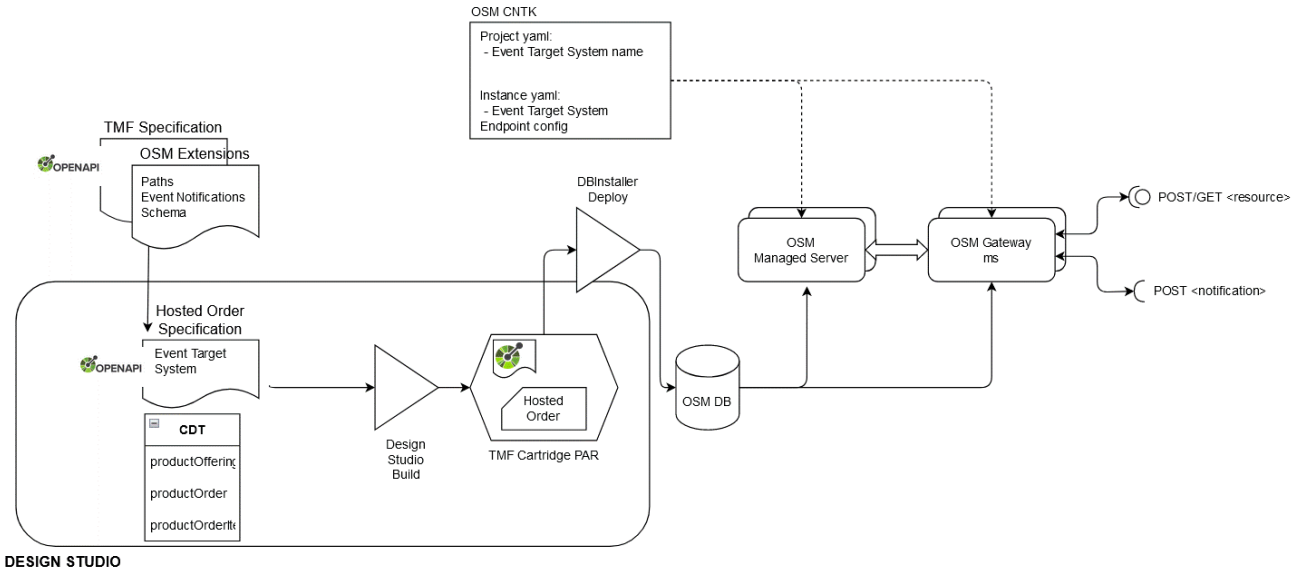
When the cartridges are deployed, the hosted specification is persisted to the OSM DB schema. From here, the Hosted Order Specification is accessed by the OSM Managed Server containers and the OSM Gateway microservice containers. The Managed Servers use this information to enable TMF-related capabilities when an incoming order is processed by this

cartridge. These capabilities include the automatic generation of TMF notifications (events) and the ongoing calculation of TMF state.

The TMF microservice containers also use the Hosted Order Specification information to expose the correct REST API to the outside world, and to translate between that API and OSM's internal components and models.

The following diagram illustrates the deployment of a Hosted Specification.

Figure 8-1 Deployment of Hosted Specification



About TMF Specifications

Configuring an OSM instance with TMF ordering support begins with the TMF specification. TMF specifications are based on the OpenAPI Specification which "*defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic*." Given the critical role of the specification itself, it is important to look inside and understand how different aspects are realized in OSM's TMF framework.

Paths Object

The paths object holds a path to individual endpoints as well as the operations permitted on an endpoint. When OSM cloud native hosts one of the TMF ordering specifications, the endpoints you see defined inside the TMF OpenAPI would be exposed by the OSM Gateway component. The **delete** and **patch** operations are unsupported by OSM as there are alternative mechanisms to achieve the same. See the Order Action Comparison table in "About the OSM Endpoints".

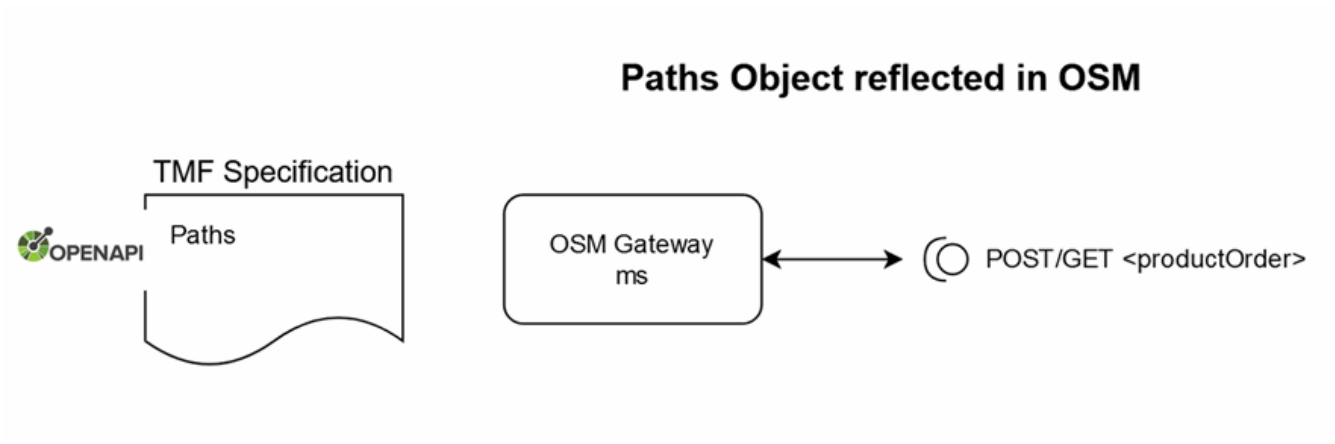
The following image shows the paths object in the specification file for TMF622 Product Order.

Figure 8-2 Paths in the specification file

```
paths:
  /productOrder:
    get:
    post:
  /productOrder/{id}:
    get:
    delete:
    patch:
  /cancelProductOrder:
    get:
    post:
  /cancelProductOrder/{id}:
    get:
```

The following diagram shows how the **paths** object is reflected in OSM.

Figure 8-3 Paths Object in OSM



Events Notifications

TMF specifications define not only the endpoints of an application but the event notifications that would travel northbound as well. These are identified in the OpenAPI document as endpoints with the keyword "listener" in the path. Publishing an event is done by posting the event to the listener address.

Event notifications are the vehicle for vital information about the resource, getting communicated to the external world. Lifecycle events such as creation, state change events and certain types of data updates are examples of notifications that OSM Gateway would emit.

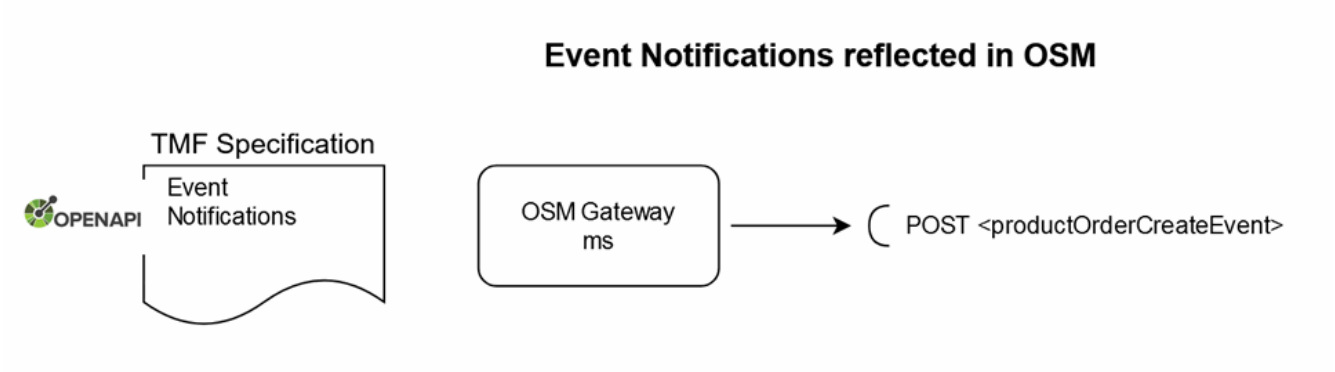
The following image shows the **listener** (Event Notifications) objects in the specification file for TMF622 Product Order.

Figure 8-4 Listeners in the specification file

```
/listener/productOrderCreateEvent:
/listener/productOrderAttributeValueChangeEvent:
/listener/productOrderDeleteEvent:
/listener/productOrderStateChangeEvent:
/listener/productOrderInformationRequiredEvent:
/listener/cancelProductOrderCreateEvent:
/listener/cancelProductOrderStateChangeEvent:
/listener/cancelProductOrderInformationRequiredEvent:
```

The following diagram shows how the Event Notifications are reflected in OSM.

Figure 8-5 Event Notifications in OSM



Schema

Each of the relevant TMF specifications includes a data model that defines the entities involved in support of the domain - product or service. With the TMF framework in OSM, the importing of a TMF specification results in the automatic creation of CDTs for all of the schemas defined in the OpenAPI.

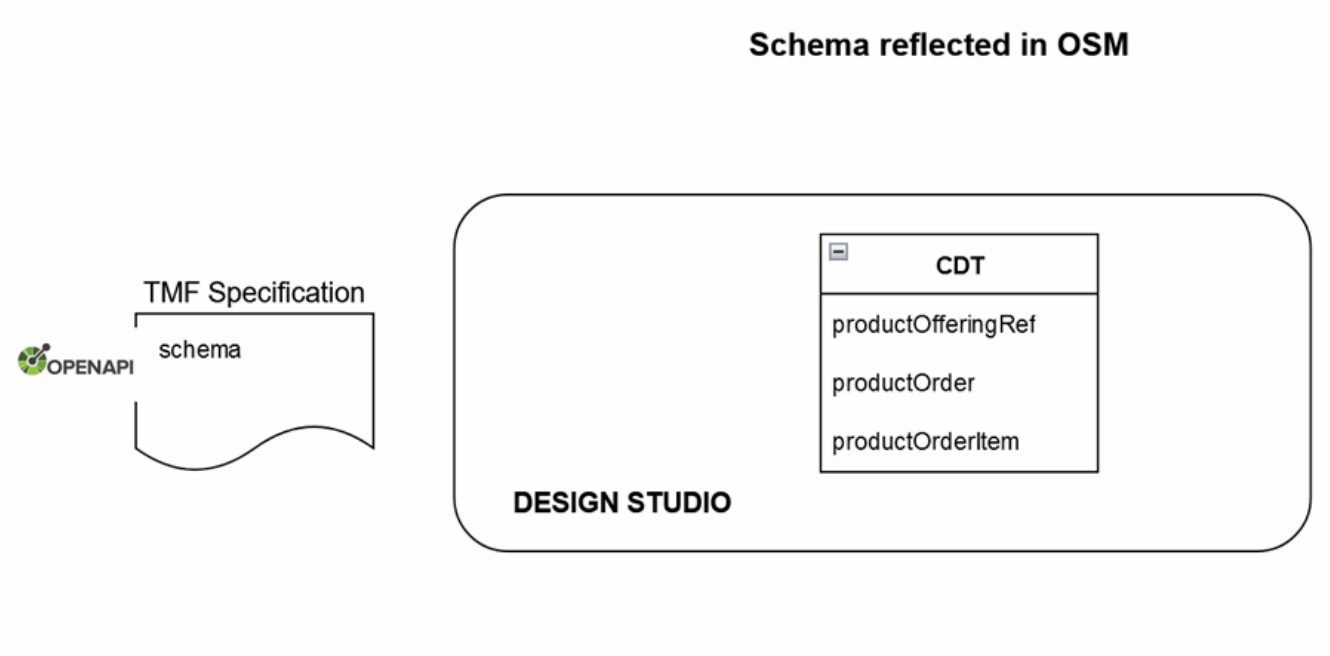
The following image shows the schema objects in the specification file for TMF622 Product Order.

Figure 8-6 Schema objects in TMF 622 Product Order

```
ProductOfferingRef:
ProductOrder:
ProductOrder Create:
ProductOrder Update:
ProductOrderItem:
ProductOrderItemStateType:
ProductOrderRef:
ProductOrderStateType:
ProductPrice:
ProductRef:
```

The following diagram shows how the schema is reflected in OSM.

Figure 8-7 Schema in OSM



About OSM Extensions

While the TMF 622 and 641 specifications provide endpoints for basic interactions with an order management system (post and get), they lack a comprehensive set of endpoints that align with OSM's advanced capabilities. In response, OSM has followed the guidelines from TMF 630 and made extensions to TMF's canonical 622 specification.

With the addition of OSM endpoints, upstream components have access to a full range of capabilities that includes creating new orders, submitting revisions, cancel orders, abort orders, suspend and resume orders, and fetch and search for orders using the full set of REST API endpoints described in the OSM extended specification. It is strongly recommended to use the specifications with OSM extensions for both Product Order and Service Order.

In the **TMFSchemas** sub-folder in the toolkit, OSM extensions have the suffix **-OSM** in the filename.

About the Specification Version

OSM's TMF framework relies on a 5-digit version scheme for hosted specifications.

- The first three digits are reserved for TMForum as TMF specifications are versioned with a three-number scheme.
- The fourth number is reserved for Oracle OSM to version extensions supported by the application itself.
- The fifth number is available for solution developers to version their additional extensions.

The OSM extended specification reflects this numbering scheme and has set the version number to 4.0.0.1.0. in the case of the 622 specification and to 4.1.0.1.0 in the case of the 641 specification.

About the OSM Endpoints

The **PATCH** and **DELETE** order level actions are not supported and removed from the OSM Extended specification:

- The ReviseProductOrder or ReviseServiceOrder task resource must be used instead of PATCH to leverage OSM's comprehensive amendment capabilities.
- OSM's powerful purge capabilities must be used instead of DELETE in order to provide uniform lifecycle management across orders at scale.

In addition to the CancelProductOrder and CancelServiceOrder task resource, OSM makes available additional task resources for complex requests that have a non-trivial lifecycle:

- SuspendProductOrder or SuspendServiceOrder: To request the suspension of an order; this is subject to the logic in the Order Lifecycle Policy as well as the configured grace period for in-progress tasks.
- ReviseProductOrder or ReviseServiceOrder: To request a revision to a running order; this is subject to TMF state validation, additional logic in the Order Lifecycle Policy (for solution-specific PONR) as well as the configured grace period for in-progress tasks.

OSM makes available additional operations on the order resource that are synchronous requests:

- ResumeProductOrder and ResumeServiceOrder: To resume a previously suspended order.
- AbortProductOrder and AbortServiceOrder: To halt an in-progress order immediately regardless of its current position in the fulfillment.

Order Action Comparison

The range of order actions in OSM prior to TMF support, map onto the range of actions implemented by OSM's new TMF framework.

The following table provides the mapping for TMF622 ProductOrder. The mapping for TMF641 ServiceOrder follows the same pattern.

Order Action	Canonical TMF Specification	OSM Extended Specification
Create	POST /productOrder	Same as canonical
Update	PATCH /productOrder/<id>	Not supported, use "Revise"
Delete	DELETE /productOrder/<id>	Not supported, use OSM purging mechanisms
Suspend	via PATCH	POST /suspendProductOrder
Resume	via PATCH	POST /resumeProductOrder/<id>
Cancel	POST /cancelProductOrder	Same as canonical
Abort	Not supported	POST /abortProductOrder/<id>
Revise	via PATCH	POST /reviseProductOrder
List	GET /productOrder	Same as canonical
Find	GET /productOrder/{id}	Same as canonical

Task Resources

A task resource is used to expose operations that are not easily represented as CRUD operations. TMF uses this pattern for CancelProductOrder and CancelServiceOrder. OSM has continued to use this pattern to model other operations such as revise, suspend, resume and abort.

Some task resources are persisted, resulting in an id which is separate and distinct from the main productOrder id (or serviceOrder id). Clients can use this id to discover the status of the task resource. (GET /reviseProductOrder/{id})

Abort and resume are simple task resources, not requiring persistence of the request. See the Order Processing Sequence Diagrams in this chapter to understand the interaction between the main resource and the task resource.

About OSM Event Notifications

The OSM event notifications work as follows:

- OSM emits additional creation and state-change events for the suspend and revise task resources. These notifications communicate the state of the task resource and NOT the main order resource.
- Event subscription mechanisms are not used (POST and DELETE on hub). Instead, an Event Target System configuration is used to define a single listener for each Hosted Order Specification. It is expected that this single listener will either consume the events itself, or serve it to a message broker (for example, Kafka).
- Where an operation is not supported (for example, DELETE), then any corresponding event notification is also unsupported (for example, ProductOrderDeleteEvent).
- OSM has scoped the AttributeValueChange events to only trigger on changes to order item state or order item characteristic changes.

Attribute Value Event Payload
The canonical TMF payload for attribute value change events includes a "fieldPath" element, intended to hold identifying information about the field on an order item that contains the change. There are two potential problems with this approach.

- This is a single field. If there are multiple changes to be communicated, then a convention would need to be introduced and communicated to the upstream system so it could effectively parse the data.
- Alternatively, if each event is restricted to a single change, then there is a risk of flooding the system with more events than are needed.

OSM has attempted to resolve these shortcomings in a robust, schema driven way. The fieldPath has been extended to be an array of compound structures that identifies the specific change including action (add, remove), JSON path to the change and the old value.

Additionally, OSM's framework does the following:

- OSM has restricted the changes that would trigger the AV event. Changes are limited to the order item state or the order item characteristic fields.
- OSM would include multiple updates in a single event. All updates, whether within a single line item or across multiple line items, would be included in an event, so long as they are all made within the same orderUpdate call to OSM core.

An example of FieldPath extension is as follows:

```
ProductOrderAttributeValueChangeEventOSM payload : {
"eventId":"c6fd423a-777b-4a7e-ae89-d4e12060461a",
```

```
"eventTime":"2023-10-19T13:26:25.587Z",
"eventType":"ProductOrderAttributeValueChangeEventOSM",

"fieldPath":[{"change":"add","path":"$..productOrderItem[?
(@.id=='0CX-1Z8QSX')].state","oldValue":""}],
"event":{"productOrder":{
"@type":"ProductOrderOSM",
"category":"salesOrder",
"description":"TMF622 OSM Product Order",
"externalId":"456850-osm",
.......

"productOrderItem":[{
"@type":"ProductOrderItem",
"id":"0CX-1Z8QSX",
"action":"add",
"state":"inProgress",
...............
```

About the OSM Schema

This section describes the OSM schema extensions.

Order State Extensions
TMF specifications define a set of order states that have been enhanced in the OSM extended specification. These states are final and cannot be altered. TMF state changes would be communicated upstream in the <resource>StateChangeEvent.

The TMF state is reflected inside the TMF object data served through the REST API (GET /productOrder or GET /serviceOrder). When using any of the OSM UIs (Web Services API, XML API, Task Web UI, Order Management Web client UI), the order state field shown reflects the OSM order state, and not the TMF state.

The following table lists the mappings and usage of the TMF order states:

Table 8-1 Mappings and Usage of the TMF Order States

TMF State	Usage	OSM Order State
acknowledged	Not used	Not applicable
rejected	Not used Rejected requests are handled through a synchronous HTTP response with appropriate HTTP response code	Not applicable
pending	Initial State. After order is created, but before any task has started. Initial State for future dated orders	Created
held	Not used	Not applicable
inProgress	Order is running	InProgress
cancelled	Order has finished being cancelled	Cancelled
completed	Order has finished execution with no failures	Completed
failed	Order has finished execution and all order lines have failed	Completed
partial	Order has finished execution with order lines having a mix of success and failure	Completed
assessingCancellation	Not used	InProgress

Table 8-1 (Cont.) Mappings and Usage of the TMF Order States

TMF State	Usage	OSM Order State
pendingCancellation	Order is running a cancellation request	Amending

The following table lists the mappings and usage of the state extensions by OSM:

Table 8-2 Mappings and Usage of the State Extensions by OSM

TMF State	Usage	OSM Order State
amending	Order is processing a revision request	Amending
amending.suspended	Order was processing a revision request but is currently suspended	Amending
amending.fallout	Order was processing a revision request but is now waiting on fallout exception resolution	Amending
amending.fallout.suspended	A revision request was waiting on fallout exception resolution but is now suspended	Amending
pendingCancellation.suspended	Order was running a cancellation request but is now suspended	Amending
pendingCancellation.fallout	Order was running a cancellation request but is now waiting on fallout exception resolution	Amending
pendingCancellation.fallout.suspended	A cancellation request was waiting on fallout exception resolution but is now suspended	Amending
inProgress.fallout	Order is running but is waiting on fallout exception resolution	InProgress
inProgress.suspended	Order was running but is currently suspended	InProgress
inProgress.fallout.suspended	An order waiting on fallout exception resolution but is now suspended	InProgress

Order Item State Extensions
TMF specifications define a set of order item states that have been enhanced in the OSM extended specification. These states are final and cannot be altered. Changes to order item state would be reflected in the <resource>AttributeValueChangeEvent.

The valid order item states include the list above except the following states involving suspension:

- inProgress.suspended
- inProgress.fallout.suspended
- amending.suspended
- amending.fallout.suspended
- pendingCancellation.suspended
- pendingCancellation.fallout.suspended

About Customer Name
When viewing fallout orders in the Order Operations and Fallout Management user interface, there is a column heading for Customer Name. This name would be populated from order data that you control at design-time. The hosted specification entity in Design Studio has a text field where a JSON path can be provided that points to a specific data field under the productOrder or serviceOrder.

- The import of a 622 specification defaults the value to **$.billingAccount.name**, but can be modified by the cartridge developer.
- The import of a canonical 641defaults the value to **$.externalReference.name** but can be modified by the cartridge developer.
- The 641 specification has no "natural" location in the schema where a customer name might be found. OSM has therefore extended the 641 schema to introduce a customer name field directly under the serviceOrder.
- If an OSM extended 641 specification is imported, then the default value will be **$.customerName**.

About Customer Extensions to TMF Specifications

The OSM specifications can be further extended to suit your specific implementation by:

- Adhering to OpenAPI guidelines when making schema extensions
- Adding to the schema of the primary object (ProductOrder or ServiceOrder) in terms of order data
- Keeping the set of Endpoints (paths) and operations and their contents unchanged
- Keeping the set of Event Notifications (listeners) and their contents unchanged
- Keeping the set of Task Resources and their contents unchanged
- Keeping the set of order state and order item state definitions unchanged (that is, preserve the set of states as-is)

If the OSM extended specification satisfies your requirements in terms of schema objects, then it can be used as-is. If however, schema additions need to be made, then you should make your additions on top of the OSM extensions. See the *OSM Modeling Guide* for some considerations when making schema updates.

About the Hosted Order Specification

Once the required schema extensions have been made to the specification (optional), the specification needs to be imported into Design Studio. Design Studio analyzes this document and creates multiple cartridge entities automatically. The most visible auto-created cartridge content is a Complex Data Type conforming to the schema in the Hosted Order Specification. A TMF order type can be created by the cartridge developer, using a CDT, as the model of the Product Order or Service Order. Other auto-created cartridge content includes the delivery fulfillment mode, internal roles for TMF support, and Order Lifecycle Policy.

The cartridge developer can then proceed with the standard cartridge development process, creating orchestration entities, processes, tasks, automation plugins, and so on. Building the cartridge results in Design Studio embedding the Hosted Order Specification into the par file along with all the other cartridge content.

About Hosting Expectations

When OSM cloud native receives an order request (create, get, cancel, suspend, resume, abort, revise) that gets handled by a TMF cartridge, the following expectations and services are in place with respect to its internal behaviour:

- A Complex Data Type (CDT) is created to match the structure and typing as per the schema for the Product Order or Service Order in the Hosted Order Specification. The cartridge developer must add this exactly once to the order template for a TMF order as a

root level element with the name ProductOrder or ServiceOrder, as applicable. This is the primary resource of the Hosted Order Specification.

- A cartridge developer can add more entries to the order template in order to facilitate the actual fulfillment of the order, outside of the primary resource. The primary resource and its underlying CDT can only be modified by using a modified Hosted Order Specification.
- While Order Recognition Rule and Order Lifecycle Policy continue to be managed by the cartridge developer, OSM samples are provided with the data transformation from incoming order request to the order template and will check cancellation and revision automatically against orders and order items in a final TMF state (completed, partial, or failed). These behaviours must be preserved in order to satisfy TMF compliance requirements.
- OSM manages the TMF state field, both at order level and at order item level. The times when cartridge developer needs to signal a state, they must use only an automation framework API to do so (for example, to signal TMF "failed" state for an order item).
- OSM automatically emits order state change events (ProductOrderStateChangeEvent, ServiceOrderStateChangeEvent) whenever there is a change to the order's TMF state.
- OSM automatically emits attribute value change events (ProductOrderAttributeValueChangeEvent, ServiceOrderAttributeValueChangeEvent) whenever specific data (order item state and anything in the order item characteristics) in the auto-generated order template changes. The most common of these would be to signal changes to the state of an order item.
- A cartridge developer can update the business content (primarily, order item characteristics) of the primary resource as part of the order fulfillment. Such updates allow new data from fulfillment to be reflected via GET calls on the primary resource, and where applicable, for events to be generated. For more details about updating TMF data, see *OSM Modeling Guide*.

About TMF Cartridges

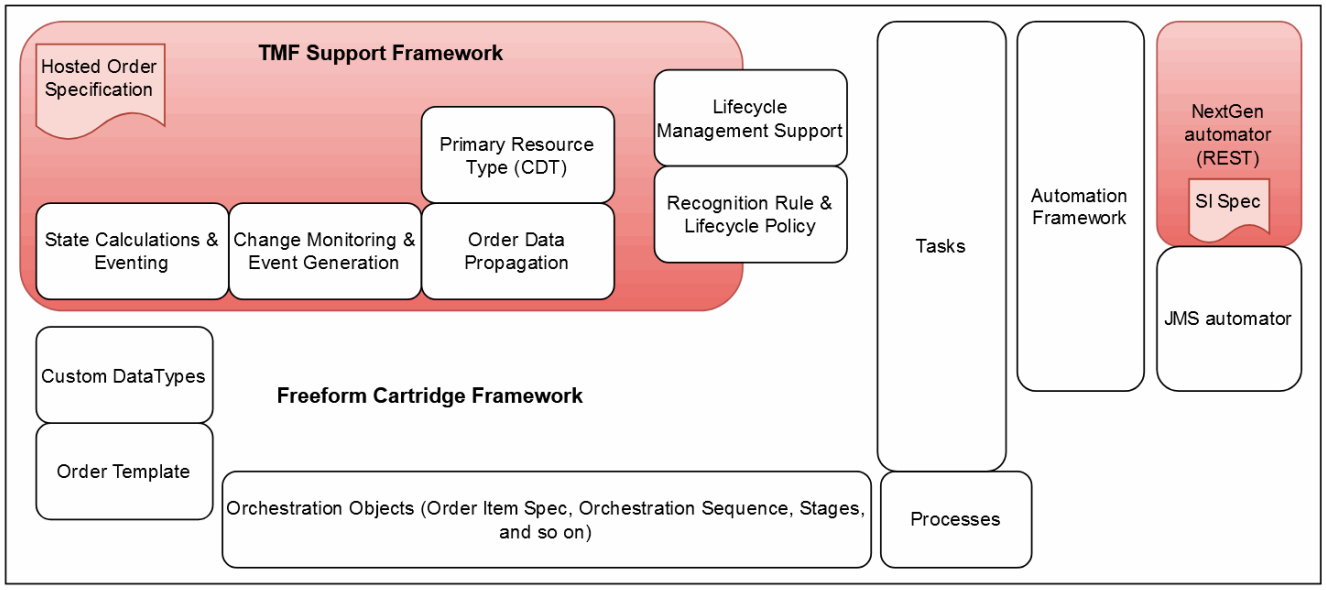
The **TMF Cartridge** type provides the most support to cartridge developers and system administrators when OSM operates on TMF orders. A TMF Cartridge is built around exactly one Hosted Order Specification, and provides the fulfillment logic for that order type (Product Order or Service Order). The TMF Cartridge is a solution cartridge, and consists of one or more component cartridges - all deployed as one unit.

The term **Freeform Cartridge** refers to the non-TMF cartridges. Freeform cartridges include all cartridges prior to 7.5.0 as well as non-TMF cartridges in 7.5.0. These cartridges are characterized by a fully open design-time experience, allowing complete flexibility in all of OSM's capabilities - order structure, order state behaviour, eventing and notifications, orchestration, automation, and so on.

> **Note:**
>
> It is important to note that these two cartridge types cannot be combined. If you have a Freeform cartridge, you cannot simply change the order type to TMF. You must start from scratch.

The following diagram illustrates the TMF support framework and the Freeform cartridge framework.

Figure 8-8 TMF Cartridge Framework



The Freeform Cartridge framework encompasses all of OSM's capabilities as exposed to the cartridge. However, when a TMF Cartridge is created from a Hosted Order Specification, some OSM capabilities are taken over by the TMF Support Framework.

This framework becomes responsible for:

- Generating the CDT to represent the primary object (order) in the Hosted Order Specification's schema
- Calculating TMF state and generating state change events
- Monitoring changes to the primary object and generating attribute-value change events
- Propagating data changes from orchestration components back into the primary object

In addition, the framework provides inputs into the Order Lifecycle Policy, as well as order lifecycle management mechanisms like fulfillment functions and required internal user roles. The cartridge developer has full access to the remaining portions of the Freeform Cartridge framework, like defining the order object, the various orchestration objects as well as the processes the tasks (manual and automated) require for fulfillment.

Both the Freeform Cartridge as well as the TMF Cartridge have access to the functionality to support REST interactions. Design Studio provides developers with the capability to send and receive REST requests, responses, and events with external systems (Target Systems) using a System Interaction Specification. See *OSM Concepts* for details. Also, refer to the *OSM Modeling Guide* for more information on working with TMF cartridges.

About Event Target System

While the Hosted Specification lists a set of notifications, there must also be configuration to define a listener address for the recipient of the events. This is done via a two-part configuration.

Design Configuration

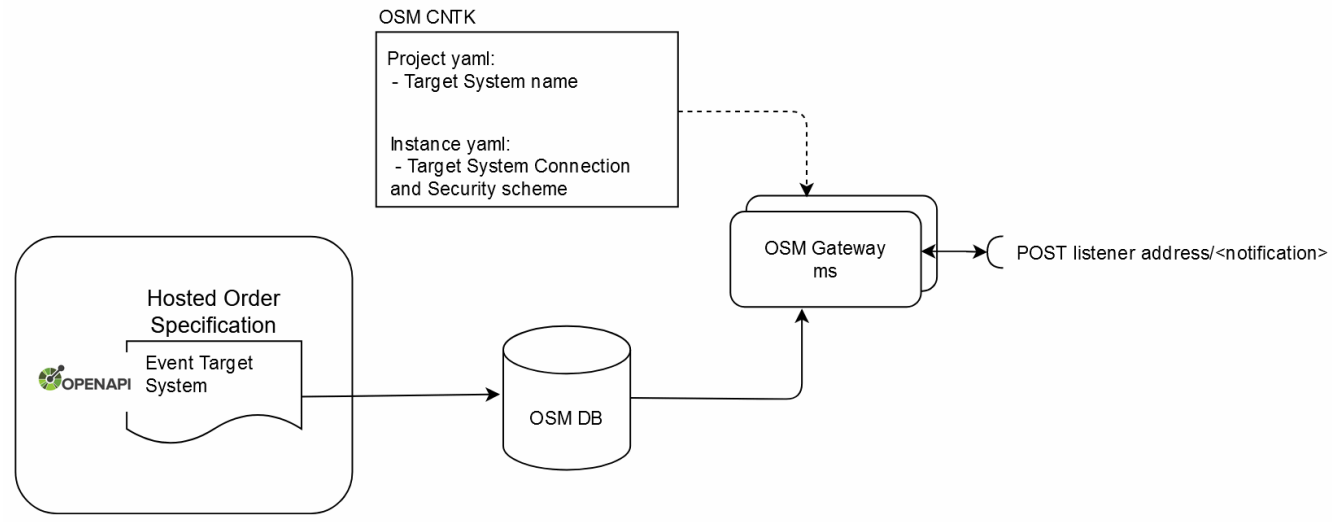
The Hosted Specification entity in Design Studio includes an "Event Target System" name which reflects a logical system name. All events coming from orders linked to a given Hosted Order Specification (including specification version) are sent to this consumer. Because OSM Gateway emits notifications to a single system only, if the solution design requires events to be sent to multiple consumers or to be filtered in user-defined ways, this is the responsibility of the single consumer defined in the Event Target System.

Deployment Configuration

The cloud native toolkit looks in the specification files for a mapping between the logical target system name (project specification) as well as the URL to the consumer and any authentication configuration (instance specification). At runtime, OSM Gateway resolves the logical target system name for a given Hosted Order Specification by reading deployment artifacts created by the toolkit during instance creation.

The following diagram illustrates the configuration of event target system.

Figure 8-9 Event Target System



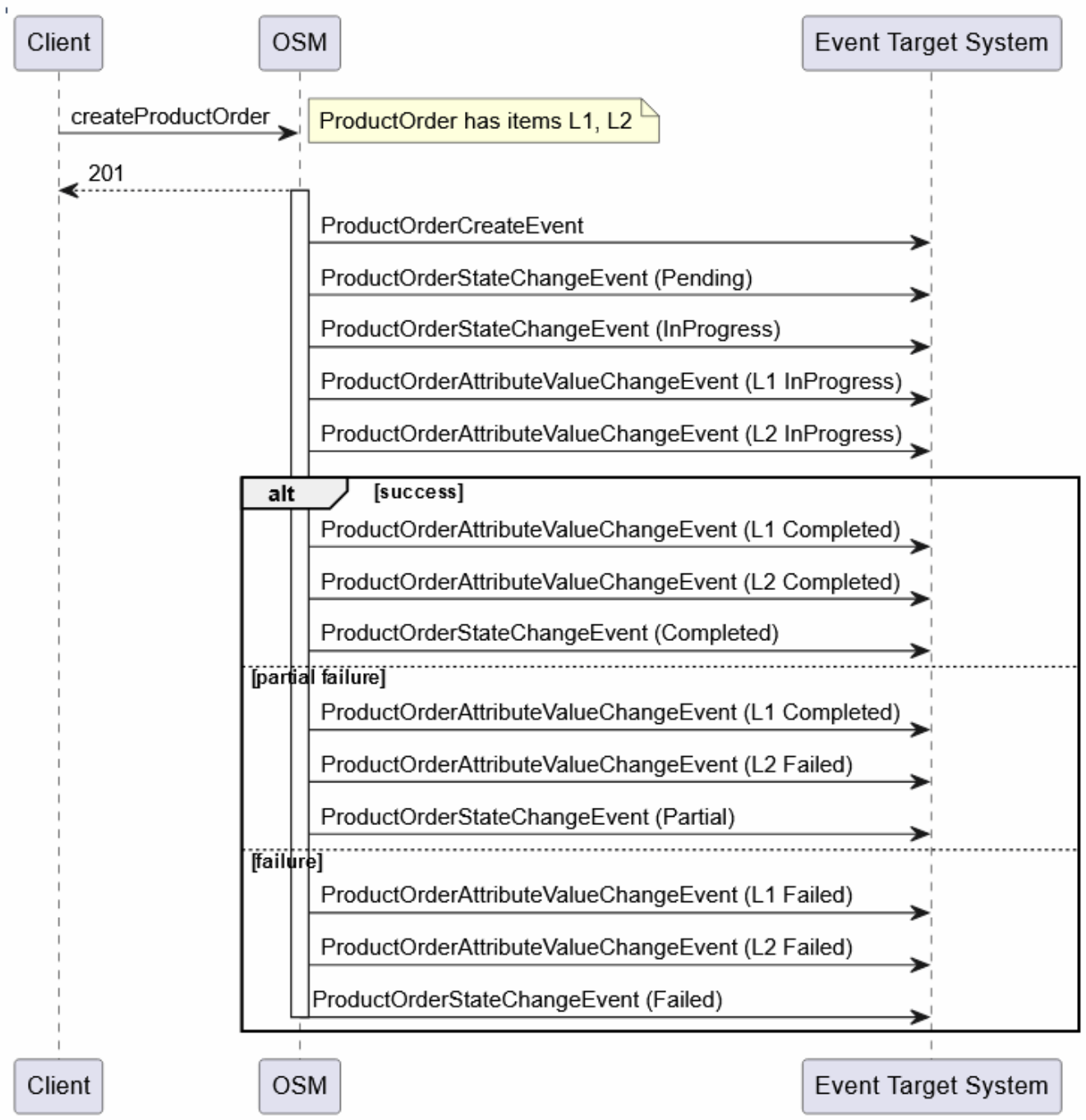
Order Processing Sequence Diagrams

This section provides order processing sequence diagrams for the following flows of a TMF order:

- Basic order flow
- Suspend and Resume
- Cancel order
- Revise order

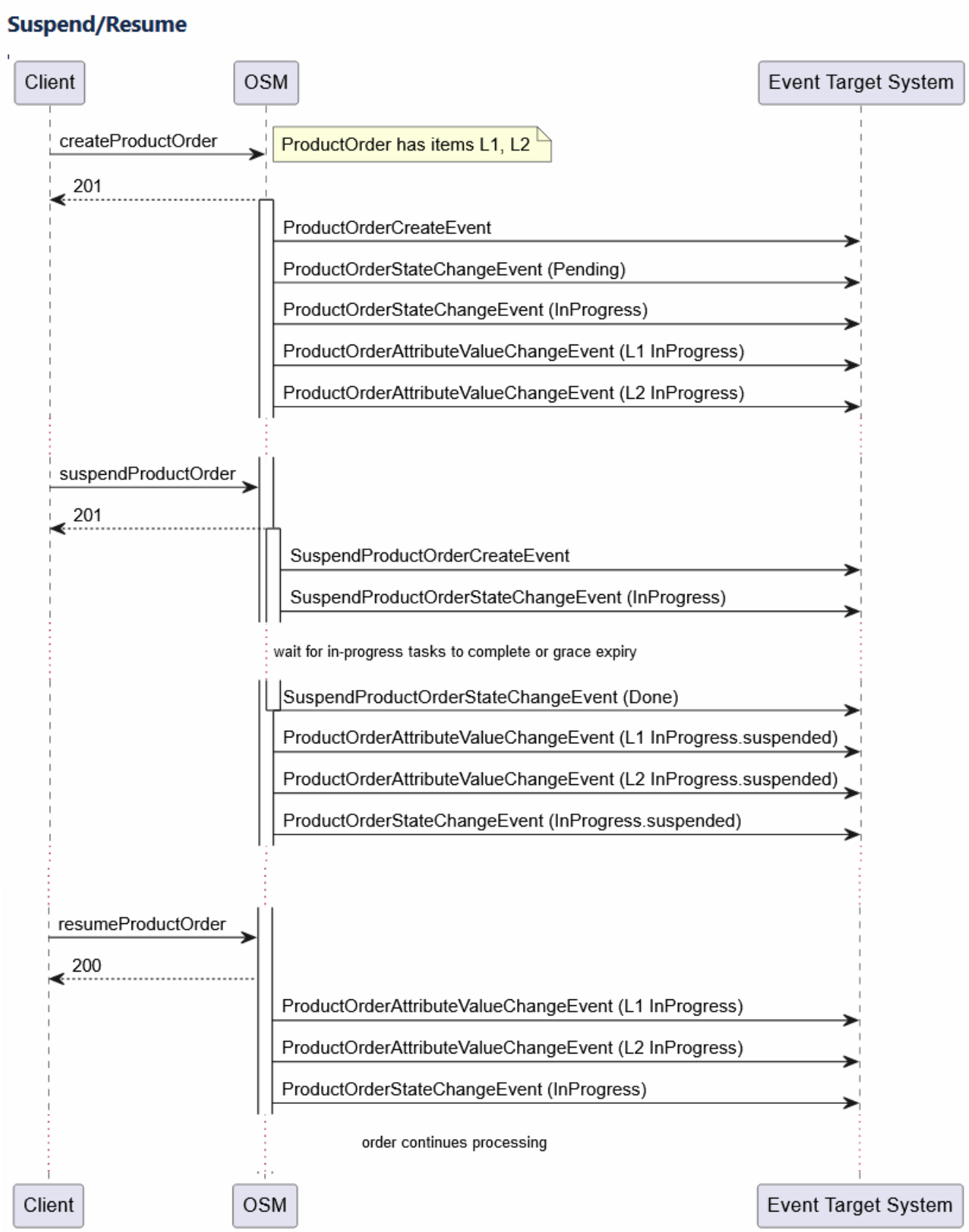
The following diagram illustrates the Basic Order flow of a TMF order.

Figure 8-10 Basic Order Flow



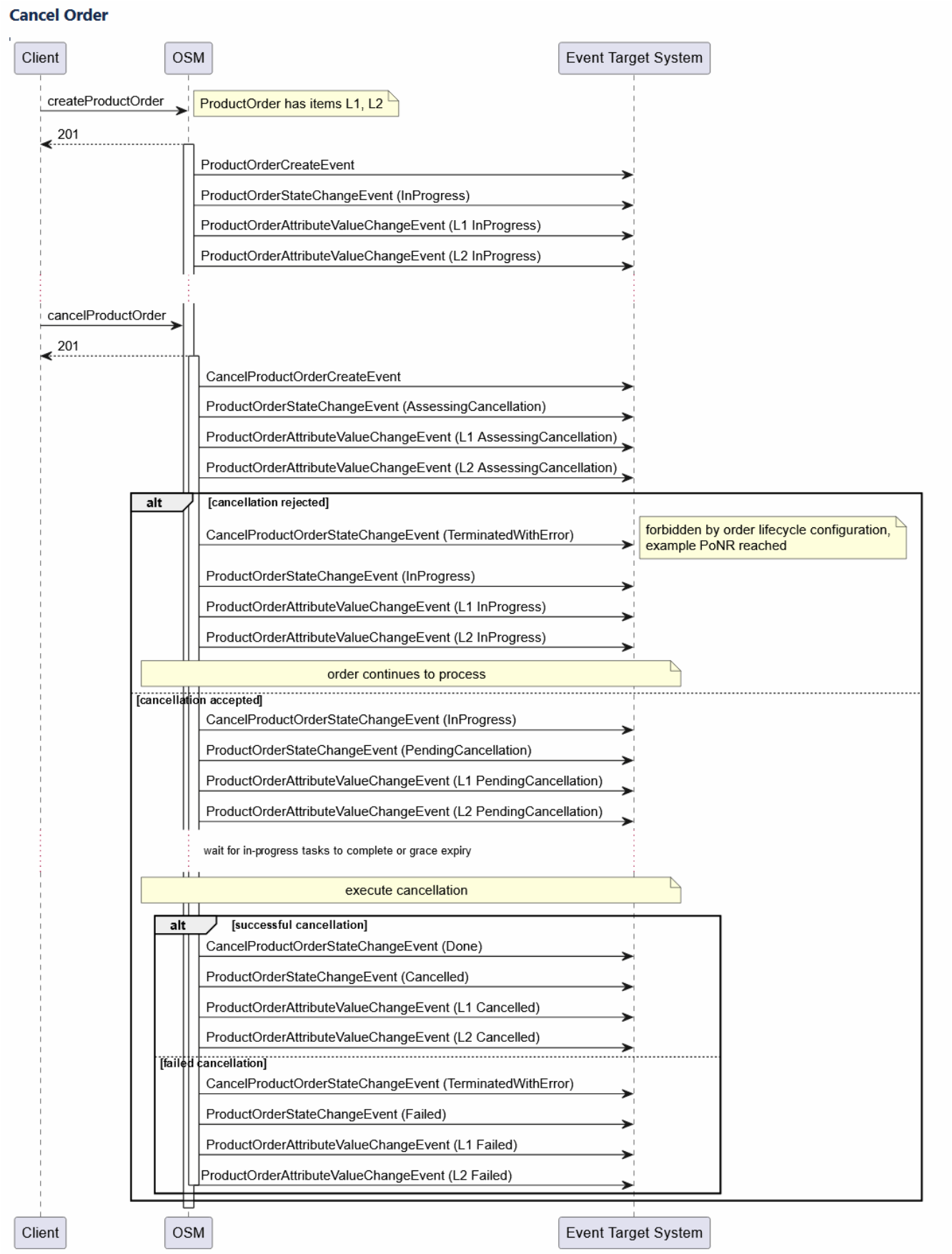
The following diagram illustrates the Suspend and Resume Order flow of a TMF order.

Figure 8-11 Suspend and Resume Order Flow



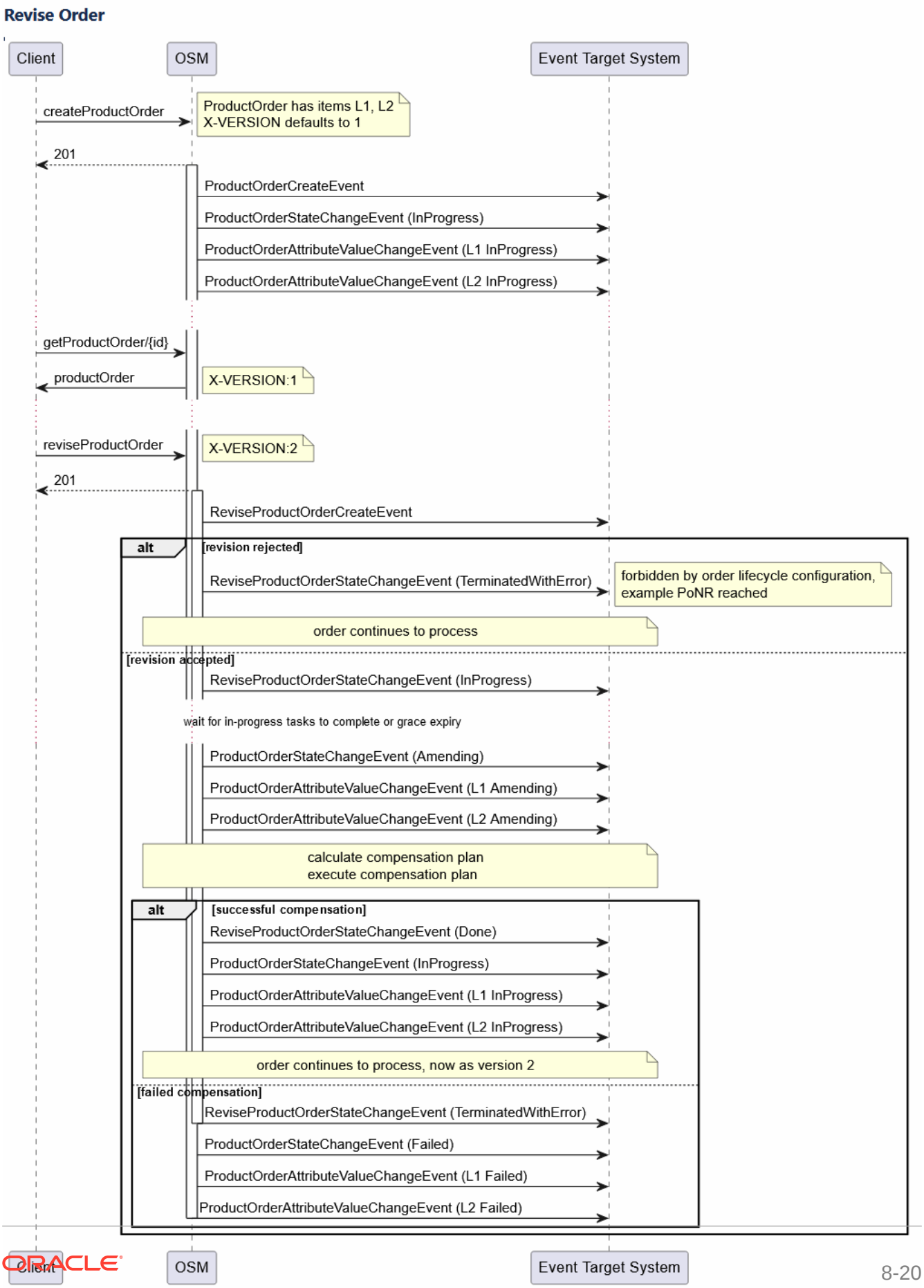
The following diagram illustrates the Cancel Order flow of a TMF order.

Figure 8-12 Cancel Order Flow



The following diagram illustrates the Revise Order flow of a TMF order.

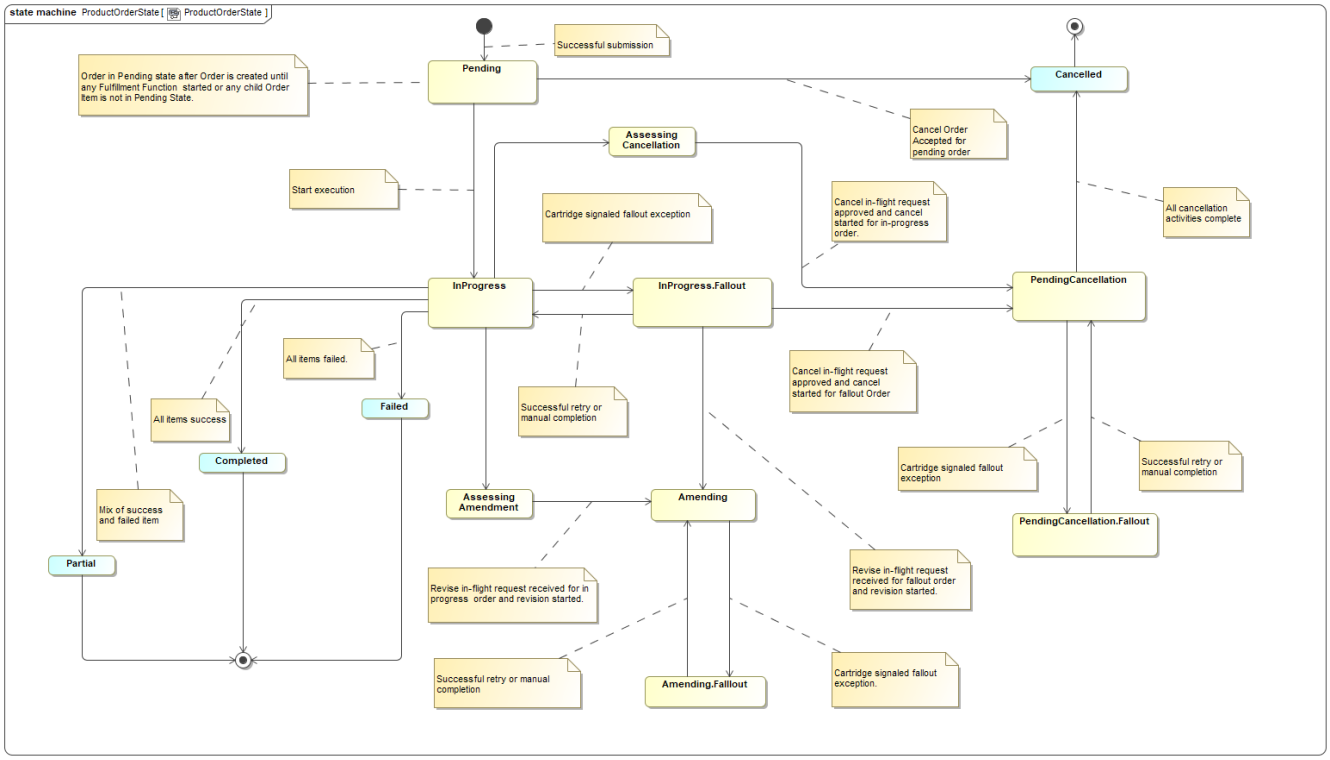
Figure 8-13 Revise Order Flow



TMF Product Order State Diagram

This section provides an illustration of the TMF Product Order State.

Figure 8-14 TMF Product Order State Diagram



9

About Runtime Order Management

This chapter provides an overview of the Oracle Communications Order and Service Management (OSM) runtime tasks. Before reading this chapter, read "Order and Service Management Overview ."

For information about using the OSM web clients, see *OSM Order Management Web Client User's Guide* and *OSM Task Web Client User's Guide*.

About Managing Orders

Order management personnel use the Task web client and the Order Management web client to manage orders. Order management tasks include:

- **Completing manual tasks.** Most of the tasks required to complete an order are automated. Manual tasks are completed in the Task web client. An order manager can add comments to the order, attach documents, display the history of the order, and receive notifications about at-risk orders or tasks.
- **Managing fallout**. You can identify errors that occur during order fulfillment, notify the appropriate individuals or systems, and take corrective measures.
- **Managing the order life cycle.** Most changes to the order life cycle occur automatically. However, an order manager can manually suspend and resume orders, cancel orders, update orders, and create orders.
- **Running reports.** You can run reports to get information about the overall order processing load. You can run the following summary reports:
  - Pending Orders
  - Order Volume
  - Completed Order Statistics
  - Completed Tasks Statistics

Assigning Tasks to OSM Users

There are two approaches to assigning tasks to users in OSM:

- A work offer approach, which is by role, where tasks are associated to a role and users performing that role may select tasks from their worklist to work on them. This is the standard task assignment approach in OSM.
- A work assign approach, where a task assignment algorithm is used to specifically assign each task to a user performing the role. This approach requires additional OSM modeling.

For each manual task, you can specify how it is assigned to an OSM user for completion. You can use the following methods:

- **Round robin** assignment automatically assigns tasks to users in a workgroup alphabetically by user name.

- **Load balancing** assignment automatically assigns users in a workgroup to balance the workload across users, based on the number of tasks assigned to each user. The user with the least number of tasks is assigned the task.

You can also create custom automatic assignment methods. For example, you might specify that the first task received is the first one assigned or that the last task received is the first one assigned.

About Workflow and Workstream Processes

When you create processes in Design Studio, you specify if a process is a workflow process or a workstream process.

With a **workflow process**, the Task web client displays the worklist after it completes each task. This is because a workflow process is intended to distribute work among different users in different workgroups. The next task in the process might be handled by a different user.

With a **workstream process**, OSM displays the order editor page for the next task automatically without first returning you to the worklist.

A workstream process can include manual and automated tasks. If an automated task occurs, OSM processes it and displays a message indicating that processing is taking place. While automated processing is occurring in the workstream, you can return to the worklist to work on other tasks. The automated task in the workstream will continue to progress to completion. This type of process is useful when the automated task in the workstream takes some time.

After the automated task finishes, and the next task becomes available, any user in the workgroup can pick up the workstream from that point. When the final task in a workstream completes, OSM returns the user to the worklist. OSM automatically displays the order editor page for the next manual task in the workstream to the user.

About the Order Lifecycle Management UI

Using the Order Lifecycle Management UI, you can view information about an order, such as fulfillment dates, the order's progress on a timeline, and order processing errors. The Order Lifecycle Management UI is available on the **Timeline** tab of the Order Management web client. If your system uses the **Oracle Configure, Price, and Quote Cloud (Oracle CPQ Cloud)** application as the order capture software, you can view the Order Lifecycle Management UI within Oracle CPQ Cloud.

The Order Lifecycle Management UI provides online Help. Click the Help icon in the UI for more detailed information.

About Managing OSM Users

Order managers use the Task web client to manage an order in runtime. To control which users can process specific types of tasks or perform specific actions in OSM, you define users in Oracle WebLogic Server. You can assign users to two types of groups:

- Use WebLogic Server groups to define who can access system administration tasks, such as who can use client tools, read log files, and start and stop OSM server components.
- Use OSM workgroups to define how users can perform order management tasks, for example:
  - You can assign permissions to carry out tasks such as assigning tasks to other users, running reports, changing order priority, and so on.

  - You can specify the data that a user can see in the Task web client.
  - You can specify the orders that OSM users can manage, based on data in the order. For example, you specify that a user can see only orders from a region or for a specific type of service.

To create OSM workgroups, you do the following:

1. Create **roles** in Oracle Communications Service Catalog and Design - Design Studio. The term *role* in Design Studio means the same as the term *workgroup* in the Task web client and the Order Management web client.

   When you create roles, you define permissions; for example, you can specify who can display reports in the Task web client or who can change the priority of a task during order processing. You can assign roles when you model OSM entities. For example, you can specify the roles that can manage an order, or process a task.

2. Assign users to workgroups by using the Order Management web client. This is typically performed as a one-time configuration task.

For example, you might create roles based on what a user can do in the Task web client. You can assign the same tasks to multiple workgroups, but users in each workgroup can work with it differently; for example, you might have a workgroup specifically for fallout management.

Roles are also used by automated tasks. For example, automated tasks use OSM roles to restrict who can receive a notification.

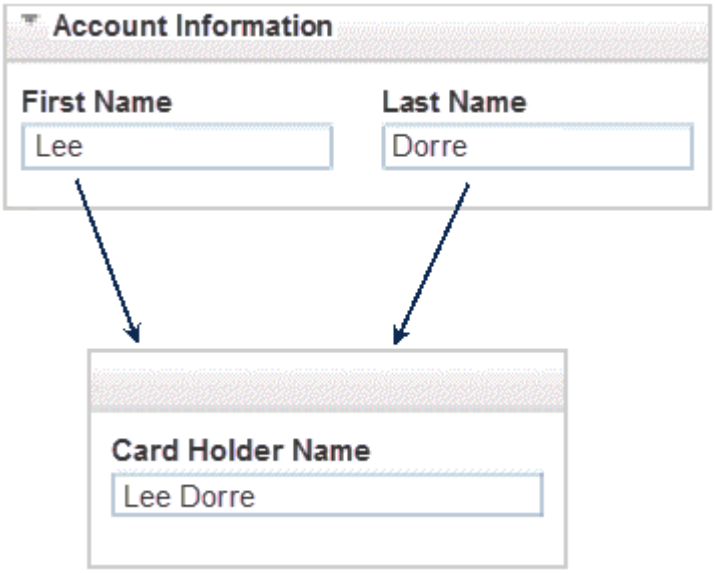
About Using Behaviors to Customize the Task Web Client

You can use **behaviors** to specify how OSM manages data. For example:

- You can specify the maximum allowed number of characters for text string data.
- You can add the values of multiple fields and display the sum in another field.
- You can specify the minimum and maximum times that a data element can be used in an order. For example, an order might require that exactly two IP addresses are added.

Figure 9-1 shows an example of changing the user interface by using behaviors. In this example, data from two fields is combined into one field.

Figure 9-1 Combining Two Fields



For more information, see the discussion about modeling behaviors in *OSM Modeling Guide*.

10

OSM Functional Overview

This chapter provides a diagram of Oracle Communication Order and Service Management (OSM) functionality.
Before reading this chapter, read the previous chapters, starting with "Order and Service Management Overview ."

OSM Functional Diagram

Figure 10-1 shows a functional diagram of OSM.

Figure 10-1 OSM Functional Diagram

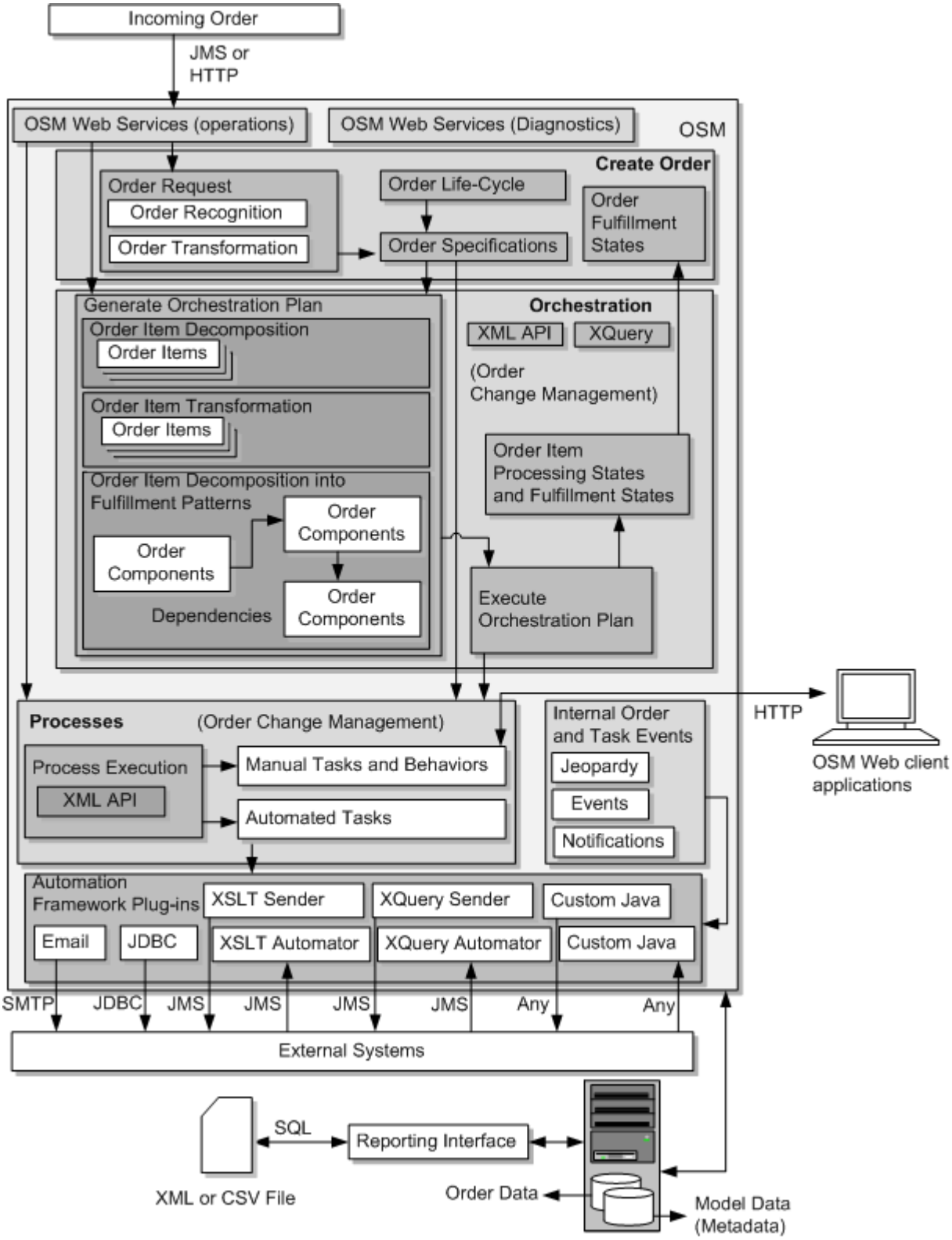


Figure 10-1 shows the three primary stages in order management: create the order, generate and run an orchestration plan, and run processes and tasks.

- To create an order, OSM receives an order as a web service operation from an order source system. When an order request is received, OSM performs order recognition to determine which type of order to create, based on the order specification. Transformation rules transform the order data into a format that can be used by OSM. See "How OSM Receives and Creates Orders."

  The order specification also defines the order lifecycle policy, which governs the states that the order can transition through. For more information, see "About the Order Lifecycle Policy."

- To generate and run an orchestration plan, OSM derives order items from the order data, and decomposes the order items into order components, based on fulfillment patterns. See "How OSM Generates and Runs an Orchestration Plan" and "About Decomposition."

  The orchestration plan includes dependencies that determine the order in which processes can run. See "About Dependencies."

  When processing a customer order, OSM can use order item transformation to transform order items for products, bundles, and offers into order items for customer-facing services. See "About Order Item Transformation" and "About COM, SOM, and TOM."

- When running processes, OSM runs manual tasks and automated tasks. Manual tasks are managed by the Task web client. Automated tasks are run as plug-ins by the automation framework. Plug-ins use EXtensible Stylesheet Language Transformations (XSLT), XQuery scripts, and Java to communicate with external systems. See "About Tasks and Processes."

Figure 10-1 shows the areas in OSM that carry out functionality related to business processes such as changes to in-flight orders:

- Fulfillment states can manage states in orders and in order items.
- Predefined processing states can manage order item states. (Note that order item processing states do not go to the order level as fulfillment states do.)
- Order change management, such as using revision orders and compensation, is performed when processing tasks. Revising an order can also require a new orchestration plan.
- Internal order and task events throughout the order management process can create events that trigger notifications to order management personnel.

Figure 10-1 shows the interfaces used by the OSM functionality:

- CRM systems use JMS queues to send an order to OSM. HTML can be used in test and demonstration systems.
- You configure orchestration by using XQuery scripts to find, get, and transform data. You can also use XQuery scripts in processes.
- The Task web client and the Order Management web client communicate with the OSM server by using HTTP.
- The reporting interface uses SQL to retrieve data from the OSM database.
- OSM uses a variety of interfaces to connect to external systems, such as billing or shipping systems.

Glossary

activation

The enabling, disabling or changing of a network resource; for example, creating a local loop for a DSL service. Activation is preceded by provisioning . Activation is typically initiated in a technical order run by OSM in the technical order management (TOM) role.

activation task

A type of automated task designed specifically to interact with the Oracle Communications ASAP product or the Oracle Communications IP Service Activator product.

amendment processing

A generic term that refers to making changes to in-flight orders. Amendments are typically made when processing a revision order, or managing an order cancellation or order failure. The amendment usually performs compensation; such as redoing or undoing tasks.

ASAP

An Oracle product used by communication service providers to activate operational support systems equipment across multiple technology domains. ASAP supports many hardware vendor's network systems, and is integrated with OSM using activation tasks.

automated task

A task that does not require manual activity. Automated tasks handle interactions with external systems such as billing systems, shipping systems, and activation systems. They can also perform custom calculations and other operations. See manual task.

automation framework

An interface that enables the integration of OSM with external applications. It is used to automate tasks and notifications to other systems. It can also be used to perform business logic (such as performing complex calculations) without interacting with an external system. The automation framework is an OSM server component that performs the work required by automation plug-ins.

automation plug-in

An OSM component that performs the operation specified by an automated task. For example, you can create automation plug-ins to update order data, complete order tasks with

appropriate statuses, set process exceptions, react to system notifications and events, send requests to external systems, and process responses from external systems. OSM provides several predefined plug-ins. You can also develop your own plug-ins.

behaviors

OSM behaviors allow you to control the validation and presentation of data elements in the OSM Task web client. For example, you can use the **Calculate** behavior to derive the value of the data in a field by adding the values in two other fields. You could use the **Information** behavior to present a tool tip for a field in the Task web client.

In addition to customizing the Task web client, you can also use the **Data Instance** behavior to retrieve data by using data providers. See data provider.

cartridge

A collection of entities and data defined in Design Studio and packaged in an archive file for deployment to a runtime server. In Design Studio, you build cartridges in cartridge projects. You can create your own custom cartridges to extend Oracle Communications applications. Additionally, you can obtain from Oracle customized cartridges that support integration with other common applications, and cartridge packs that bundle cartridges containing metadata for particular technology domains; for example, order specifications, recognition rules, processes, behaviors, and so on.

central order management (COM)

The OSM role that receives orders from one or more order-source systems such as Siebel CRM, creates an OSM customer order, and manages the fulfillment of the order across other enterprise systems including billing and OSM in the service order management (SOM) role. OSM operating in the COM role also receives status information from these systems and provides visibility of an order's status back to the order-source system.

CFS

See customer-facing service (CFS).

COM

See central order management (COM).

common fulfillment state

A fulfillment state that can be assigned to an order item, order component, or order. Common fulfillment states are user-defined values that provide consistent state definitions for various entities. Common fulfillment states are available for the entire workspace. To define a common fulfillment state, you define its name, and then apply either a mapped fulfillment state or a composite fulfillment state to it. For example, a common fulfillment state named **Failed** could be used for a fulfillment state mapped from **Error**, or a composite state of **Error + Unsuccessful**.

See mapped fulfillment state and composite fulfillment state.

compensation

The process of comparing the requirements in a revision order to the requirements in the base in-flight order, and making the changes. For example, if a customer initially orders Bronze-level DSL service but upgrades to Gold-level service while the original order is in place, tasks may need to be done, redone, or undone. OSM automatically calculates the compensation required to accommodate the changes to orders.

compensation plan

An internal process that OSM defines and runs to process a revision order. The compensation plan defines which tasks need to be done, redone, or undone.

composite fulfillment state

The fulfillment state that results from combining the fulfillment states from order items into a single fulfillment state.

control data

Metadata in an orchestration order that is used to manage how the orchestration plan runs. OSM extracts control data from an order. Control data provides information about order items, order components, and dependencies required to generate the orchestration plan. This includes status and timestamps for its order items and components. When an orchestration plan is run, the order data, including control data, can be updated as transactions are completed.

Design Studio automatically generates control data for order components. You manually model control data for order items.

creation task

The task that contains data required for the order. The creation task specifies what information must be provided to the order before it can start processing. It is used by OSM in the order process; for example, when an order is canceled, the order is returned to the creation task. A creation task is defined in the order specification.

CRM

Customer relationship management. A system for managing a company's interactions with customers, clients and sales prospects; for example, Oracle's Siebel CRM.

customer-facing service (CFS)

A service that implements what the customer ordered, independent from how the order is implemented on the network. A CFS is typically the service as the customer would recognize it;

for example, Internet service. By contrast, a resource-facing service (RFS) is the service as implemented on the network; for example ADSL or VDSL.

customer order

An order processed by OSM in the central order management (COM) role. OSM creates a customer order from the order received from the CRM system. The customer order typically fulfills billing functions and calculates the requirements for a service order that is sent to OSM in the service order management (SOM).

Data Dictionary

A logical collection of data elements and data types in a workspace, enabling you to leverage common definitions across an entire Oracle Communications solution. For example, the Data Dictionary enables you to create order templates, atomic actions, and service specifications, and share the data defined for those entities across your OSM, ASAP, and Inventory applications. Entities in a workspace contribute data types to the Data Dictionary, and data schemas in a workspace (which are accessible across all projects) contribute data elements to the Data Dictionary.

The Data Dictionary enables you to integrate and correlate data models for multiple applications, reduce the size and complexity of a solution model, simplify the application integration by eliminating data translation among applications, and validate data model integrity. See also data schema.

data element

A structured or simple type data definition. When modeling data for a project, you create data elements that you can reuse throughout your model. There are two types of data elements: simple data elements and structured data elements.

See simple data element, structured data element.

data provider

An adapter that can retrieve order data from external systems in an XML format. Design Studio provides several built-in data providers to retrieve external XML instances from specific sources such as a JDBC database or a SOAP web service. Additionally, you can create your own custom data provider. Data providers are used when defining **Data Instance** behaviors.

data schema

An XML schema that provides a formal description of a data model, expressed in terms of constraints and data types governing the content of elements and attributes. All data elements are created and saved in data schemas, which are accessible across all projects in a workspace. Design Studio automatically creates a project-specific data schema when you create a new project. You can use this default schema to contain the data you require to model the project, you can create multiple schemas in the same project, or you can create schemas

in common projects. You can model your projects using data from any combination of these data schemas.

decomposition

The process by which order line items in a sales order are transformed into order items, which are then organized into order components. For example, OSM can use the following algorithm to achieve decomposition:

- Map order items to functions (fulfillment function order components)
- Map function order items to fulfillment systems (fulfillment system order components)
- Map fulfillment system order items to processing granularity (granular order components)

See also order component, executable order component, orchestration stage.

decomposition rule

Rules that determine the order items in each order component during decomposition. OSM evaluates every order item in the source order component against the conditions that you define for the decomposition rule.

Unlike many other OSM modeling entities, decomposition rules are not directly referenced by other parts of the model. OSM selects decomposition rules by matching the source and target order components of the decomposition rule to the order components in the orchestration stages in the orchestration sequence.

See orchestration, order item, order component.

default process

The first process that runs after an order is created. A default process can either be an orchestration process (which will be backed by a dynamically generated orchestration plan) or a workflow or workstream process.

delay

A process activity that specifies that a process stops until a condition evaluates as true. In OSM there are two types of delays, timer delays and event delays. A timer delay retries the evaluation of the rule at a fixed time interval. An event delay retries the evaluation of the rule only when order data changes.

dependency

A relationship in which a condition related to one order component or order item must be satisfied before another order item can be processed. For example, it may be necessary to perform provisioning before billing can occur for the same order item. Dependencies can have the following relationships:

- Between different order components for the same order item.

- Between different order components for different order items.
- Between order items across orders.
- Time-based dependencies.

See also inter-order dependency, intra-order dependency.

design and assign

The process of determining the resources required to implement a service, and assigning the resources based on the available inventory. For example, the design for a DSL service might specify a port and a local loop, which would be assigned based on the inventory. Design and assign is typically initiated in a service order run by OSM in the service order management (SOM) role.

Design Studio

An integrated design environment for the development of solutions based on the Oracle Communications OSS Applications. Design Studio enables solution designers to configure application-specific and multi-application solutions by leveraging application-specific concepts. Design Studio is built on an open architecture based on the Eclipse framework, and it uses a wide variety of innovative technologies.

entity

A functional unit created and edited in Design Studio; for example, tasks, processes, physical and logical resources, and order specifications. Entities are collected in projects and deployed to runtime environments to support your business processes.

event delay

See delay.

executable order component

An order component with an associated process. Typically this is a component decomposed to its final level of granularity. Executable order components are generated during the last orchestration stage in an order.

See also decomposition.

expected duration

The amount of time an order, or some part of the order, (for example an order component, fulfillment pattern or task), is expected to take to complete processing. OSM uses the expected duration to calculate the expected start date and the expected completion date.

expected start date

The date on which an order is expected to start being processed. Expected start date is determined by calculating the expected order duration and factoring this in with the requested delivery date for order items on the order.

external fulfillment state

The status returned from a fulfillment system to an order component. This may be the exact status returned by the system, or automation may be used to translate the status before it is put on the order. It is a key input into a fulfillment state mapping.

See fulfillment state.

fallout

The failure of an order or task during processing. Fallout occurs whenever an order or task encounters a situation that prevents it from processing normally. Causes for fallout include missing data or the inability to access a fulfillment system. Fallout management includes detecting, investigating, resolving, or escalating failed orders and tasks.

fallout exception

A mechanism initiated from the OSM Task Client or from an automated task automation plug-in to interrupt or stop an order or task, to redirect it to any task in the same process or any other process, or to transition a task into a fallout execution mode.

fallout management

A process that includes detecting, investigating, resolving, and escalating failed orders and tasks. Administrators perform fallout management with the OSM web clients. The clients allow them to search for failed orders and tasks, identify the reason for the failure by viewing order and task details, and resolve errors and dependencies to allow order processing to proceed. In cases where the fallout scenarios cannot be resolved, you can cancel or terminate the order.

follow-on order

An order that is submitted to modify a completed order. Follow-on orders are not processed until their order-item dependencies on the in-flight orders allow them to proceed.

fulfillment

Operations that fulfill a customer's order, such as providing, modifying, resuming, or canceling a customer's requested products and services.

fulfillment function

An operation that must be performed to process an order; for example, initiating billing, shipping a modem, or activating a service.

fulfillment mode

An entity that represents the intent of an order. For example, the fulfillment mode could indicate whether the order is intended for qualification, delivery to fulfillment systems, testing and so on. Every customer order can specify a fulfillment mode.

fulfillment pattern

An entity that includes the fulfillment function order components and dependencies required to fulfill a product order. Each order item in an order is mapped to a fulfillment pattern. OSM uses the fulfillment pattern to determine the necessary fulfillment functions, order components, and dependencies to generate an orchestration plan.

fulfillment state

The state of an order or order item aggregated and translated from status values returned by external systems. This state can be used to provide status visibility to upstream systems and to users by using the Order Management web client.See also common fulfillment state, composite fulfillment state, external fulfillment state, mapped fulfillment state.

fulfillment state map

The Design Studio entity that contains both the definition of common fulfillment states and fulfillment state mappings. A common fulfillment state defined on one fulfillment state map is available to all fulfillment state mappings in the workspace.

See common fulfillment state and fulfillment state mapping.

fulfillment state mapping

The Design Studio entity that maps external fulfillment states to values from the common fulfillment state list. The resulting fulfillment state is referred to as a mapped fulfillment state.

fulfillment system

A system that carries out the actions necessary to complete the order; for example, activate services on the network, ship equipment, or run billing. To process an order, OSM sends commands to fulfillment systems to carry out their functions and return the status of the fulfillment action.

fulfillment topology

The fulfillment systems required to fulfill orders. For example, a fulfillment topology might include a CRM system, OSM running in the COM role, multiple instances of OSM running in the SOM and TOM roles, a billing system, a shipping system, and so on. The fulfillment

topology also defines the relationships between the fulfillment systems; for example, the activation systems and shipping systems that OSM in the TOM role interacts with.

future-dated order

An order that has a requested delivery date that is later than the current date and time. For example, a customer order to have a new VoIP service added at the beginning of the next month is a future-dated order. OSM uses the order orchestration plan to calculate the order start date of future dated orders so the order can be completed by the time the customer wants it.

See also expected duration and expected start date.

inbound order

See customer order.

in-flight changes

Changes that are made to an order that is being processed.

in-flight order

Any order that is not in a closed state (**Closed** or **Aborted**). An in-flight order still has the potential for further work to be performed on it.

inter-order dependency

A dependency between order items in different orders.

intra-order dependency

A dependency between order items in the same order. An intra-order dependency can refer to external information, but not to data in other orders.

IP Service Activator

Internet Provider Service Activator. An Oracle product used by communication service providers to define and fully automate the activation of services on large-scale multi-vendor IP networks. IP Service Activator delivers end-to-end network control and enables real time reaction to new service and customer demands.

line item

See order line item.

lifecycle policy

An order attribute that defines the states that an order can have; for example, In Progress and Suspended, and the rules governing transitions between states.

JMS queue

A method used by systems to communicate with each other. To communicate with external systems, the OSM server uses mostly Java Message Service (JMS) queues. JMS is part of the standard Java platform. A JMS queue is a staging area that you configure when you install OSM. Systems communicate via JMS queues by publishing messages to them and receiving messages from them.

manual task

Tasks performed manually by OSM operations personnel using the Task web client. See also automated task.

mapped fulfillment state

The fulfillment state that results from a fulfillment state mapping.

metadata

The data that you create when modeling OSM in Design Studio. Metadata is *data about data*, that is, metadata defines how data is created and used in the OSM system. OSM uses metadata to determine how to process order data. For example, OSM uses metadata from the order specification as a template to create orders at runtime.

mnemonic

A synonym for an entity name. Mnemonic is a legacy term for OSM. The proper name is entity name.

multi-instance data element

A data element that is permitted to have more than one instance. For example, you configure the **ControlData/OrderItem** structure as a multi-instance data element so that OSM can create an instance of the structure for every order line item extracted off the inbound order.

namespace

A method for uniquely naming elements and attributes in an XML document. Design Studio supports entity and cartridge namespaces. You pair the entity or cartridge name with a namespace name to create a fully qualified namespace. For example, you can pair entity names with a namespace name to enable different groups of Design Studio users to create different entities without concern for naming conflicts. Services can be implemented independently by different teams and then deployed into a single runtime environment.

Not every OSM entity has a separate namespace (example: tasks and processes). For these types of entities, a unique name is created by attaching the cartridge name and version number to the entity name.

notification

Messages sent by OSM to alert users of order problems (jeopardy notifications) or changes to an order's state, status or data (event notifications). By default, OSM sends most types of notifications to the Task web client Notifications page. You can also specify that notifications be sent by e-mail.

Oracle Application Integration Architecture (Oracle AIA)

Integrates business processes across multiple applications. OSM integrates with Oracle AIA for Communications. Oracle AIA runs on top of Oracle Fusion Middleware.

Oracle Application Integration Architecture (Oracle AIA) Order-to-Activate Cartridges

A set of OSM cartridges that integrate with the Oracle Communications Order to Cash Integration Pack for Oracle Communications Order and Service Management (Order to Cash Integration Pack for OSM). The Order-to-Activate cartridges and the integration pack enable OSM to be part of an order fulfillment solution that covers the entire order-to-activate process from order creation to service activation.

Oracle Configure, Price, and Quote Cloud (Oracle CPQ Cloud)

Oracle's order capture software that enables companies to select products, assign pricing, create quotes, and submit orders.

Oracle WebLogic Server

Oracle's application server for building and deploying enterprise Java EE applications. The Oracle WebLogic Server hosts the OSM server, OSM integration, and related interfaces.

orchestration

The process OSM uses to manage the fulfillment of an order across many fulfillment systems. Dependencies may require that these interactions be run in a specific order to ensure that order items are sent to the proper systems, and that the required steps, in the proper sequence are run.

orchestration order

An order that requires an orchestration plan for fulfillment. Orchestration orders contain control data for an orchestration plan. The default process for an orchestration order is an orchestration process. Most orders are orchestration orders. See process-based order.

orchestration plan

A dynamically generated plan that is used to manage the fulfillment of an order. Order fulfillment often requires interaction with many fulfillment systems, and various dependencies may require that these interactions be run in a specific order. The orchestration plan includes the order, order component, and the type and the sequence of order component execution. An orchestration plan is generated for each order based on the metadata defined for the type of order being processed.

For example, an order is captured by Siebel CRM and is sent to OSM for processing. Using the recognition rules and other entities provided by the OSM cartridges in the Order to Cash Integration Pack for OSM, OSM decomposes the order and dynamically generates an orchestration plan that is used to manage the fulfillment of the customer's order across other enterprise systems.

orchestration sequence

A set of orchestration stages for an order. Orchestration sequences specify the set of orchestration stages for an order. Orchestration stages and sequences together define how an order is decomposed.

See also decomposition, orchestration stage.

orchestration stage

A step in an orchestration sequence used to decompose an order and generate an orchestration plan.

See also decomposition, order.

order

An order in the OSM format. You model orders by creating order specifications in Design Studio.

There are many order variants including:

- customer order
- follow-on order
- future-dated order
- inbound order
- in-flight order
- orchestration order
- process-based order
- revision order
- service order

- technical order

order component

A collection of order items that can be processed together because they meet some common criteria as determined by an orchestration stage. Order components are modeled in Design Studio, based on factors such as a function that needs to be performed, the systems that need to perform that function, and what other items can be processed in the same group.

See also decomposition, executable order component.

order component ID

An ID associated with an order component that can be used in decomposition. When implementing fulfillment systems, for example, you can configure OSM to decompose an order by using decomposition rules or by using the component IDs.

order data

The data that is used for fulfilling an order; for example, the customer name and address, required bandwidth, and so on.

order data key

Uniquely identifies a data element or structure in an order by differentiating the data element or structure based on a data element value. Order data keys are important when identifying order data changes during compensation and for multi-instance data elements.

order definition

See order specification.

order duration

See expected duration.

order entity

See order specification.

order fallout

See fallout.

order fulfillment state composition rule set

The Design Studio entity used to aggregate and evaluate the fulfillment states of root-level order items and compose them into a single composite fulfillment state for the entire order.

See fulfillment state.

order header

The part of an incoming sales order that contains information applicable to the entire order such as the sales order number, the order action (Add, Cancel, Delete, and so forth), and the customer name and address. An order in the OSM format does not have a header; only orders received from external order source systems.

order item

An order line item transformed so that it can be processed in OSM. Order item properties include the action required to implement it, such as Add, Suspend, and Delete. Order items are decomposed into order components during orchestration.

See also order component.

order item fulfillment state composition rule set

The Design Studio entity used to aggregate and evaluate the fulfillment states of order component order items and child order items and compose them into a single composite fulfillment state for the entire order item.

See fulfillment state.

order item transformation

The process of using rules to derive one set of order items from another set of order items.

order key

A unique value that enables the system to match incoming revision orders to the corresponding OSM order. If the order key matches an order that is currently in progress, the order is considered to be a revision that amends a base order. For example, you can specify to use the customer reference ID as the order key. In that case, when OSM processes an order, it looks for previous orders that have the same customer reference ID, and amends it.

The order key can be any data or combination of data associated with the order. It is configured in Design Studio as an XPath expression to a data element that will uniquely match an amended order to its corresponding OSM order. For example, you might specify a customer reference ID as the following Xpath expression: `root/Cust_Ref_ID`

order life cycle

The sequential states through which an order passes and the transactions it undergoes from the time it is received in OSM until the time it is resolved. States include **Not Started**, **In Progress**, **Suspended**, and **Completed**. Each order state is associated with a set of transactions that can be performed while the order is in that particular state. Transactions

include **Update Order**, **Cancel Order**, **Complete Task**, and **Raise Exception**. The life cycle of an OSM order is governed by the order state model and **order life cycle**.

Order Lifecycle Management UI

A user interface that enables you to view the progress of an order that is being managed by an OSM system.

order lifecycle policy

A set of policies that controls the states in which an order can be, and the transactions allowed in those states. The order lifecycle policy also determines which roles can perform which transactions. For example, while an order is in the In Progress state, you might want your Customer Service role to be able to perform the Update Order, Cancel Order, and Suspend Order transactions, while your Fallout role is able to perform the Raise Exceptions transaction. Every order type you create must be associated with an order lifecycle policy.

order line item

Specific items such as individual products, services, and offers on an incoming order. OSM transforms order line items into order items.

Order Management web client

An OSM web application that displays an order's orchestration plan, including dependencies, order components, and order items. The Order Management web client is used by fallout administrators responsible for locating orders with errors, determining the causes of failures, and taking the necessary corrective actions; operations and management personnel who monitor the progress of orders; and orchestration plan designers who can use this application to test and validate orchestration-based orders during the modeling and implementation of OSM solutions. The Order Management web client also has an Administration area used to manage user workgroups, calendars and schedules, email notifications, and system errors.

order priority

A value that OSM uses to determine which orders should be given more processing resources. OSM uses order priority to determine the next thing to be done. Orders with higher priority will be processed before orders with lower priority. In situations where resources are constrained (for example, the system is using all available CPU, memory, or other resources to process orders), orders with higher priority will process faster than orders with lower priority.

order recognition

The process of determining the type of an incoming order so it can be mapped to an order specification in OSM. During order recognition, OSM steps through an ordered list of recognition rules to determine which rule applies to the customer order. Each rule is associated with an order specification.

See recognition rule.

order reference number

A value associated with an order specified in one of the OSM web clients or OSM APIs. OSM uses an order reference number as an identifier to external systems. Reference numbers can be used as keys to correlate orders between systems.

order specification

An order entity defined in Design Studio. The order specification is the central entity in OSM. It defines the basic information OSM requires for it to be able to process orders. It specifies such things as what data is allowed in an order, (defined in the order template), what are the range of order priorities, whether amendments are allowed and how they are processed, how to handle jeopardy, fallout, permissions and so on.

Other entities such as tasks, processes, and notifications require that you specify an order specification to which it relates. Order specifications can inherit from other order specifications, and multiple order specifications can be modeled and can exist simultaneously. Also known as an order definition and order entity.

order state

The condition of the order. For example:

1. An order is created in the **Not Started** state.
2. When processing begins on the order, the state changes to the **In Progress** state.
3. When the order is complete, it transitions to the **Completed** state.

See order life cycle.

order state transition

Changes from one order state to another order state as a result of a transaction. Each order state has a set of allowable transitions. For example, when an order is completed, it transitions from the **In Progress** state to the **Completed** state.

order template

A part of an order specification that defines what order data OSM will use to process and fulfill the order. For example, the order template defines the data required for order items as well as the data required in an order header. You create or modify order templates by adding data elements from one or more data dictionaries.

order transformation

The manipulation and enrichment of the structure and contents of an order through a set of rules. Transformation rules are applied to an incoming sales order in a recognition rule.

Three types of transformation rules are available: *order priority rules*, which define the priority of the order in relation to others; *order reference rules*, which define the order reference number; and *order data rules*, which add to or modify incoming customer order data.

order validation

A process that validates that an order is syntactically correct. When an inbound order is recognized, OSM validates it based on validation rules defined in the order recognition rule.

For example, a validation rule can determine that all mandatory fields are populated, that valid characters (numeric or alphanumeric) are used for fields, and that the order has a valid status code such as **Open**.

OSM order management web services API

The primary interface for external systems to OSM. The OSM order management web services provide for inbound order operations such as creating, managing, retrieving, updating, or canceling an order. Web services are web APIs that support interoperable machine-to-machine interaction over a network such as the Internet. Web services run on a remote system hosting the requested services such as OSM. Web service interfaces are described by the web service definition language (WSDL).

OSM security callback

A callback interface that allows you to generate an audit trail log of users before they gain access to order data that is considered sensitive. The security callback interface is designed to intercept order access from defined functions such as **GetOrder**, XML API **WorkList.Request**, and **Task web client Order Data History** page.

OSM server

The server that manages OSM runtime functionality, including inbound order operations and outbound communications with external systems. The OSM server is deployed on Oracle WebLogic Server.

OSM web clients

The two OSM GUI applications are called the Order Management web client and the Task web client. The Order Management web client displays an order's orchestration plan, including dependency, orchestration stages, order components, order items, and processes. The Task web client is used for monitoring and managing the tasks in an order.

point of no return

The point in the orchestration of an order item after which revisions can no longer be accepted.

process

A sequence of tasks and subprocesses that need to be carried out to fulfill all or part of an order. For example, an ADSL fulfillment process could include the following tasks that can take

place over a number of days: assign a port, ship modem to customer, activate DSLAM, send customer survey, and verify order. The process includes definitions of the relationships between tasks and the sequence in which they are run.

process-based order

An order that does not include an orchestration plan. See orchestration order .

processing granularity

Decomposition groups order items into optimally executable order components. For example, if monthly fees, VoIP adapter, and VoIP phone are all billed by the same billing system, they can be grouped into a single executable order component. This is called processing granularity. See decomposition.

processing state

Processing states are a predefined set of states that an order item can enter. OSM aggregates the values returned from external systems in each order component for the order item to determine the overall processing state of the affected order item.

product

A conceptual model entity that represents something that your business sells. Because Design Studio is primarily used for service fulfillment rather than sales, products are often identifiers associated with information from other systems.

product catalog

The complete collection of your products, offers, and bundles. A product catalog is typically stored as a data repository on the CRM or other order-source system; for example, the Siebel Sales Catalog or Oracle Product Information Management Data Hub.

product specification

Groups of related products that share common attributes. For example, suppose you sell products for three levels of DSL service. Though there are different values to differentiate the service levels, the products are structurally identical and provisioned by the same system; they are variations of the basic DSL service. As a result, a single product-specification to fulfillment-pattern mapping can be used for all of them.

The use of OSM product specifications has been deprecated, and new OSM product specifications cannot be created in Design Studio 7.2.4 and later. Conceptual model products, customer-facing service (CFS), resource-facing service (RFS), resources, and actions should be used instead.

project

An entity that contains artifacts (entities, data, rules, code, and so forth) that you use to model and deploy Design Studio cartridges. Your solution uses various types of projects. For

example, you use projects for version management, for single sourcing data, for resource organization, and to build cartridges that can be deployed to a server. You can create various types of projects and you can extend cartridges that you purchase with your own projects. Oracle Communications supports a library of extensible cartridges that are fully compatible with Design Studio and provide a basis from which to assemble solutions.

provisioning

Identifying the network resources required to enable a service. For example, to provision a phone service, the service and resource management (SRM) system finds a phone number. The provisioning data is used during activation to enable the service on the network.

recognition rule

Rules that enable OSM to validate an incoming order and transform it into an OSM order format.

related order

An order that contains order items that depend on another order.

reporting interface

A tool for generating reports about OSM orders, tasks, and notifications. The Reporting Interface augments the reports that are available through the OSM web clients. See *OSM Reporting Interface Guide* for more information.

requested delivery date

The date on which an order is requested to be delivered.

resource-facing service (RFS)

A service as it is implemented on the network; for example, an ADSL service. By contrast, a customer-facing service (CFS) is the service that the customer purchased and would recognize; for example, Internet service.

revision order

An order that modifies a previously submitted order that is still being processed. For example, a customer may want to switch to a higher level of service before an order is completed. Revision orders may require compensation. The system can process revision orders until the original order reaches its point of no return. A revision order is sometimes called a supplemental order.

See also follow-on order.

RFS

See resource-facing service (RFS).

role

A set of permissions to access functions in the Task web client and Order Management web client that can be assigned to users. Functions include viewing reports, assigning tasks, and querying orders. In addition to granting OSM web client permissions, you can also grant permissions at the order and task levels. Roles are created by using Design Studio. The Administration area of the Order Management web client refers to roles as workgroups, although they are both the same thing.

rule

Rules are defined as part of an order specification to work on data in the order. Rules are used in many OSM activities to evaluate conditions and determine next process steps. For example, you can specify to delay the next task in a process until a specified data element includes a certain value.

rule engine

An OSM processing component that evaluates rule and timer delays for transition to the next task. The engine is implemented as one or more Oracle database jobs. The rule engine is configured as one or multiple jobs to improve performance.

sales order

An order received by OSM to obtain a product or products, typically generated by a CRM system or other order-source system. OSM converts the sales order to OSM format after which it is referred to as a customer order or as an order.

A sales order is sometimes called an inbound order.

security callback interface

See OSM security callback.

service order

An order processed by an OSM instance acting in the service order management (SOM) role. Service orders work with service and resource management (SRM) systems to design services and assign resources.

service order management (SOM)

The OSM system role that processes the service orders. Service orders work with service and resource management (SRM) systems to design services and assign resources. OSM in the

SOM role receives orders from OSM in the central order management (COM) role, and sends orders to OSM in the technical order management (TOM) role.

simple data element

Reusable data types that contain no child dependencies. A simple data element has no structure, and is associated (directly or indirectly) to a primitive type (int, boolean, char, and so forth).

SOM

See service order management (SOM).

specification

A blueprint that defines the composition of an entity, including the attributes and relationships between an entity and other objects. There are different types of specifications for different types of entities, such as telephone numbers, networks, customer-facing services, and resources. Specifications are defined in Design Studio and deployed into runtime environments, where entities can be created based on them.

In OSM, this may refer specifically to an order specification.

structured data element

Reusable data types that include embedded data types and are containers of simple data elements and other structured data elements.

subprocess

A process started by another process. A subprocess is used to organize any large process into smaller more re-usable pieces.

task

An individual step that is required for the processing of an order. Tasks are defined by the order specification in Design Studio, and can be either a manual task (performed by human action) or automated task (performed by an automation plug-in).

task state

A state describing the milestones of a task in a process. The task state also determines how it can be worked on. OSM provides the following task states: Received, Assigned, and Accepted. You can, however, create your own task states. For example, you can define a Suspended task state to indicate the progress of automated tasks, or you can define a Completed task state to indicate that user is finished with the task and the order is ready to move to the next task in the process.

task status

A representation of how a task can transition to the next step in a process. The task status shows how the task transitions in a process; for example, if the task transitioned the process to the next task, or if it caused the process to fail. Changing the status of the task determines the next step in the order process. The statuses that you define appear as task transition options in the OSM web clients.

For example, if you have a task called Assign Port, and the two statuses are Port Available and Port Unavailable, the status determines whether the process can proceed to the next task. Task status also controls notifications, so when the status is Port Available, OSM can send a message saying Successful.

Task web client

An OSM GUI application used for monitoring and managing the tasks in an order. This application is typically used by order processing personnel to ensure that all the tasks are completed. It is also used by order fallout managers. You can also suspend and resume orders, cancel orders, and create orders manually.

technical order

An order processed by OSM in the technical order management (TOM) role. Technical orders work with external systems to implement activation, shipping, installation, and other fulfillment actions.

technical order management (TOM)

The OSM role that processes technical orders. Technical orders work with external systems to implement activation, shipping, installation, and other fulfillment actions. OSM in the TOM role receives orders from OSM in the service order management (SOM) role.

timer delay

See delay.

TOM

See technical order management (TOM).

transaction

An action taken by the OSM system on an order. For example, the Suspend Order transaction stops all processing on the order and transitions the order to the *Suspended* state. Also called an order state transaction.

Some other transactions are Abort Order, Complete Task, Process Amendment and Raise Exception. Most transactions perform transitions that change the state of the order to a different state. However, some transactions do not perform a transition to another state. For

example, the Update Order transaction can make changes to an order without changing the order's state.

transformation

See order transformation and order item transformation.

trouble ticket (TT)

A request to the trouble ticketing system indicating that an error occurred during the processing of an order. Different from a fallout task in that trouble tickets come from front-end systems such as Siebel CRM.

unresolved dependency

A dependency with at least one unmet condition.

validation

See order validation.

WebLogic Server

See Oracle WebLogic Server.

workgroup

A group of users with assigned permissions to access functions in the Task web client and Order Management web client. Functions include viewing reports, assigning tasks, and querying orders. In addition to granting OSM web client permissions, you can also grant permissions at the order and task levels. Workgroups are created with Design Studio and managed using the Administration area of the Order Management web client. Design Studio refers to workgroups as roles, although they are the same thing.

worklist

A list of manual tasks assigned to OSM operations personnel who use the Task web client to manage orders. When an order arrives at a task, it is added to the worklist of all the members of all the workgroups assigned to work on that task. Users can select a task from their worklist to view the assigned task in the order process. Worklist also refers to the main page in the Task web client used for managing orders.