Oracle® Communications Service Catalog and Design

Design Studio Modeling OSM Orchestration

Release 8.1
F96245-01
July 2024

Oracle Communications Service Catalog and Design Design Studio Modeling OSM Orchestration, Release 8.1

F96245-01

Contents

22 Working with Order Item Parameter Bindings

23 Working with Transformation Sequences

24 Working with Transformation Managers

25 Working with Mapping Rules

Preface

This document contains information about the procedures and tasks that are necessary to configure and deploy Oracle Communications Order and Service Management (OSM) process entities and cartridges using Oracle Communications Service Catalog and Design - Design Studio.

Audience

This guide is intended for business analysts, architects, development managers, developers, and designers who are responsible for system integration or solution development involving the Oracle Communications operational support systems applications.

Ideally, you should be knowledgeable about your company's business processes, the resources you need to model, and any products or services your company offers.

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

Working with Orchestration Cartridge Projects

You create cartridge projects for Oracle Communications Order and Service Management (OSM) orchestration to contain data related to the transformation, decomposition, and orchestration of orders and activities (service and network provisioning) both manual and automated to streamline the delivery of services. You can configure entities and resources from the cartridge that are included in the generated cartridge archive file, modify existing cartridge properties, and define categories.

The OSM cartridge project is the working area for the OSM orchestration configuration. Modeled data is stored in a data schema, and is used to create the order templates. You package cartridges as PAR files, which contain all configured elements and components required for deployment.

When modeling orchestration cartridge projects, refer to the following topics:

- Creating New Cartridge Projects
- Closing Projects
- Opening Projects
- Managing Project Dependencies
- Working with Existing OSM Models
- Order and Service Management Project Editor

2

Getting Started with Design Studio for OSM Orchestration

Order management is the transformation, decomposition, and orchestration of orders and activities (both manual and automated) to streamline the delivery of services. As an order management system, Oracle Communications Order and Service Management (OSM) incorporates business systems and business entities that deal with customers, products, customer orders, and supporting processes (such as billing and payments). OSM receives and recognizes customer orders from order capture systems and coordinates the actions to fulfill the order line items across provisioning, shipping, inventory, billing and other edge fulfillment systems.

Design Studio for OSM is a design-time configuration application that simplifies the delivery and management of services across multiple systems and work groups.

When modeling orchestration, you can configure in Design Studio the data needed to translate customer orders from disparate input systems, and to coordinate the actions necessary to fulfill the order across the provisioning, shipping, inventory, billing, and other edge fulfillment systems.

Related Topics

About OSM Users

Creating a Cartridge

About OSM Users

The following is a list of roles who typically use Design Studio to model and deploy OSM orchestration.

Service Administrator

Service administrators have in-depth knowledge of Operation Support Systems (OSS) configurations, internal fulfillment applications, third-party fulfillment applications, and service creation environments. They understand how to model the technical configurations associated with products (in consultation with network subject matter experts and product administrators), and they are responsible for fulfillment topology and task status configurations.

Service administrators typically have experience with service modeling, OSS configurations, service dependencies, and deployment topology (including interactions with partner OSS systems) and some coding skills. Service administrators understand products, services, resources, and network elements and can model these when required.

Service Configuration Testing Specialist

Service configuration testing specialists are responsible for testing the configurations produced by service administrators when new configuration builds are available. Testing specialists interact with service administrators to understand aspects of the configuration and report and resolve issues (discovered through testing) before a configuration goes into production. Service configuration testing specialists are technical users of Design Studio, and they are

skilled in the testing and deployment capabilities provided by Design Studio. They understand technologies such as web services, XML, and Java.

Provisioning Service Modeler

Provisioning service modelers understand how to provision a service. This user typically designs the higher level artifacts within a traditional OSM cartridge (for example, the order template, process flows, and tasks). Provisioning service modelers are not necessarily technical users with in-depth knowledge of J2EE technologies (such as Java, XSLT, XQuery, or XML schema). Rather, this user understands the high-level business objectives and has a detailed understanding of the network and system processing that is needed to provision a service.

Order Fallout Management Configurator

Order fallout management configurators are responsible for the configuration of the Order Fallout Management component, including minimal configuration required to seed applications, load application errors, and map application errors to common errors and messages. This user possesses the configuration skills necessary for notification and escalation of order fallout.

The order fallout management configuration has advanced knowledge of OSM, and will understand how to map application errors and messages to common errors and messages. This user may work with the CRM administrator to configure notifications and escalations, and may possess some CRM experience.

Creating a Cartridge

There are multiple types of modeling schemes you can use when configuring cartridges for orders that use orchestration. You can use:

- A top-down modeling method (a product-oriented approach) where you configure services using one or more orchestration fulfillment patterns as the basis for the configuration.
- A bottom-up modeling method (a topology-centric approach), where you first model services in Design Studio (with no associated fulfillment patterns). Use this method when products and their associated attributes are unknown early in the configuration cycle. You can reconcile the services with the fulfillment patterns (mapped from conceptual model products or OSM orchestration product specifications from the primary data source) as they become available.

The procedure below outlines the basic best practice steps to configure an OSM cartridge in Design Studio.

To create a cartridge for orders that use orchestration:

1. Create an OSM cartridge project.

   The OSM cartridge project is your working area for the OSM configuration. When you create a new cartridge project, Design Studio displays the new cartridge project in the Studio Projects view of the Design perspective. In addition to the newly-generated entity, the project contains entities for an order and for the data schema (which becomes part of the Design Studio Data Dictionary). See "Working with OSM Cartridge Projects" for more information.

2. Model the incoming orders.

   To recognize the source of in-bound orders and interpret the order's structure, OSM uses an ordered list of recognition rules. Additionally, you must determine how to select order items on the incoming order, what data you need from the order item to fulfill the order

item, and what control data from the order item you need to help create the orchestration plan.

To model incoming orders:

a. Define recognition rules.

   Recognition rules help determine the order type and source of the incoming order and what validations and transformations, if any, are required. When you define recognition rules, you configure XQuery expressions for the recognition and validation rules, specify the data structure of the incoming customer order message, and map the incoming order to a specific order type. See "Working with Recognition Rules" for more information.

b. Define order item specifications.

   For each type of order that you receive from your order capture systems, you must configure one order item specification to identify the properties that help determine how to fulfill those items. See "Working with Order Items" for more information.

3. Model the decomposition.

   To ensure that OSM can properly direct order items on incoming orders to the appropriate downstream processes, you configure multiple order components and rules to enable OSM to sort the order items at increasingly greater levels of detail. This process is called decomposition.

   To model the decomposition of an order:

   a. Define the order components.

      Order components represent interactions with local fulfillment systems, and contain all of the data needed by the downstream system subprocesses to complete the fulfillment task. Defining order components is an iterative process. At this step, you identify (at minimum) the top level order components (those that represent your fulfillment functions) that you'll reference when you create orchestration stages. If (at this point in the design time) information about your systems is in place, you can also create the order components that represent systems.

      See "Working with Order Components" for more information.

   b. Define the orchestration process.

      Orchestration processes are dynamically generated at run time, based on process data that you configure in Design Studio. Orchestration processes enable OSM to fulfill incoming customer orders using fulfillment systems required to complete the order. See "Working with Orchestration Processes" for more information.

   c. Define the stages.

      Orchestration stages are the logical steps OSM must take to ensure that the order items on an incoming order are sent to the proper systems in a sequence that honors any dependencies among the order items. See "Working with Orchestration Stages" for more information.

   d. Define the orchestration sequence.

      Orchestration sequences describe how OSM organizes the order items on incoming orders into hierarchies to be sent to external systems. Orchestration sequences include information about how to locate the order items on an incoming order and the set of orchestration stages through which the sequence transitions. See "Working with Orchestration Sequences" for more information.

   e. Define decomposition rules.

Decomposition rules specify the conditions in which OSM decomposes one or more order components into another order component. OSM evaluates every order item in the source order component against the conditions that you define for the decomposition rule. If an order item passes all specified conditions, OSM includes the order item in the target order component. See "Working with Decomposition Rules" for more information.

f. Define which order components are executable.

   Executable order components contain all of the order items necessary to run an instance of the subprocess task associated with the order component. See "Order Component Specification Editor Details Tab" for more information about associating a process with an order component and specifying which order components are executable.

4. Define the orchestration fulfillment patterns.

   You use orchestration fulfillment patterns to describe the structure of a group of related products in your product catalog. The structure describes the characteristics that define the data collected during order capture. See "Working with Orchestration Fulfillment Patterns" for more information.

   When configuring orchestration fulfillment patterns, you:

   a. Define the order component transitions.

      Transitions specify the logical sequence between order components within a single fulfillment pattern, or between order components associated with different fulfillment patterns. Transitions describe a decomposition from a source order component to a target order component. See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for more information.

   b. Define order item dependencies among order components.

      When OSM receives a customer order from an external system, it determines all dependencies that exist among the order items on the order. You must define the order item dependencies among the order components in one fulfillment pattern and among those in multiple fulfillment patterns. See "Working with Order Items" for more information about creating order item dependencies. See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for information about keying the dependency on order item properties other than the order item name.

      You can also define dependencies between order items within a set of order components that are not based on fulfillment patterns. See "Working with Orchestration Dependencies" for more information.

5. Create (or import) products and map to fulfillment patterns.

   Conceptual model products and OSM orchestration product specifications define a type or class of products (for example, DSL). New products should be created in the conceptual model and mapped to conceptual model fulfillment patterns. These conceptual model fulfillment patterns are then realized into OSM orchestration fulfillment patterns.

   Products and orchestration product specifications include dynamic attributes (characteristics) for a specific type of product. In Design Studio, you can connect to a web service to import conceptual model products (for example, from a CRM system or a product primary catalog) and their attributes, then map the products to existing fulfillment patterns. See "Importing Products" for more information. See "About Products" for information about creating new products in Design Studio.

6. (Optional) Model fulfillment states.

You use fulfillment states to provide an aggregated status of orders and order items. Fulfillment state functionality maps the statuses received from external systems into normalized statuses for the order components, order items, and the order itself.

When configuring fulfillment states, you:

a. Define external fulfillment states for order components.

   Create a list of values for the order component that matches the statuses returned by the external systems or automations. An external fulfillment state is available on the order component where it is defined and on any order component that extends that order component. See "Order Component Specification Editor External Fulfillment States Tab" for more information.

b. Create and configure fulfillment state maps.

   Create one or more lists of values for the common fulfillment states and create mappings to translate external fulfillment states into mapped fulfillment states. See "About Fulfillment State Maps" for more information.

c. Create and configure order item fulfillment state composition rule sets and order fulfillment state composition rule sets.

   Create the composition rule sets to determine the fulfillment state of an order or order item from the fulfillment state of its child items. Composition rule sets are based on the order item and order hierarchy, and compose fulfillment states into composite fulfillment states that reflect the state of entire order items or orders. See "About Order Item Fulfillment State Composition Rule Sets" and "About Order Fulfillment State Composition Rule Sets" for more information.

7. (Optional) Model order item transformation.

   The order transformation manager provides you with the ability to transform order items. For example, you can use order transformation manager to transform customer-focused order items (what the customer bought) to service-focused order items (the services that equate to what the customer bought). It enables you to set up guidelines for order transformation that do not need to be changed due to product changes. The easiest way to implement order item transformation is to use the functionality provided in design patterns and then customize it for your needs.

   When configuring order item transformation, you:

   a. Run the Model Project's **Common Model Base Data** design pattern.

      This provides you with the conceptual model entities that will help you to model your transformation.

   b. Create an order item specification to contain the transformed order items.

      See "Working with Order Items" for more information.

   c. Run the OSM **Calculate Service Order** design pattern and configure the transformation sequence.

      This creates a transformation sequence that is pre-configured with four stages. After running the design pattern, ensure that the configuration of the transformation sequence meets your needs. See "Working with Transformation Sequences" for more information.

   d. Configure the conceptual model services appropriate to your situation.

      In the conceptual model you configure entities to model the domains and services that you are provisioning. See "Working with Conceptual Models" for more information.

   e. Create a transformation manager.

Create a transformation manager to map your transformation sequence to the service domains within your provider function. See "Working with Transformation Managers" for more information.

f. Model the data for your transformation.

Use order item parameter bindings to map data from the conceptual model entity to the order item. The Order Item Parameter Binding design pattern is available to assist with this. See "Working with Order Item Parameter Bindings" for more information.

Use mapping rules to map the data from the original order items to the transformed order items. See "Working with Mapping Rules" for more information.

8. Test configurations and orders in Design Studio.

When you've completed modeling, you can test the model and review the configuration. To test a cartridge, you submit the order to an OSM environment and monitor the results in a web user interface. See "Deploying Cartridge Projects with Optimize Deploy" and "Packaging and Deploying OSM Cartridges" for more information.

9. Deploy the cartridge.

You package the cartridge with the entities, libraries, and resources that you want to include in the cartridge file. Then, you connect to the Design Studio environment to deploy your cartridges. See "Packaging and Deploying OSM Cartridges" for more information.

Related Topics

About OSM Users

Getting Started with Design Studio for OSM Orchestration

3

Working with Recognition Rules

Oracle Communications Order and Service Management (OSM) can recognize the source of any incoming XML-based order, transform each order from the incoming format to one that is recognized by OSM, enrich the data on the order, and validate the data, if required.

To recognize the source of in-bound orders and interpret the order's structure, OSM uses an ordered list of recognition rules that you configure in Design Studio. OSM evaluates each rule in an order that you define and selects the first rule that evaluates to true. The selected rule determines the order type and source of the incoming order and the required validations and transformations, if any.

When modeling recognition rules, refer to the following topics:

About Recognition, Validation, and Transformation Rules

OSM can recognize the source of any incoming XML-based order, transform each order from the incoming format to one that is recognized by OSM, enrich the data on the order, and validate the data, if required.

Recognition Rules

Recognition rules determine the order type and source of incoming orders. When OSM receives an incoming order from an external system, it first assembles a list of all currently active order recognition rules and evaluates each rule, in order of the defined relevance, until one rule evaluates to true.

Validation Rules

In addition to recognition rules, you can define validation rules that OSM runs against in-bound messages to confirm that the XML message is syntactically valid. Validation passes only when every data node returned evaluates to true.

Transformation Rules

Transformation rules enable the OSM server to transform the data structure and contents of the order into a form that OSM can recognize, and enrich the order data, when necessary. For example, you may use transformation to add information to the customer order data or to transform the generic order items and their attributes into specific attributes of an order in your provisioning system.

Creating New Recognition Rules

You create recognition rules to enable OSM to recognize incoming orders, transform the message into a format that OSM can process, determine whether the message is syntactically valid, and enrich the order data, if necessary.

To create new recognition rules:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Order Recognition Rule**.

   The Order Recognition Rule wizard is displayed.
2. In the **Project** field, select the OSM project in which to save the recognition rule.
3. In the **Name** field, enter a name for the recognition rule.

   The name must be unique among recognition rule entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the recognition rule.
5. (Optional) Select a location for the recognition rule.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

   Design Studio adds the recognition rule to the project in Studio Projects view and opens the rule in the Order Recognition Rule editor.

Related Topics

Configuring Recognition Rules

Order Recognition Rule Editor

Working with Recognition Rules

Configuring Recognition Rules

Use the Order Recognition Rule editor to configure and edit recognition rules.

To configure a recognition rule:

1. Select **Studio**, then **Show Design Perspective**.
2. In the Studio Projects view, double-click any recognition rule.

   The Order Recognition Rule editor is displayed.
3. In the **Input Message** field, specify the data structure of the incoming customer order.

   Do one of the following:

- To select a structure or node, right-click in the **Input Message** field and select **Select Input Message**.
- To drag a structure or node from the Dictionary view into the **Input Message** field, right-click in the **Input Message** field and select **Open Data Element View**.
- To select a structure or data node, click **Select**.

4. Associate the recognition rule with a single order type. Click:
   - **Select** to select from a list of all orders in the workspace.
   - **New** to create a new order to associate with the rule. See "Creating New Orders" for more information.
   - **Open** to review the order details in the Order editor.

   OSM creates an order of the type you select when the corresponding recognition rule evaluates to true.
5. Assign a relevancy to the rule.

   Relevancy determines the order in which the OSM server evaluates the order recognition rules at run time. OSM evaluates a rule with a higher relevance before it evaluates a rule with a lower relevance. For example, OSM evaluates a rule with a relevancy value of 2 before it evaluates a rule with a relevancy value of 1. OSM uses at run time the first order recognition rule that evaluates to true. The relevancy value can be any whole integer greater than or equal to zero.
6. (Optional) Click the relevancy **Select** button.

   The Edit Relevancy dialog box opens, which displays a list of all recognition rules and their relevancy values in the workspace. You can refer to the list to help determine the value for the new order recognition rule.
7. Click **Recognition Rule**.

   The Recognition Rule area expands to reveal the subtabs area.
8. Click the **XQuery** subtab.
9. Configure the XQuery-based rule or element.

   See "Order Recognition Rule Editor" for more information.
10. (Optional) Click **Properties**.

    The Recognition Rule Properties view opens, where you can define annotation and a description for the rule.
11. Click the **Instances** tab.
12. (Optional) Associate a Data Instance behavior with the recognition rule.

    You can define a Data Instance behavior to obtain data that is not included in the order data and make that data available to the recognition rule. Click:
    - **Add** to add a Data Instance behavior.
    - **Properties** to configure the Data Instance behavior.

    See "Defining Data Instance Behavior Properties" for more information.
13. Click the **Notes** tab.
14. Describe the intended use of the rule.

    For example, you might describe the functionality of a complex rule or provide instructions on its use.

15. (Optional) Click **Validation Rule**.

    The Validation Rule area expands to reveal the subtabs.

16. Repeat steps 7 through 14 for the Validation subtabs.

    See "Order Recognition Rule Editor Recognition Tab" for more information about validation rules.

17. (Optional) Click the **Transformation** tab.

18. (Optional) Configure order priorities, references, and data rules.

    Click **Order Priority**, **Order Reference**, and **Order Data** to expand each area.

    Repeat steps 8 through 13 to configure rules for each area. See "Order Recognition Rule Editor Transformation Tab" for more information about transformation rules.

Related Topics

Creating New Recognition Rules

Order Recognition Rule Editor

Working with Recognition Rules

Order Recognition Rule Editor

Use the Order Recognition Rule editor to configure order recognition rules.

- Use the **Recognition** tab to create the recognition and validation rules, associate the rules with an order, and define the relevancy. See "Order Recognition Rule Editor Recognition Tab" for more information.
- Use the **Transformation** tab to define the order priority, reference, and data rules. See "Order Recognition Rule Editor Transformation Tab" for more information.

The following fields are common to multiple tabs or areas in the Order Recognition Rule editor:

Field	Use
Description	Edit the display name of the rule.
Namespace	Enter the unique XML namespace in which to include the recognition rule.
Input Message	Specify the data structure of the incoming order. You can select from the Data Dictionary a previously imported XML structure or a node from a previously imported XML structure. At run time, when a recognition rule evaluates to true, OSM adds this input message to the order template (as an XML data type) to contain the incoming message.

Related Topics

Creating New Recognition Rules

Configuring Recognition Rules

Working with Recognition Rules

Order Recognition Rule Editor Recognition Tab

Use the **Recognition** tab to create the recognition and validation rules, associate the rules with an order, and define the relevancy.

Field	Use
Target Order	Specify an order with which to associate the recognition rule. Every recognition rule must be associated with an order. Click: • **Select** to select from a list of all orders in the workspace. • **New** to create a new order to associate with the rule. See "Creating New Orders" for more information. • **Open** to open the specified order in the Order editor.
Target Order Version	Select a specific version of an order if you want the recognition rule to recognize an order version other than the default version. To enter a specific order version, deselect the **Default** check box.
Fail Order Reason	Select the **Fail Order** check box, and enter a reason to display to Order Management web client users if this order has failed during validation or transformation. If the order fails, it is assigned the failed state and placed in fallout.
Relevancy	Assign a specific relevancy value to the rule. The value can be any whole integer greater than or equal to zero. The default is 5. Relevancy determines the order in which the OSM server evaluates the order recognition rules at run time. OSM evaluates a rule with a higher relevance before it evaluates a rule with a lower relevance. For example, OSM evaluates a rule with a relevancy value of 2 before it evaluates a rule with a relevancy value of 1. At run time, OSM uses the first order recognition rule that evaluates to true.
Recognition Rule	Enter an XQuery expression or a pointer to denote how OSM will recognize the incoming order. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. For example, in a simple scenario, you may have the recognition based on namespace: `fn:namespace-uri(.) = 'http://www.example.org/exampleNamespace'` Or, in a more complicated scenario, you might create an XQuery expression that looks for a specific namespace but also interrogates the data within the incoming order. See "About Recognition, Validation, and Transformation Rules" and "About Recognition Rule XQuery Expressions" for more information. Incoming orders that match no recognition rules are marked as failed orders and put into order fallout. See "Defining Order Fallout" for more information.

Field	Use
Validation Rule	Enter an XQuery expression or a pointer to denote how OSM will validate the incoming order. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. For example, the XQuery expression can parse the data to determine whether mandatory fields are populated on the order. See "About Recognition, Validation, and Transformation Rules" and "About Validation Rule XQuery Expressions" for more information. Incoming orders that do not pass validation are marked as failed orders and put into order fallout. See "Defining Order Fallout" for more information.

Related Topics

Creating New Recognition Rules

Configuring Recognition Rules

Order Recognition Rule Editor

Working with Recognition Rules

Order Recognition Rule Editor Transformation Tab

Use the **Transformation** tab to define the order priority, reference, and data rules.

Field	Use
Order Priority	Enter a digit, an XQuery expression, or a pointer to denote how OSM will set the priority of the incoming order. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. Enter a digit or define an XQuery that evaluates to an order priority number between 0 and 9 inclusive, with 9 being the highest priority. If you do not define a priority, the OSM server sets the order priority to 5. For example, you might define data-driven rules to set the order priority of incoming orders based on one or a set of property values in the incoming message. See "About Recognition, Validation, and Transformation Rules" and "About Order Priority XQuery Expressions" for more information.
Order Reference	Enter an XQuery expression or a pointer to a rule that evaluates to an order reference number. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. The order reference is a specific order header field on an OSM order. It can be used for storing cross-referencing information from other systems. The Order Reference transformation generates data for this header field. See "About Recognition, Validation, and Transformation Rules" and "About Order Reference XQuery Expressions" for more information.

Field	Use
Order Data Rule	Enter an XQuery expression or a pointer to a rule that augments incoming order data. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. For example, to add data to the order that is not included in the input message, you can configure XQuery rules to extract that data and add it to the creation task. **Note:** The Order Data Rule is for generating creation task data that is outside of the ControlData structure. See "About Recognition, Validation, and Transformation Rules" and "About Order Data Rule XQuery Expressions" for more information.

Related Topics

Creating New Recognition Rules

Configuring Recognition Rules

Order Recognition Rule Editor

Working with Recognition Rules

4

Working with Order Items

Oracle Communications Order and Service Management (OSM) uses order items to process the order line items from the customer order. Order items consist of an action and a subject and represent the intent of the customer order. For example, customers want to add (the action) DSL service (the subject). Or they want to delete, update, suspend, resume, or move some existing service. Additionally, order items include subjects that are not limited to a specific service. For example, the order item may refer to a product, discount, bundle, offer, or pricing event.

For each type of order that you receive from your order capture systems, you must configure one order item in Design Studio specification to identify the properties that help determine how to fulfill those items.

When modeling order item specifications, refer to the following topics:

- About Modeling Order Item Control Data
- Modeling Order Item Control Data
- Modeling Transformed Order Item Control Data
- Creating New Order Item Specifications
- Order Item Specification Editor

About Modeling Order Item Control Data

The order template of an order item specification must have a reserved area, the **ControlData/OrderItem** structure, that is used by decomposition and orchestration functions. The **Control Data/OrderItem** structure is populated by orchestration plan generation with the order items from the sales order and is based on data you define for the **Order Item Properties** tab in the Order Item Specification editor. OSM uses the order item specification to generate order items into the **ControlData/OrderItem/** structure to include all of their properties from the in-bound message.

The **OracleComms_OSM_CommonDataDictionary** model project contains a predefined data schema that defines the **ControlData** structure and its data elements. If you have this model project in your workspace, you can use its predefined data schema when modeling the **ControlData/OrderItem/** structure (you can drag the **ControlData/OrderItem/** structure from the data schema onto the **Order Template** tab of the Order Item Specification editor). Otherwise, you must manually define the data element information for the structure in the data schema of whichever cartridge project you choose to include these data elements and then drag the **ControlData/OrderItem/** structure (you manually defined) from that data schema onto the **Order Template** tab of the Order Item Specification editor.

See "Modeling Order Item Control Data" for instructions on modeling order item control data.

Order components also require a data structure for control data. See "About Modeling Order Component Control Data" for more information.

Related Topics

Working with Order Items

Order Item Specification Editor

About Modeling Order Component Control Data

Modeling Order Item Control Data

To model order item control data:

1. Do one of the following:
   - If you are using the **OracleComms_OSM_CommonDataDictionary** model project (recommended), verify it is in your workspace.

     If you do not have the model project in your workspace, that means you earlier dismissed and disabled the prompt to create it. To re-enable the prompt and accept it, see the discussion on orchestration preferences in "Defining OSM Preferences".
   - If you are not using the **OracleComms_OSM_CommonDataDictionary** model project, define the data element information for the **ControlData/OrderItem/** structure in the data schema of whichever cartridge project you choose to include these data elements, using these guidelines:
     - Configure the **ControlData/OrderItem/** structure as multi-instance with element constraint values of **Minimum** set to **0** and **Maximum** set to **unbounded** so that OSM creates an instance of the structure for every line item extracted off the customer order.
     - Set the maximum string length of **ControlData** to a reasonable value; the default value is often too low.
2. Create the order item specification for the orchestration order.

   See "Creating New Order Item Specifications".
3. Do one of the following:
   - If you are using the **OracleComms_OSM_CommonDataDictionary** model project, drag the **ControlData/OrderItem/** structure from its data schema onto the **Order Template** tab of the Order Item Specification editor.
   - Drag the **ControlData/OrderItem/** structure from the data schema where you manually defined it onto the **Order Template** tab of the Order Item Specification editor.
4. In the **Order Template** tab of the Order Item Specification editor, do the following:
   - For every order item property you defined in the **Order Item Properties** tab, define a storage area; for example, if you defined **lineId** and **lineItemName** as order item properties, you would define the storage areas **ControlData/OrderItem/lineId** and **ControlData/OrderItem/lineItemName**. See "Order Item Specification Editor Order Item Properties Tab" for defining order item property.
   - Use the exact spelling and case of the order item property name for its defined storage area in the order template (**ControlData/OrderItem/***Order_Item_Property_Name*).
   - Ensure the data schema element properties support the property values, such as type (string, number, structure) and maximum length.
   - Map line items that are qualifiers to a product on a parent line item to the parent product or orchestration product specification. For example, map a QoS line item to the parent orchestration product specification of a broadband product. The QoS line item is generated as an order item, but it must contain a reference to its parent through a parent key identifier.
5. Save the order item specification.

6. Include the **ControlData/OrderItem** structure in the creation task.

Related Topics

Working with Order Items

About Modeling Order Item Control Data

Modeling Transformed Order Item Control Data

If you are using the order transformation manager, you need to model the **ControlData/TransformedOrderItem** structure in addition to the **ControlData/OrderItem** structure. The method for modeling this structure is different from that for modeling the regular order item structure.

To model transformed order item control data:

1. Enable the order transformation manager, either in the orchestration process or by adding a transformation task.

   New validation errors about missing data nodes will be generated in the Problems view.

   See "Orchestration Process Editor" for more information about enabling order transformation manager in the orchestration process. See "Working with Tasks" for more information about adding a transformation task.
2. Select any one of the validation errors relating to the missing control data, right-click on it, and select **Quick Fix**.

   The Quick Fix window is displayed.
3. In the **Select a fix** field, ensure that **Quick Fix: Add transformedOrderItem, componentKey, duration and calculatedStartDate to Control Data** is selected.
4. Click **Select All** to select all of the related problems.
5. Click **Finish**.

   The Add Control Data Quick Fix window is displayed indicating what actions will be taken.
6. Click **OK**.

   The appropriate data elements are added to the OracleComms_OSM_CommonDataDictionary, inherited by the order, and added to the creation task for the order.

Related Topics

Working with Order Items

About Modeling Order Item Control Data

Creating New Order Item Specifications

To create new order item specifications:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Order Item Specification**.

   The Order Item Specification wizard is displayed.
2. In the **Project** field, select the project in which to save the order item specification.

3. In the **Name** field, enter a name for the order item specification.

   The name must be unique among order item specification entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the order item specification.

   Design Studio uses the last saved namespace as the default.

5. (Optional) Select a location for the order item specification.

   By default, Design Studio saves the order item specification to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Decide whether to use the distributed order template for the order item specification and select or deselect the Support Distributed Order Template accordingly. If the order item specification is going to contain transformed order items, you must select this option. See *OSM Modeling Guide* for more information about the distributed order template and the order transformation manager. See "Referencing Items from a Distributed Order Template in XQuery Expressions" for more information about the format for referencing items in a distributed order template from XQuery expressions.

7. Click **Finish**.

   Design Studio adds the new order item specification to the Solution view and Studio Projects view and opens the new entity in the Order Item Specification editor.

Related Topics

Order Item Specification Editor

Working with Order Items

Working with Orchestration Dependencies

Order Item Specification Editor

Use the Order Item Specification editor to model the properties, applicable conditions, hierarchies, and permissions.

The order item specification that you configure in Design Studio contributes to the control data that OSM adds to the created order. The control data provides the information OSM requires to run the orchestration plan and includes information about the order items and the order components in the orchestration plan.

When modeling order items, refer to the following topics:

- Order Item Specification Editor Order Item Properties Tab
- Order Item Specification Editor Property References Tab
- Order Item Specification Editor Orchestration Conditions Tab
- Order Item Specification Editor Order Item Hierarchies Tab
- Order Item Specification Editor Order Template Tab

- Order Item Specification Editor Order Item Dependencies Tab
- Order Item Specification Editor Permissions Tab

The following fields are common to multiple tabs in the Order Item Specification editor:

Field	Use
Description	Edit the display name of the order item.
Namespace	Select an existing namespace or enter a unique namespace in which to include the order item.
XQuery	The **XQuery** tab enables you to configure XQuery-based rules or elements or identify the source of the XQuery-based rules or elements. Enter an XQuery expression or a pointer for a condition for the mapping rule to be used. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. Click **Properties** to open the Order Item Specifications Properties view, where you can define annotations for the rule. See "XQuery Examples" for more information.
Instances	Define a Data Instance behavior to obtain data that is not included in the order data and make that data available to the XQuery expression. Click **Add** to add a Data Instance behavior. Select the Data Instance behavior, and click **Properties** to configure the Data Instance behavior. See "Defining Data Instance Behavior Properties" for more information.
Information	Describe the intended use of the XQuery expression. For example, you might describe the functionality of a complex expression or provide instructions on its use.

Related Topics

Creating New Order Item Specifications

Working with Orchestration Dependencies

Working with Order Items

Order Item Specification Editor Order Item Properties Tab

Use the **Order Item Properties** tab to define the properties on the order item required during the orchestration plan generation.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Order Item Properties	Configure the order items required during the orchestration plan generation. Order item properties are name/value pairs. You configure an XQuery expression to extract from the source element the value of the property. Click: • **Add** to enter the name of a new property. • **Remove** to delete a property. **Note**: Order Item Properties cannot be Data Structure Definitions, which means that the name of the Order Item Property cannot be the same as the name of a Data Structure Definition within ControlData/OrderItem in the order template.
Location	In the **Location** field, you can map the corresponding property to a specific data node under ControlData/OrderItem in the order template. The node to which you map the property does not have to have the same name as the property itself. Click **Select** to select a data node from the order template.
Property Expression	You configure an XQuery expression to run against the order item property and extract the property value. Additionally, you can define a Data Instance behavior to obtain data that is not included in the order data and make that data available to the XQuery expression. See "Order Item Specification Editor" and "About Order Item Specification Order Item Property XQuery Expressions" for more information.

Related Topics

Working with Orchestration Dependencies

Creating New Order Item Specifications

Working with Order Items

Order Item Specification Editor Property References Tab

Use the Order Item Property References tab to map order item properties to map order item properties to specific functions.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Support Distributed Order Template	Select this option to use the distributed order template for this order item. If you are using order transformation manager and this order item specification is going to be used for transformed order items, this option must be selected. For all other order items, it is optional. See *OSM Modeling Guide* for more information about the distributed order template and the order transformation manager. See "Referencing Items from a Distributed Order Template in XQuery Expressions" for more information about the format for referencing items in a distributed order template from XQuery expressions.

Field	Use
Fulfillment Pattern Mapping Property	Select the property that contains the orchestration fulfillment pattern for the corresponding line item. Available options include the properties you defined in the **Order Item Properties** field.
Order Item Name Property	Select the order item property that contains the name of the order item as it should appear in the Order Management UI. Available options include the properties you defined in the **Order Item Properties** field. The order item name is displayed in the Order Management web client wherever the order item is displayed. For example, the order item name is displayed in the Order Info region and the **Orchestration Plan** tab in the Order Details page.
Requested Delivery Date Property	Select the order item property that contains the date by which the line item must be fulfilled. Available options include the properties you defined in the **Order Item Properties** field. OSM uses the requested delivery date to determine processing start times and dependencies.
Order Item Recognition Property	Select the order item property that identifies the PSR Entity specified by the customer order line item. Available options include the properties you defined in the **Order Item Properties** field. This property will be used to identify the order item parameter binding and add the correct parameters to the order item specification.
Order Item Action Property	Select the order item property that identifies the action type of the order item. Available options include the properties you defined in the **Order Item Properties** field.
Order Item ID Property	Select the order item property that specifies a unique identifier for the order item, for example, Asset Integration ID. This property should remain the same across revisions. Available options include the properties you defined in the **Order Item Properties** field.
Dynamic Parameter Property	Select the order item property in which dynamic parameters for the order item should be stored.

Order Item Specification Editor Orchestration Conditions Tab

Use the **Orchestration Conditions** tab to associate with the order item specification those conditions that return true or false values. OSM uses the conditions to determine with which order components the order item is associated.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Orchestration Conditions	Associate Boolean conditions with the order item specification. Click: • **Add** to enter the name of new condition. For each condition, you must configure an XQuery expression. • **Remove** to delete the condition from the order item specification.

Field	Use
Condition Expression	Configure an XQuery expression to specify a Boolean condition that must evaluate to true. You can reference these conditions from decomposition rules and orchestration dependencies. Additionally, you can define a Data Instance behavior to obtain data that is not included in the order data and make that data available to the XQuery expression. See "Order Item Specification Editor" and "About Order Item Specification Condition XQuery Expressions" for more information.

Related Topics

Working with Orchestration Dependencies

Creating New Order Item Specifications

Working with Order Items

Order Item Specification Editor Order Item Hierarchies Tab

Use the **Order Item Hierarchies** tab to define the relative hierarchies among order items.

An order item hierarchy represents a logical parenting of order items in the order. Incoming customer orders may contain multiple order item hierarchies. For example, a single order item may exist in multiple hierarchies.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Physical Hierarchy	Select the hierarchy for OSM to use for the order items on the order, or delete the selection to process the order item without a hierarchy.
Composition Hierarchy	Select the hierarchy for OSM to use in determining fulfillment states, or delete the selection for OSM to use only the order item's order components to determine the order item fulfillment state.
Dependency Hierarchy	Select the hierarchy for OSM to use in determining dependencies between order items, or delete the selection for OSM to use only the order item's order components to determine the order item dependencies. You cannot select Use for Child Completion Dependency with the Dependency Order Item Hierarchy. Design Studio generates an error message if you do.
Order Item Hierarchies list	Associate the order item with one or multiple hierarchies. Click: • **Add** to enter the name of a new hierarchy. For each hierarchy, you must configure an XQuery expression to select the key value and parent key value for the hierarchy. • **Remove** to delete the hierarchy from the order item specification.

Field	Use
Key Expression	Specify an XQuery expression to select the key value for the corresponding order item within a specific hierarchy (for example, within a line number). See "Order Item Specification Editor" and "About Order Item Specification Order Item Hierarchy XQuery Expressions" for more information.
Parent Key Expression	Specify an XQuery expression to select the parent key value for the corresponding order item within a specific hierarchy (for example, within a parent line number). See "Order Item Specification Editor" and "About Order Item Specification Order Item Hierarchy XQuery Expressions" for more information.

Related Topics

Working with Orchestration Dependencies

Creating New Order Item Specifications

Working with Order Items

Order Item Specification Editor Order Template Tab

Use the **Order Template** tab to model the data that OSM requires for orchestration; this data is referred to as order item control data. You can drag data from the Dictionary view into the Order Template area, or you can right-click in the Order Template area to select data from the Data Dictionary dialog box.

The recommended data schema to use to model order item control data is the predefined data schema of the **OracleComms_OSM_CommonDataDictionary** model project. See "About the OracleComms_OSM_CommonDataDictionary Model Project" for information on creating this project in your workspace.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Order Template	Add all of the data necessary to create orchestration plans that will include this order item. There is a minimum set of control data that you must define for a cartridge to build properly. For each order item, you must create a structure called **ControlData**, which contains a child structure called **OrderItem**. See "About Modeling Order Item Control Data" and "Modeling Order Item Control Data" for information on how to model the **ControlData/ OrderItem** structure. The following are examples of data nodes you can model below **OrderItem**: • **itemName** • **menuItemType** • **productSpec** • **size** The data you add here contributes to the order template data (added to the order template of the order as read-only data).
Behaviors	(Optional) Define behaviors for each data node in the order template. Select a data node in the Order Template area to view the behaviors defined for the node or to create new behaviors. See "Working with Behaviors" for more information.

Related Topics

Working with Orchestration Dependencies

Creating New Order Item Specifications

Working with Order Items

Order Item Specification Editor Order Item Dependencies Tab

Use the **Order Item Dependencies** tab to configure order item dependencies between order items on different orders.

See "Working with Orchestration Dependencies" for more information about configuring order item dependencies that are not based on orchestration fulfillment patterns.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Order Item Dependencies	Associate the order item with one or multiple dependencies with order items on different orders. Click: • **Add** to enter the name of a new dependency. For each dependency, you must configure an XQuery expression to select the order item on which the corresponding order item depends. • **Remove** to delete the dependency from the order item specification.
Order Item Selector Expression	Specify an XQuery expression to select the order item on which the corresponding order item depends. See "Order Item Specification Editor" and "About Order Item Inter-Order Dependency XQuery Expressions" for more information.

Related Topics

Creating New Order Item Specifications

Working with Order Items

Order Item Specification Editor Permissions Tab

Use the **Permissions** tab to permit specific roles access to order item search queries in the Order Management web client and define the data set that their queries return.

> **Note:**
>
> See "Order Item Specification Editor" for information about fields that appear on multiple Order Item Specification editor tabs.

Field	Use
Roles	Specify which roles can query order item information in the Order Management web client. Click: • **Select** to add an existing role to the list in the **Roles** field. • **New** to create a new role to add to the list. See "Creating New Roles" for more information. • **Open** to review the role in the Role editor. • **Remove** to delete the role from the list.

Field	Use
Query Tasks	Select the tasks that will generate the query view used by Order Management web client users. At run time, the OSM server returns a specific set of data when you use the search query functionality in the Order Management web client. You determine which data set the OSM server returns by creating or selecting query tasks. The data associated with the tasks that you select here will be the data returned to you from the run-time query. You can select tasks that you use in processes, or you can create tasks that you use only for run-time queries. For each role you add to the list in the **Roles** field, you can define two types of query task views: a summary and detail view. Select a task in the **Roles** field, then click: • **Add** to add an existing task to the **Query Tasks** list. • **New** to create a new task to add to the list. See "Creating New Tasks" for more information. • **Open** to review the task in the Task editor. • **Remove** to delete the task from the list. For each task in the list, specify whether to include the task data in the summary query view or the detail query view. See "Working with Roles" for more information about setting up new roles.
Summary	Select to display the corresponding task data set in the Order Management web client **Summary** tab. The **Summary** tab provides a selection of the most important information about the selected order, component, or item, and it is displayed when you open the Order Details page. You can include data from multiple query tasks in the **Summary** tab. The Order Management web client displays on the **Summary** tab all of the data from all of the tasks for which you specify the **Summary** option.
Details	Select to use the task data set as a view in the Order Management web client **Data** tab. The tasks for which you select this option appear as choices in the Order Management web client **Data** tab **View** field. You can specify that multiple tasks appear as options in the **View** field; each option will present the web client user with a different view, each containing a specific set of data.

Related Topics

Working with Orchestration Dependencies

Creating New Order Item Specifications

Working with Order Items

5

Working with Order Components

Oracle Communications Order and Service Management (OSM) separates the order items on an incoming customer order into related groups, each group meeting some common criteria. These groups of related order items are called order components. An order component represents a system interaction with a downstream fulfillment system and contains all of the data needed by the downstream system subprocesses to complete the fulfillment task.

You create and model the specifications for each order component in Design Studio, including the service that the fulfillment system fulfills, the fulfillment topology, and the detail at which the system processes the order items.

When modeling order components, refer to the following topics:

About Order Components

Order components are containers for order items. The contents of an order component are determined by the evaluation of decomposition rules and by the orchestration fulfillment pattern that is associated with the order items on the incoming order. The order in which OSM creates order components and the manner in which OSM includes order items in order components is based on the orchestration sequence used to process the order. Order components that are executable are mapped to a static process configured in Design Studio.

Required Control Data Structures

Orchestration plan generation requires two structures which you must model at design time at the order component level and the order item specification level:

```
ControlData
   OrderItem
   Functions
      Order_Component_Name
         componentKey
         OrderItem
            orderItemRef
```

For information on how to model the **Control Data/OrderItem** structure, see "About Modeling Order Item Control Data".

For information on how to model the **ControlData/Functions** structure, see "About Modeling Order Component Control Data". Design Studio automatically creates this structure if auto generation of order component control data is enabled. See "About Autogeneration of Order Component Control Data".

For general information about modeling control data, see "About Modeling Control Data."

Order Component Control Data

When OSM finishes generating the plan to orchestrate for the incoming order, it produces a set of order components, one per unique combination of function, fulfillment system, and processing granularity. Each order component contains the appropriate set of order items and dependencies between order items within that order component and order items on other order components.

In addition to the order components, OSM produces a set of control data. The control data, unique to each incoming order, provides information OSM requires to fulfill the order. For example, OSM uses the control data to track the status of the entire order and to track the status of the individual order items. During fulfillment, order component transactions update this control data with responses from the external systems used to fulfill line items.

Order component control data is configured by modeling the **ControlData/Functions** structure in the **Order Template** tab of the Order Component Specification editor. The **OracleComms_OSM_CommonDataDictionary** model project contains a predefined data schema that defines the **ControlData** structure and its data elements. If you have this model project in your workspace, Design Studio automatically creates the required order component control data for orchestration orders. For more information on the **ControlData/Functions** structure, see "About Modeling Order Component Control Data".

Order Component Order Items

OSM uses reference nodes to manage order items across multiple components. Reference nodes are pointers to values contained in different data nodes, and they enable you to create information once and reuse it in multiple locations in your data model. Rather than copying the order item data to each component, OSM creates in the component control data a reference node back to the original order item.

You model the reference node for the order component control data in the **Order Template** tab of the Order Component Specification editor as **ControlData/Functions/** *Order_Component_Name*/**OrderItem/orderItemRef**. Design Studio automatically inserts the modeling of this reference node on the order component control data if you have the **OracleComms_OSM_CommonDataDictionary** model project in your workspace. For more information on the **ControlData/Functions** structure, see "About Modeling Order Component Control Data".

For more information about reference nodes, see "About Reference Nodes" and "Adding Reference Data Nodes".

Related Topics

About the OracleComms_OSM_CommonDataDictionary Model Project

About Modeling Order Component Control Data

Creating New Order Component Specifications

Order Component Specification Editor

Working with Order Components

Working with Orchestration Sequences

Working with Decomposition Rules

About the OracleComms_OSM_CommonDataDictionary Model Project

The **OracleComms_OSM_CommonDataDictionary** model project enables autogeneration of orchestration control data. This model project is managed as a sealed cartridge. Do not modify the data elements of its data schema. See "Working with OSM Cartridge Projects" for information on how to create this orchestration model project in your workspace.

The data schema of the **OracleComms_OSM_CommonDataDictionary** model project comprises predefined data elements of which control data structures are composed. The data schema includes the base **ControlData** structure and the data elements it references: base order item data elements and base function data elements.

You use the data elements of this model project to manually model control data for order items. Design Studio uses the data elements of this model project to automatically model control data for order components. When an order component is created and added to the orchestration fulfillment pattern that is part of the orchestration plan, Design Studio automatically includes the **ControlData** structure from this data schema into the order template of the order and order component. The structures and types defined in the data schema of this model project are used as base types. For example, the data schema contains a **FulfillmentFunctionType** data element that describes all of the required data needed for a fulfillment function. When you create a new function order component, Design Studio for OSM creates a new and separate data element with the name of the function order component that uses **FulfillmentFunctionType** as a base type. The new function structure (**ControlData/ Functions/***Order_Component_Name*) is stored in the data schema of the project in which the function order component is contained; that is, it is no longer stored in the data schema of the **OracleComms_OSM_CommonDataDictionary** model project.

The **OracleComms_OSM_CommonDataDictionary** model project works for existing cartridge projects where the **ControlData** structure is already defined. When you create the model project in your existing workspace, you enable Design Studio to automatically update your existing **ControlData** structure with data for order components newly created thereafter.

See "About Autogeneration of Order Component Control Data" for information on orchestration entities that must be set up for Design Studio to be able to automatically model order component control data.

About Modeling Order Component Control Data

The **Control Data/Functions** structure is populated by orchestration plan generation for every order component of the orchestration plan and is based on data defined on the **Order Template** tab in the Order Component Specification editor.

The **ControlData/Functions** structure is a data repository area that is required to process the fulfillment function; this data area includes the order items the decomposition process has generated for the fulfillment function.

The control data structure for order components is modeled automatically or manually depending on the following:

- If you have the **OracleComms_OSM_CommonDataDictionary** model project in your workspace (recommended), and if the order component is both associated with an

orchestration order and associated with an orchestration fulfillment pattern that is part of the orchestration plan, Design Studio automatically creates its control data structure.

- If you do not have the **OracleComms_OSM_CommonDataDictionary** model project in your workspace, you must manually model the control data structure for each order component that is associated to an orchestration fulfillment pattern that is part of the orchestration plan. You must also manually add a reference node to it that points back to the order item control data structure. See "Modeling Order Component Control Data Manually" for information on adding this reference node.

Design Studio automatically defines the data on the **Order Template** tab in the Order Component Specification editor by using the **OracleComms_OSM_CommonDataDictionary** model project. See "About the OracleComms_OSM_CommonDataDictionary Model Project" for information on this project.

If you do not use the **OracleComms_OSM_CommonDataDictionary** model project, you must manually define the simple and structured data elements for the **Control Data/Functions** structure in the cartridge project of your choice and then manually add the data from its data schema onto the **Order Template** tab of the Order Component Specification editor. Manually modeling the **Control Data/Functions** structure for an order component is not recommended and should only be required if you decide not to use the **OracleComms_OSM_CommonDataDictionary** model project (for example, to minimize the upgrade impact of existing cartridges).

See "Modeling Order Component Control Data Automatically" for information on how to ensure that Design Studio can model the order component control data automatically by using the **OracleComms_OSM_CommonDataDictionary** model project.

See "Modeling Order Component Control Data Manually" for instructions on how to model the order component control data manually.

> **Note:**
>
> You must model a structure for order item control data in addition to modeling a structure for order component control data. For information on modeling a structure for order item control data, see "About Modeling Order Item Control Data".

Related Topics

About Order Components

About Modeling Order Item Control Data

About Autogeneration of Order Component Control Data

When the **OracleComms_OSM_CommonDataDictionary** model project is in your workspace, Design Studio automatically generates control data structures for order components and associates them with the order provided the following conditions are met:

> **Note:**
>
> The following conditions reflect basic orchestration setup requirements for all orchestration-related cartridges. Order component control data is automatically created if you preconfigure your orchestration cartridges correctly.

- The orchestration setup of your cartridges is in place. For example, the following orchestration configuration has been modeled:
  - The order item specification is created and contains the base **ControlData/OrderItem** structure.

    When creating a new solution, the order item specification is typically created before order components are created. You must create or import at least a skeleton order item specification that contains the base **ControlData/OrderItem** structure; this allows the autogeneration of a required reference node (**ControlData/Functions/***Order_Component_Name***/OrderItem/orderItemRef**) in the control data of function order components that are subsequently created.
  - The order has an *orchestration process* as the default process (it is an orchestration order).
  - The orchestration process references an orchestration sequence.
  - The orchestration sequence references the order item specification.
  - The orchestration sequence has orchestration stages.
  - The orchestration stages produce the required order components (for example, the base order components **Function**, **System**, and **ProcessingGranularity** are set up).

    You must model the orchestration stages to produce the appropriate order components; this step determines which order components participate in decomposition (orchestration plan generation).
  - The orchestration sequence is associated to an order item specification.
- The order components are associated to an orchestration fulfillment pattern.

If the preceding conditions are met, Design Studio automatically adds the following relevant data elements for control data to the **Order Template** tab of the order component specification:

- **ControlData**
- **ControlData/Functions**
- **ControlData/Functions/***Order_Component_Name*
- **ControlData/Functions/***Order_Component_Name***/orderItem**
- **ControlData/Functions/***Order_Component_Name***/componentKey**
- **ControlData/Functions/***Order_Component_Name***/calculatedStartDate**
- **ControlData/Functions/***Order_Component_Name***/duration**
- **ControlData/Functions/***Order_Component_Name***/orderItem/orderItemRef**

  orderItemRef is a reference data type that points to the node **/ControlData/OrderItem** that has been inherited from the order item specification.

Autogeneration of order component control data works as follows:

1. You select an orchestration process as the default process to which an order is submitted (to create the orchestration order).
2. Design Studio looks for the **OracleComms_OSM_CommonDataDictionary** model project in your workspace; if it does not exist, Design Studio prompts you to create it.

   If you agree to create the model project when prompted, Design Studio creates it in your workspace.

   The **OracleComms_OSM_CommonDataDictionary** model project contains the base **ControlData** structure and the data elements it references: base order item data elements and base function data elements.
3. You associate the order component with the orchestration order.
4. You add the order component to an orchestration fulfillment pattern that is part of the orchestration plan.
5. Design Studio does one of the following:
   - If the **ControlData** structure does not exist for the order, the base **ControlData** structure is automatically included from the data schema of the **OracleComms_OSM_CommonDataDictionary** model project into the Order Template of the order and order component. The required orchestration data for the new order component is added to the structure as **ControlData/Functions/** *Order_Component_Name*.
   - If the **ControlData** structure already exists for the order, Design Studio updates the existing **ControlData** structure in the order with the required orchestration data for the new order component (creating the substructure **ControlData/Functions/** *Order_Component_Name*). Design Studio adds the structure to the Order Template tab of the order component.

Modeling Order Component Control Data Automatically

You must define configuration settings and set up orchestration entities correctly to ensure that Design Studio can model the order component control data automatically by using the **OracleComms_OSM_CommonDataDictionary** model project.

> **Note:**
>
> If you are not using the **OracleComms_OSM_CommonDataDictionary** model project, do not follow these instructions. See "Modeling Order Component Control Data Manually" and model the order component control data structure manually.

To model order component control data automatically (or rather, ensure Design Studio can model it automatically):

1. Verify that the **OracleComms_OSM_CommonDataDictionary** model project is in your workspace.
2. If you do not have the model project in your workspace, see the discussion on defining orchestration preferences (for the **Common Data Dictionary** field) in "Defining OSM Preferences".
3. Verify that orchestration entities are set up for the orchestration plan in which the order component is to be included.

   See "About Autogeneration of Order Component Control Data" for more information.

4. Create the order component specification for the orchestration order.

   See "Creating New Order Component Specifications".

5. Add the order component to the orchestration fulfillment pattern.

   Design Studio creates the order component control data structure automatically by using the data schema elements of the **OracleComms_OSM_CommonDataDictionary** project as base elements for the structure.

6. Verify that Design Studio has added the order component control data structure to the following areas:

   - **Order Template** tab of the order component
   - **Order Template** tab of the order

   If you do not see the order component control data structure in the preceding areas after adding the order component to the orchestration fulfillment pattern, your OSM orchestration entities may not be set up correctly. See "About Autogeneration of Order Component Control Data" for information on the orchestration entities that must be in place for auto generation of order component control data.

7. Save all orchestration entities.

Modeling Order Component Control Data Manually

If you do not have the **OracleComms_OSM_CommonDataDictionary** model project in your workspace, you must model order component control data manually.

> **Note:**
>
> You must model a structure for order item control data in addition to modeling a structure for order component control data. For information on modeling a structure for order item control data, see "About Modeling Order Item Control Data".

To model order component control data manually:

1. Manually define the data elements for the order component control data structure in the data schema of the cartridge project you choose to include these data elements as follows:

   - Create the **ControlData/Functions** structures as root level data elements at the data schema level; do not embed them in a nested data schema data structure.

     If you are modeling data where you recompose in the project data schema the data structures you compose in the Order Template (so that they have the same structure hierarchy from root to tip), use a single **ControlData** structure; that is, the **ControlData/Functions** structure and the **ControlData/OrderItem** structure in the data schema must belong to the same **ControlData** data element.

   - Define a substructure for every order component in the orchestration fulfillment pattern; for example, **ControlData/Functions/***Order_Component_Name*. These are the order components you define in the Order Component Specification editor; for example, **ControlData/Functions/BillingFunction**, where **BillingFunction** is an order component you define.

     Use the exact spelling and case of the order component name for its defined storage area (structure) in the order template. For example, if you named an order component

BillingFunction, use **ControlData/Functions/BillingFunction**; do not use **ControlData/Functions/billingfunction**.

> **Note:**
>
> Some order components do not need to have data under **ControlData/Functions**: for example, those that are not in the orchestration fulfillment pattern.

- For the *Order_Component_Name* data element in **ControlData/Functions/***Order_Component_Name*, set **Minimum** to **0** and set **Maximum** to **unbounded** in the **Multiplicity** field.
- Define a **componentKey** node as a string below the structure to contain the component key for order components; for example, **ControlData/Functions/***Order_Component_Name***/componentKey**.

  The component key is used for calculating the processing granularity. You implement the calculation that generates a component key per order item. OSM groups order items with the same component key into order components if a decomposition rule specifies such a grouping.
- Define a reference node to **ControlData/OrderItem**, **ControlData/Functions/***Order_Component_Name***/OrderItem/orderItemRef**

  where:
  - **OrderItem** has **Minimum** set to **0** and **Maximum** set to **unbounded** in the **Multiplicity** field
  - **orderItemRef** is a reference node to **ControlData/OrderItem**

2. On the Order Component Specification editor in the **Order Template** tab, add from the data schema the data elements you defined to model the order component control data structure as follows:

```
ControlData
   Functions
      Order_Component_Name
         componentKey
         OrderItem
            orderItemRef
```

3. Save the order component.
4. On the Order editor in the **Order Template** tab, add from the data schema the substructure you defined for every order component in the orchestration fulfillment pattern; for example, **ControlData/Functions/***Order_Component_Name*. The **ControlData/Functions** structure on the order must be populated with data for each of the order components in the orchestration fulfillment pattern.

   Use the exact spelling and case of the order component name for its defined storage area (structure) in the order template. For example, if you named an order component **BillingFunction**, use **ControlData/Functions/BillingFunction**; do not use **ControlData/Functions/billingfunction**.
5. Save the order.

Creating New Order Component Specifications

You create order components to represent a system interaction with a local fulfillment system. The order component contains all of the data needed by the downstream system subprocess to complete the fulfillment task.

To create new order component specifications:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Order Component Specification**.

   The Order Component Specification wizard is displayed.

2. In the **Project Name** field, select the project in which the order component specification will be saved.

3. (Optional) In the **Extends** field, select an existing order component specification if you want the new order component specification to extend the functionality of an existing order component specification. Extending from an order component causes the child order component to inherit data elements, external fulfillment types and some other data, but does not cause the child order component to inherit data like permissions and duration, since those settings would be specific to the child order component. You will see the inherited information in the editor for the child order component specification.

   Click **Select** to open the Select Extends dialog box, where you can select an existing specification. If a suitable order component specification does not yet exist, click **New** in the Select Extends dialog box to create the order component specification. When finished, click **OK**. Your selection populates the corresponding **Extends** field in the Order Component Specification wizard.

4. In the **Name** field, enter a name for the order component specification.

   The name must be unique among order component specifications within the same namespace.

5. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the order component specification.

   Design Studio uses the last saved namespace as the default.

6. (Optional) Select a location for the order component specification.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

7. Click **Finish**.

   Design Studio adds the new order component specification to the Studio Projects view and opens the new entity in the Order Component Specification editor.

Related Topics

About Order Components

Order Component Specification Editor

Working with Order Components

Adding New Functional Order Components

Functional-level order components (fulfillment functions) contain all order items that require the same action to be performed, such as provisioning, or billing, or installation. Order items that map to the same orchestration fulfillment pattern typically require the same action. You associate fulfillment functions with the processes that run the order items.

To add a functional order component to your fulfillment topology:

1. Create a new order component.

   You create a new order component to represent the new fulfillment function. See "Creating New Order Component Specifications" for more information.

2. In the Order Component Specification editor, click the **Details** tab.

3. (Optional) In the **Extends** field, select an existing order component specification if you want the new order component specification to inherit the functionality of the existing order component specification. Extending from an order component causes the child order component to inherit data elements, external fulfillment types and some other data, but does not cause the child order component to inherit data like permissions and duration, since those settings would be specific to the child order component. You will see the inherited information in the editor for the child order component specification.

   Click **Select** to open the Select Extends dialog box, where you can select an existing order component specification. If a suitable order component specification does not yet exist, click **New** in the Select Extends dialog box to create the order component specification. When finished, click **OK**. Your selection populates the corresponding **Extends** field in the Order Component Specification wizard.

4. In the **Process** field, specify the subprocess task used to run the order component or to run order components that receive order items from this order component.

   Generally, the subprocess task that you associate with the fulfillment function will include an automation plug-in that submits the decomposed data directly to a fulfillment system or to your integration architecture. For example, if you are adding a billing fulfillment function, the subprocess task will submit the data to a billing system.

5. In the Applies to Order Component area, select the order components that will act as a source of order items to the new function.

   The order components in the Applies to Order Component area populate the corresponding order component specification with order items through decomposition rules. See "Order Component Specification Editor Details Tab" for more information.

6. Click the **Order Template** tab.

7. Configure the control data required to fulfill the order items in the fulfillment function.

   If you have the **OracleComms_OSM_CommonDataDictionary** model project in your workspace, Design Studio automatically adds the required control data structure to the Order Template area. If you do not have this model project in your workspace, you must manually configure the control data.

   See "About Modeling Order Component Control Data" and "Order Component Specification Editor Order Template Tab" for more information.

8. Click the **Duration** tab.

9. Define the processing duration for the fulfillment function.

   See "Order Component Specification Editor Duration Tab" for more information.

10. Click the **Permissions** tab.
11. Define the roles you will permit to access fulfillment function-level search queries in the Order Management web client and define the data set that their queries return.
12. Click **Save**.
13. In the Studio Projects view, double-click the orchestration fulfillment pattern to which you will associate the new fulfillment function.

    The Orchestration Fulfillment Pattern editor opens.
14. In the **Fulfillment Mode** field, select the fulfillment mode with which you want to associate the new functional component.
15. Click the **Orchestration Plan** tab.
16. In the **Order Components** subtab, click the check box adjacent to the new fulfillment function.

    The new fulfillment function is now associated with the orchestration fulfillment pattern.
17. Highlight the new fulfillment function.
18. Click the **Transitions** subtab.
19. In the **Dependencies** subtab, specify the dependencies between the new fulfillment function and any source and target order components.

    See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for more information.
20. Click **Save**.
21. Ensure that the new fulfillment function is a child of the stage that determines your fulfillment functions.

    For example, if you have a stage called DetermineFulfillmentFunctions:

    a. Open DetermineFulfillmentFunction in the Orchestration Stage editor.
    b. In the **Produces Order Component** field, select the new functional component that you are adding.
    c. Click **Save**.

    See "Working with Orchestration Stages" for more information.
22. (Optional) Create a new decomposition rule.

    If adding the functional component creates new decompositions, create a new rule. See "Creating New Decomposition Rules" for more information.
23. Add a new fulfillment system.

    Often, when you introduce a new fulfillment function, you will also require a new fulfillment system. See "Adding New Fulfillment Systems" for more information.

Related Topics

Working with Orchestration Fulfillment Patterns

Working with Decomposition Rules

Adding New Fulfillment Systems

A fulfillment system is a system that implements a fulfillment function that must be performed to fulfill order items. To add a new system to your fulfillment topology (for example, adding a new billing system to represent a geographical area), you create a new order component to

represent the system and map the order component to your existing functional order components. See "About Order Components" for more information about creating order components at the functional, system, and granularity levels.

To add a new fulfillment system:

1. Create a new order component.

   You create a new order component to represent the new fulfillment system. See "Creating New Order Component Specifications" for more information.

2. In the Order Component Specification editor, click the **Details** tab.

3. (Optional) In the **Extends** field, select an existing order component specification if you want the new fulfillment system order component to inherit the functionality of the existing order component specification. Extending from an order component causes the child order component to inherit data elements, external fulfillment types and some other data, but does not cause the child order component to inherit data like permissions and duration, since those settings would be specific to the child order component. You will see the inherited information in the editor for the child order component specification.

   Click **Select** to open the Select Extends dialog box, where you can select an existing specification. If a suitable order component specification does not yet exist, click **New** in the Select Extends dialog box to create the order component specification. When finished, click **OK**. Your selection populates the corresponding **Extends** field in the Order Component Specification wizard.

4. In the Applies to Order Component area, select the order components that will act as a source of order items to the new fulfillment system order component.

   The order components in the Applies to Order Component area populate the new fulfillment system order component with order items through decomposition rules. See "Order Component Specification Editor Details Tab" for more information.

5. Click the **Component ID** tab.

6. Define an XQuery expression, which OSM evaluates against each order item in the order component to determine the instance of the order component to which the order item will be assigned.

   See "Order Component Specification Editor Component ID Tab" for more information.

7. Click **Save**.

8. Create a new decomposition rule.

   You create a new decomposition rule to map an existing fulfillment function order component to the new fulfillment system order component.

   See "Creating New Decomposition Rules" for more information.

9. In the Decomposition Rule editor **Details** tab, identify the order item specification and fulfillment patterns to which the decomposition rule applies.

   See "Decomposition Rule Editor Details Tab" for more information.

10. Click the **Source/Target** tab.

11. Specify the order components against which OSM will evaluate the new decomposition rule and the new target order component to receive the order items that pass the decomposition rule conditions.

    See "Decomposition Rule Editor Source/Target Tab" for more information.

12. Click the **Conditions** tab.

13. Specify the conditions that OSM applies against the order items in the source order components.

    See "Decomposition Rule Editor Conditions Tab" for more information.

14. Click **Save**.

Related Topics

Working with Order Components

Working with Decomposition Rules

Order Component Specification Editor

Use the Order Component Specification editor to configure the order components that OSM uses to fulfill incoming orders.

Each order component represents a system interaction with a fulfillment system. Order components contain all of the data needed by the fulfillment system subprocess to complete the fulfillment tasks.

This editor contains several different tabs. For details about the fields on each tab, see the following topics:

- Order Component Specification Editor Details Tab
- Order Component Specification Editor Order Template Tab
- Order Component Specification Editor Duration Tab
- Order Component Specification Editor Component ID Tab
- Order Component Specification Editor Permissions Tab
- Order Component Specification Editor External Fulfillment States Tab
- Order Component Specification Editor Provider Function Tab
- Order Component Specification Editor Realization Tab
- Order Component Specification Editor System Interaction Tab

The following fields are common to multiple tabs in the Order Component Specification editor:

Field	Use
Description	Edit the display name of the order component.
Namespace	Select any existing namespace defined in the workspace or enter a unique namespace in which to include the order component. Design Studio uses the last saved namespace as the default.

Related Topics

About Order Components

Creating New Order Component Specifications

Working with Order Components

Order Component Specification Editor Details Tab

Use the **Details** tab to specify an optional base order component, the process that runs the order component, and the order components that act as a source of order items to the corresponding order component specification.

Field	Use
Extends	Select an existing order component to leverage the data and extend the functionality of that existing order component. Extending from an order component causes the child order component to inherit data elements, external fulfillment types and some other data, but does not cause the child order component to inherit data like permissions and duration, since those settings would be specific to the child order component. You will see the inherited information in the editor for the child order component specification.
Process	Specify the subprocess task used to complete the order component or to complete order components that receive order items from this order component.
Order Component Executable	Select if this order component contains all of the order items necessary to run an instance of the subprocess task associated with this order component or associated with the order component from which this order component receives order items.
Use Calculated Start Date	Select to use the calculated start date. If selected, the order component will not start until the calculated start date is reached. If the current date and time are greater than calculated start date, then the calculated start date is ignored. If disabled, the order component starts as soon as all preceding dependencies are resolved.
Applies to Order Component area	Specify the order components that act as a source of order items to the corresponding order component specification. The order components in the Applies to Order Component area populate the corresponding order component specification with order items through decomposition rules. See "Working with Decomposition Rules" for more information. • Click **New** to create a new order component specification. See "Creating New Order Component Specifications" for more information. • Click **Open** to open the corresponding specification in the Order Component Specification editor. See "Order Component Specification Editor" for more information.
Base Component **Functions** **Search**	Use these fields to limit the number of order components that appear in the Applies to Order Component area. The Order Component Specification editor displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.

Related Topics

Order Component Specification Editor

About Order Components

Creating New Order Component Specifications

Working with Order Components

Order Component Specification Editor Order Template Tab

Use the **Order Template** tab to configure the control data required to fulfill the order items on the corresponding order component.

Field	Use
Order Template	Displays the data defined for the order component. You must have the required structure for order component control data in the Order Template area; Design Studio automatically adds this structure to the Order Template area. See "About Modeling Order Item Control Data" for more information. The order in which the data is displayed here is the order in which it will appear in the Order Management web client or in the XML API if this task is an automated task. To reposition a data node in the list, select it, right-click, and select **Move Up** or **Move Down**. See "About the Order Template Context Menu" for descriptions of actions you can perform in the Order Template context menu.
Behaviors	Select a data node in the Task Data area to view the behaviors defined the node or to create new behaviors. Right-click the behavior and select **Show Properties** to define the behavior information. See "Defining Manual Task Behaviors" for more information.

Related Topics

Order Component Specification Editor

About Order Components

Creating New Order Component Specifications

Order Component Specification Editor Duration Tab

Use the **Duration** tab to define the processing durations for each order component. OSM uses either the order component specification duration or the orchestration fulfillment pattern order component duration, whichever is longer, to calculate the start date for order components. See "Orchestration Fulfillment Pattern Editor Order Components Subtab" for more information.

If **Use Calculated Start Date** is enabled, the order component will not start until the calculated start date is reached. If the current date and time are greater than calculated start date, then the calculated start date is ignored.

If **Use Calculated Start Date** is disabled, the order component starts as soon as all preceding dependencies are resolved.

The calculated start dates are only used if there is sufficient time to meet the date. Otherwise, the order component begins immediately.

See "Order Component Specification Editor Details Tab" for more information about **Use Calculated Start Date**.

OSM uses several configured durations to calculate the start times for order components. See *OSM Modeling Guide* for more information.

Field	Use
Optimistic Processing Duration area	Specify the minimum amount of time OSM can expect to process a single order item for the associated order component. Do one of the following: • Define a specific duration in the **Duration** fields. • In the **Duration Expression** field, enter an XQuery expression or a pointer to a file or URI to determine the processing duration based on a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Component Specification Duration XQuery Expressions" for more information.
Most Likely Processing Duration area	Specify the most likely amount of time OSM can expect to process a single order item for the associated order component. Do one of the following: • Define a specific duration in the **Duration** fields. • In the **Duration Expression** field, enter an XQuery expression or a pointer to a file or URI to determine the processing duration based on a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Component Specification Duration XQuery Expressions" for more information.
Pessimistic Processing Duration area	Specify the maximum amount of time OSM can expect to process a single order item for the associated order component. Do one of the following: • Define a specific duration in the **Duration** fields. • In the **Duration Expression** field, enter an XQuery expression or a pointer to a file or URI to determine the processing duration based on a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Component Specification Duration XQuery Expressions" for more information.

Related Topics

Order Component Specification Editor

Orchestration Fulfillment Pattern Editor Duration Subtab

Order Component Specification Editor Component ID Tab

Use the **Component ID** tab to define an XQuery expression that OSM evaluates against each order item in the order component. OSM stores the resulting value in the **ControlData/Functions/***FunctionName***/componentKey** data element and uses it to determine the instance of the order component to which the order item will be assigned. OSM groups all order components that share the same component ID in the same order component instance. For example, you might use the component ID to determine the processing granularity of order items at run time.

See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Component Specification Custom Component ID XQuery Expressions" for more information about this XQuery field.

Related Topics

Order Component Specification Editor

About Order Components

Creating New Order Component Specifications

Order Component Specification Editor Permissions Tab

Use the **Permissions** tab to permit specific roles access to order component search queries in the Order Management web client and define the data set that their queries return.

> **Note:**
>
> See "Order Component Specification Editor" for information about fields that appear on multiple Order Component Specification editor tabs.

Field	Use
Roles	Specify which roles can query order component information in the Order Management web client. • Click **Select** to add an existing role to the list in the **Roles** field. • Click **New** to create a new role to add to the list. See "Creating New Roles" for more information. • Click **Open** to review the role in the "Role editor." • Click **Remove** to delete the role from the list.
Query Tasks	Select the tasks that will generate the query view used by Order Management web client users. At run time, the OSM server returns a specific set of data when you use the search query functionality in the Order Management web client. You determine which data set the OSM server returns by creating or selecting query tasks. The data associated with the tasks that you select here will be the data returned to you from the run-time query. You can select tasks that you use in processes, or you can create tasks that you use only for run-time queries. For each role you add to the list in the **Roles** field, you can define two types of query task views: a summary view and a detail view. Select a task in the **Roles** field, then click any of the following: • **Add** to add an existing task to the **Query Tasks** list. • **New** to create a new task to add to the list. See "Creating New Tasks" for more information. • **Open** to review the task in the "Task editor." • **Remove** to delete the task from the list. For each task in the list, specify whether to include the task data in the summary query view or the detail query view. See "Working with Roles" for more information about setting up new roles.
Summary	Select to display the corresponding task data set in the Order Management web client **Summary** tab. The **Summary** tab provides a selection of the most important information about the selected order, component, or item and is displayed when you open the Order Details page. You can include data from multiple query tasks in the **Summary** tab. The Order Management web client displays on the **Summary** tab all of the data from all of the tasks for which you specify the **Summary** option.

Field	Use
Details	Select to use the task data set as a view in the Order Management web client **Data** tab. The tasks for which you select this option appear as choices in the Order Management web client **Data** tab **View** field. You can specify that multiple tasks appear as options in the **View** field; each option will present the web client user with a different view, each containing a specific set of data.

Order Component Specification Editor External Fulfillment States Tab

Use the **External Fulfillment States** tab to define the external fulfillment states available to the order component and order components that extend it.

Field	Use
External Fulfillment States	Displays the external fulfillment states defined for the order component. Do any of the following: • Click **Add** to create a new external fulfillment state. • Select one or more external fulfillment states and click **Remove** to remove the external fulfillment states. • Right-click a selected external fulfillment and select **Rename** to rename the selected external fulfillment state.
Details subtab	Edit the display name of the external fulfillment state. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Information subtab	Enter any additional information about the external fulfillment state that is required by your specific situation. If you are localizing OSM, use the list above the field to set different values for different locales.

Related Topics

Order Component Specification Editor

About Configuring Fulfillment States

Order Component Specification Editor Provider Function Tab

Use the **Provider Function** tab to define the provider functions to which the order component and order components that extend it are available.

Field	Use
Provider Function	Use this field to indicate the provider functions to which the order component should be available. Do any of the following: • Click **Select** to select an existing provider function to which the order component should apply. • Select a provider function and click **Remove** to remove the provider function from the list for this order component. • Click **Add** to create a new provider function. See "About Provider Functions" for more information. • Select a provider function and click **Open** to open the provider function in the Provider Function editor.

Related Topics

Order Component Specification Editor

Order Component Specification Editor Realization Tab

Use the **Realization** tab to select fulfillment functions to which the order component and order components that extend it are associated.

Field	Use
Realizes	Use this field to select the fulfillment function to which you want to associate the order component. Do any of the following: • Click **Select** to select an existing fulfillment function to which to associate the order component. • Click **Open** to open the editor for the selected fulfillment function.

Related Topics

Order Component Specification Editor

About Fulfillment Functions

Order Component Specification Editor System Interaction Tab

Use the **System Interaction** tab to import a specification (OpenAPI YAML) file that describes an external system's REST API.

Field	Use
Specification File	Use this field to select an OpenAPI YAML file that you want import or delete. In the event that a system interaction specification of the same type and version is already being used by another order component, use this field to select an existing System Interaction from your workspace. • Delete icon: Deletes the System Interaction entity defined in the Order Component Specification along with its OpenAPI and AsyncAPI specification files in that project. • **Import**: Click to import a new OpenAPI YAML file that describes the REST API of an external system.
Version	Displays the version of a specification file.
Target System	Specify the name of the external system. This is a logical name for the external system participating in this system interaction. This name must match with the target system name defined in the OSM CNTK project and instance specification files. This name 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".

Note:

When you configure system interactions, set the value of the **OSM_RUNTIME_TYPE** cartridge management variable to `MultiService`.

Related Topics

Order Component Specification Editor

Working with System Interaction Specifications

6

Working with Orchestration Processes

Orchestration processes enable Oracle Communications Order and Service Management (OSM) to fulfill incoming customer orders across participating fulfillment systems. Orchestration processes are dynamically generated at run time, based on process data that you configure in Design Studio.

OSM decomposes incoming orders defined as orchestration process orders into logical groupings of order line items, called order components. Order components contain all of the line items that are necessary to send to specific downstream fulfillment systems. Each order component represents a system interaction to a local fulfillment system.

An orchestration process is associated with one orchestration sequence. Orchestration sequences describe how OSM organizes the line items on incoming orders into the order components to be sent to external system.

When working with orchestration processes, refer to the following topics:

- Creating New Orchestration Processes
- Orchestration Process Editor

Creating New Orchestration Processes

Orchestration processes define how OSM fulfills incoming customer orders across participating fulfillment systems.

To create new orchestration processes:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Orchestration Process**.

   The Orchestration Process wizard is displayed.

2. In the **Project Name** field, select the project in which the orchestration process will be saved.

3. In the **Name** field, enter a name for the orchestration process.

   The name must be unique among orchestration process entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the orchestration process.

   Design Studio uses the last saved namespace as the default.

5. (Optional) Select a location for the orchestration process.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new Orchestration Process entity to the Studio Projects view and opens the new entity in the Orchestration Process editor.

Related Topics

Orchestration Process Editor

Working with Orchestration Processes

Orchestration Process Editor

Use the Orchestration Process editor to associate the orchestration process with an orchestration sequence.

Field	Use
Description	Edit the display name of the process.
Namespace	Select any existing namespace defined in the workspace, or enter a unique namespace in which to include the orchestration sequence. Design Studio uses the last saved namespace as the default value.
Orchestration Sequence	Configure an orchestration sequence to associate with the orchestration process. Do any of the following: • Click **Select** to select an orchestration sequence from the list of all sequences defined in the workspace. • Click **New** to create a new orchestration sequence to associate with the orchestration process. • Select a sequence and click **Open** to open the specified orchestration sequence. See "Working with Orchestration Sequences" for more information.
Invoke Order Transformation Manager	Select to invoke order transformation manager for the defined provider functions when generating the orchestration plan.
Provider Function	Define one or more provider functions that should be invoked when the orchestration plan is being generated. Do any of the following: • Click **Select** to select a provider function from the list of all of the provider functions defined in the workspace. • Select a provider function and click **Open** to open the specified provider function. • Click **Add** to create a new provider function to associate with the orchestration process. • Select a provider function and click **Remove** to remove the association between that provider function and the orchestration process. See "About Provider Functions" for more information about provider functions.

Related Topics

Creating New Orchestration Processes

Working with Orchestration Processes

7

Working with Orchestration Sequences

Orchestration sequences describe how Oracle Communications Order and Service Management (OSM) organizes the line items on incoming orders into hierarchies to be sent to external systems. Orchestration sequences include information about how to locate the line items on an incoming order and the set of orchestration stages through which the sequence transitions.

When modeling orchestration sequences, refer to the following topics:

- Creating New Orchestration Sequences
- Orchestration Sequence Editor

About Orchestration Sequences

You configure orchestration sequences for specific order item type. For each sequence, you also specify the order item location on incoming customer orders, which fulfillment modes apply to the sequence, and the stages that comprise the sequence. See "Working with Orchestration Stages" for more information.

Creating New Orchestration Sequences

Orchestration sequences define how OSM decomposes incoming customer orders.

To create new orchestration sequences:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Orchestration Sequence**.

   The Orchestration Sequence wizard is displayed.
2. In the **Project Name** field, select the project in which the orchestration sequence will be saved.
3. In the **Name** field, enter a name for the orchestration sequence.

   The name must be unique among sequence entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the orchestration sequence.

   Design Studio uses the last saved namespace as the default value.
5. (Optional) Select a location for the orchestration sequence.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

Design Studio adds the new Orchestration Sequence entity to the Studio Projects view and opens the new entity in the Orchestration Sequence editor.

Related Topics

Orchestration Sequence Editor

Orchestration Sequence Editor

Use the Orchestration Sequence editor to configure the location of the order line items on an order and to define the sequence of stages in which OSM evaluates the order line items.

Field	Use
Description	Edit the display name of the sequence.
Namespace	Select any existing namespace or enter a unique namespace in which to include the orchestration sequence. Design Studio uses the last saved namespace as the default.
Order Item	Select the order item specification to which this orchestration sequence applies. The order item includes the fixed set of properties and conditions with which the sequence will work during decomposition.

Order Item Selector	Define an XQuery expression that evaluates incoming orders and extracts all of the order items. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls.
	For example, in a simple scenario, you might define the order item selector as:
	`declare namespace ff="http://www.example.org/example";`
	`.//ff:exampleOrderItem`
	During the creation of the orchestration plan, OSM must extract all of the order items from the in-bound order to determine how to fulfill each item.
	The **Order Item Selector** specifies how to recognize order items from the in-bound message. OSM uses this to identify recurring entities in the sales order, and then applies the order item specification (that you defined in the **Order Item** field) to generate order items into the ControlData structure. See"About Order Sequence Order Item Selector XQuery Expressions" for more information.

Fulfillment Mode	Define an XQuery expression that calculates the fulfillment mode for incoming orders. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. You define an orchestration sequence for each fulfilment mode. The XQuery expression that you define returns a structure against which the incoming message must match and enables you to vary the order components that are created in the orchestration sequence. See"About Order Sequence Fulfillment Mode XQuery Expressions" for more information.

Field	Use
Orchestration Stages	Select the orchestration stages through which the order decomposition will transition. Click: • **Select** to select from a list of all stages defined in the workspace. • **New** to create a new orchestration stage to include in the orchestration sequence. See "Creating New Orchestration Stages" for more information. • **Open** to open the specified orchestration stage in the Orchestration Stage editor. See "Orchestration Stage Editor" for more information. **Note:** If you include a stage that is dependent on a parent stage, you must include the parent stage in the sequence.

Related Topics

About Orchestration Sequences

8

Working with Orchestration Stages

Oracle Communications Order and Service Management (OSM) uses a set of stages, called orchestration stages, to separate the line items on an incoming order into related groups, each group meeting some common criteria. OSM sends these groups of line items to external systems to help fulfill the order. Orchestration stages are the logical steps OSM must take to ensure that the line items on an incoming order are sent to the proper systems in a sequence that honors any dependencies among the order line items.

When modeling orchestration stages, refer to the following topics:

- Creating New Orchestration Stages
- Orchestration Stage Editor

Creating New Orchestration Stages

Orchestration stages are the logical steps OSM must take to ensure that the line items on an incoming order are sent to the proper systems in a sequence that honors any dependencies among the order line items.

To create new orchestration stages:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Orchestration Stage**.

   The Orchestration Stage wizard is displayed.
2. In the **Project Name** field, select the project in which the orchestration stage will be saved.
3. In the **Name** field, enter a name for the orchestration stage.

   The name must be unique among stage entity types within the same namespace.
4. (Optional) Select a location for the orchestration stage.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
5. Click **Finish**.

   Design Studio adds the new Orchestration Stage entity to the Studio Projects view and opens the new entity in the Orchestration Stage editor.

Related Topics

Orchestration Stage Editor

Working with Orchestration Stages

Orchestration Stage Editor

Use the Orchestration Stage editor to specify the other stages on which the corresponding stage is dependent (those stages in the orchestration sequence that OSM must evaluate first), and to specify the order components produced by the orchestration stage.

Orchestration stages are the logical steps OSM must run to ensure that the line items on an incoming order are sent to the proper systems in a sequence that honors the order line dependencies.

Field	Use
Description	Edit the display name of the stage.
Depends on Orchestration Stages	Select or create the stages on which the corresponding stage is dependent. Click **Select** to open the Select Depends on Stage dialog, which contains all of the stages defined in the workspace. Click: • **New** to open the Orchestration Stage wizard, where you can create a new orchestration stage. See "Creating New Orchestration Stages" for more information. • **Open** to open the specified stage in the Orchestration Stage editor. • **Remove** to delete the stage from the current editor.
Produces Order Component	Select the order components which OSM produces during the corresponding stage. The Orchestration Stage editor displays all order component specifications defined in the workspace. Click: • **New** to create a new order component specification to associate with the stage. See "Creating New Order Component Specifications" for more information. • **Open** to open the corresponding specification in the "Order Component Specification Editor."
Base Component **Functions** **Search**	Use these fields to limit the number of order components that appear in the **Produces Order Component** area. The Orchestration Stage editor displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.

Related Topics

Creating New Orchestration Stages

Working with Orchestration Stages

9

Working with Fulfillment Modes

A fulfillment mode denotes whether the order is intended for delivery to fulfillment systems, for testing purposes only, to request a target fulfillment system, and so forth. Every in-bound customer order that Oracle Communications Order and Service Management (OSM) receives can specify a fulfillment mode. In Design Studio, you create the fulfillment modes that OSM receives from your order capture systems. Fulfillment modes enable you to vary the order components that are created in an orchestration sequence for a single type of order.

When you define multiple fulfillment modes, OSM can receive two identical incoming orders, each with a different fulfillment mode, and each fulfillment mode defined with a different orchestration sequence. OSM generates a different orchestration plan for each incoming order (based on the fulfillment mode and the associated orchestration sequence), each containing a unique set of executable order components targeted to edge fulfillment systems, and each managed by a unique set of dependencies among the line items.

When modeling fulfillment modes, refer to the following topics:

- Creating New Fulfillment Modes
- Fulfillment Mode Editor

Related Topics

Working with Order Components

Working with Orchestration Sequences

Creating New Fulfillment Modes

You create fulfillment modes in Design Studio to represent whether an incoming order is intended for testing, for fulfillment using downstream provisioning systems, to determine whether the order is technically feasible, and so forth. OSM uses the fulfillment mode on the incoming order to determine the orchestration sequence required to fulfill the order.

To create new fulfillment modes:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Fulfillment Mode**.

   The Fulfillment Mode wizard is displayed.
2. In the **Project Name** field, select the project in which to save the fulfillment mode.
3. In the **Name** field, enter a name for the fulfillment mode.

   The name must be unique among fulfillment mode entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace, or enter a unique namespace in which to include the fulfillment mode.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the fulfillment mode.

By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

a. Click the Folder field **Browse** button.

b. Navigate to the directory in which to save the entity.

c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new fulfillment mode entity to the Studio Projects view and opens the new entity in the Fulfillment Mode editor.

Related Topics

Fulfillment Mode Editor

Working with Fulfillment Modes

Fulfillment Mode Editor

Use the Fulfillment Mode editor to edit the description and namespace of fulfillment mode. You associate fulfillment modes with orchestration sequences. See "Working with Orchestration Sequences" for more information.

Field	Use
Description	Edit the display name of the fulfillment mode.
Namespace	Select any existing namespace defined in the workspace, or enter a unique namespace in which to include the fulfillment mode. Design Studio uses the last saved namespace as the default value.

Related Topics

Creating New Fulfillment Modes

Working with Fulfillment Modes

10

Working with Milestone Events

Oracle Communications Order and Service Management (OSM) uses milestone events to send fulfillment state updates about orders, order items, and order components to upstream systems. When milestone events are configured, OSM aggregates the fulfillment states in the downstream system automatically and sends events to the upstream system.

When modeling milestone events, refer to the following topics:

About Milestone Events

Milestone events are notifications that OSM sends to upstream systems, such as a CRM system. A milestone event is an aggregation of fulfillment statuses. Milestone events are driven by fulfillment states, about which OSM receives notifications from external fulfillment systems. You can configure OSM to send milestone events to upstream systems when an order, an order item, or an order component reaches a specific fulfillment state.

You can configure milestone events for the following:

- Orders (Order Milestone Event)
- Order components (Order Component Milestone Event)
- PONR for order items (Order Item PONR Milestone Event)
- Order items (Order Item Milestone Event)

When these milestone events are configured in Design Studio, OSM triggers and sends milestone updates to upstream systems.

Creating New Milestone Events

OSM generates milestone events by using the fulfillment states and external fulfillment states of orders, orders items, and order components to update the upstream systems about the status of orders, orders items, and order components. You create milestone events by creating and configuring Milestone Event entities.

To create new milestone events:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Milestone Event**.

   The Milestone Event wizard is displayed.
2. In the **Project** field, select the project in which the milestone event will be saved.
3. In the **Name** field, enter a name for the milestone event entity.

The name must be unique among the milestone event entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the milestone event entity.

   Oracle Communications Design Studio uses the last saved namespace as the default value.

5. (Optional) Select a location for the milestone event.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new Milestone Event entity to the Studio Projects view and opens the new entity in the Milestone Event editor.

7. On the Milestone Event editor, click the **Add** button.
   Design Studio opens the Milestone Selection dialog box. This dialog box lists the milestone events that you can set up.

8. On the **Milestone Selection** dialog box, select an item for which you want to create a milestone event and click **OK**.

   The **Details** tab and the **Automation** tab for the selected item are displayed.

9. In the **Details** tab, specify the criteria for triggering milestone events for the selected event type.
   The fields and values displayed in the **Details** tab vary and are based on the selected event type.

   For information about the fields in the Details tab, refer to the corresponding **Details** tab topic for the selected milestone event type.

10. In the **Automation** tab, click **Add** to configure an automation entry and then specify XQuery sender or XSLT sender details and the routing details in the **Properties** sub-tab. For information about the fields in the **Automation** tab, refer to "Milestone Event Editor Automation Tab".

Related Topics

Milestone Event Editor

Milestone Event Editor Automation Tab

Milestone Event Editor

Use the Milestone Event editor to configure milestone events for the following:

- Order Components
- Order Items
- PONR for Order Items
- Orders

Field	Use
Description	Edit the display name of the milestone event entity.
Milestone Selection	This opens upon clicking the **Add** button. Select a milestone from the list. The Milestone Selection dialog box lists the following options for which you can configure milestones events: • OrderComponentMilestoneEvent • OrderItemMilestoneEvent • OrderItemPONRMilestoneEvent • OrderMilestoneEvent
Details tab	Specify the criteria for triggering milestone events for each milestone type. The fields displayed in this tab vary based on the selected event type. For information about the fields, refer to the corresponding **Details** tab topic for the selected milestone event type.
Automation tab	Specify details about automation for the milestone events. The fields in this tab vary based on the selected event type. For information about the fields, refer to "Milestone Event Editor Automation Tab"

Related Topics

Creating New Milestone Events

Order Milestone Event Editor Details Tab

Order Component Milestone Event Editor Details Tab

Order Item Milestone Event Editor Details Tab

Order Item PONR Milestone Event Editor Details Tab

Order Milestone Event Editor Details Tab

Use the Details tab to configure milestone events.

Field	Use
Event Name	Enter a name for the milestone event.
Order	Click the **Select** button and choose the order type for which you want to trigger a milestone event.
Applicable Fulfillment State	Select the fulfillment states, for which you want this milestone event triggered when the orders of the chosen type reach. The available states depend on the selected order type.
Properties sub-tab	This is displayed upon clicking a fulfillment state in the **Applicable Fulfillment State** list. The properties sub-tab displays a list of execution modes. Select the execution modes for which you want to trigger milestone events.

Related Topics

Milestone Event Editor Automation Tab

Creating New Milestone Events

Order Item Milestone Event Editor Details Tab

Use the Details tab to configure milestone events.

Field	Use
Event Name	Enter a name for the milestone event.
Order Item	Click the **Select** button and choose the order item type for which you want to trigger milestone events.
Fulfillment Pattern	Click the **Select** button and choose the fulfillment pattern of the selected order item for which you want to trigger milestone events.
Applicable Fulfillment State	Select the fulfillment states, for which you want this milestone event triggered when the order items of the chosen type reach.
Properties sub-tab	This is displayed upon clicking a fulfillment state in the **Applicable Fulfillment State** list. The properties sub-tab displays a list of execution modes. Select the execution modes for which you want to trigger milestone events.

Related Topics

Milestone Event Editor Automation Tab

Creating New Milestone Events

Order Component Milestone Event Editor Details Tab

Use the Details tab to configure milestone events.

Field	Use
Event Name	Enter a name for the milestone event.
Order Component	Click the **Select** button and choose the order component for which you want to trigger a milestone event.
Applicable Fulfillment State	Select the fulfillment states, for which you want this milestone event triggered when the order components of the chosen type reach. Clicking on a fulfillment state displays the Properties sub-tab. The properties sub-tab displays execution mode details. Select the execution modes for which you want to trigger milestone events.
Applicable External Fulfillment State	Select the external fulfillment states, for which you want this milestone event triggered when the order components of the chosen type reach. The available external fulfillment states depend on the chosen order component and the fulfillment states configured in the external fulfillment system. Clicking on a fulfillment state displays the Properties sub-tab. The properties sub-tab displays execution mode details. Select the execution modes for which you want to trigger milestone events.

Field	Use
Properties sub-tab	This sub-tab is displayed upon clicking a fulfillment state in the **Applicable Fulfillment State** list or in the **Applicable External Fulfillment State** list. The properties sub-tab displays a list of execution modes. Select the execution modes for which you want to trigger milestone events.

Related Topics

Milestone Event Editor Automation Tab

Creating New Milestone Events

Order Item PONR Milestone Event Editor Details Tab

Use the Details tab to configure milestone events.

Field	Use
Event Name	Enter a name for the milestone event.
Order Item	Click the **Select** button and choose the order item for which you want to trigger PONR milestone events.
Fulfillment Pattern	Click the **Select** button and choose the fulfillment pattern of the selected order item for which you want to trigger milestone events.
PONR	Select an option from the list. The available PONR options are displayed based on the chosen fulfillment pattern.

Related Topics

Milestone Event Editor Automation Tab

Creating New Milestone Events

Milestone Event Editor Automation Tab

Use the **Automation** tab to specify automation details for the configured milestone events.

Field	Use
Add Automation dialog box	This is displayed when you click the **Add** button on the **Automation** tab. In the **Add Automation** dialog box, specify a name, the automation type, and the event type. For Automation Type, you can select either XSLT sender or XQuery sender.
Properties sub-tab	This sub-tab is displayed when you click the **Properties** button on the **Automation** tab. Specify details about XQuery sender, XSLT Sender and routing details. For Routing, specify the JNDI queue to which you want to send the milestone event.
View	Use this to send additional data, such as order data, along with the event. Click the **Select** button and then choose the view that contains the data that you want to send to the queue, along with the milestone event.

Related Topics

Creating New Milestone Events

Milestone Event Editor

11

Working with Decomposition Rules

Decomposition rules specify the conditions in which Oracle Communications Order and Service Management (OSM) decomposes one or more order components into another order component. OSM evaluates every order item in the source order component against the conditions that you define for the decomposition rule. If an order item passes all specified conditions, OSM includes the order item in the target order component.

You can use decomposition rules when the orchestration fulfillment pattern associated with the order item is insufficient to determine whether additional order components are necessary. Decomposition rules are more flexible than orchestration fulfillment patterns, as they enable you to define XQuery rules to examine order item property values.

When modeling decomposition rules, refer to the following topics:

- Creating New Decomposition Rules
- Decomposition Rule Editor

Creating New Decomposition Rules

You create new decomposition rules to specify the conditions in which OSM decomposes one or more order components into other order components.

To create new decomposition rules:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Decomposition Rule**.

   The Decomposition Rule wizard is displayed.
2. In the **Project** field, select the project in which the decomposition rule will be saved.
3. In the **Name** field, enter a name for the decomposition rule.

   The name must be unique among decomposition rule entity types within the same namespace.
4. (Optional) Select a location for the decomposition rule.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
5. Click **Finish**.

   Design Studio adds the new decomposition rule to the Studio Projects view and opens the new entity in the Decomposition Rule editor.

Related Topics

Decomposition Rule Editor

Working with Decomposition Rules

Decomposition Rule Editor

Use the Decomposition Rule editor to configure decomposition rules, which specify the conditions in which OSM decomposes one or more order components into another order component.

When modeling decomposition rules, refer to the following topics:

- Decomposition Rule Editor Details Tab
- Decomposition Rule Editor Source/Target Tab
- Decomposition Rule Editor Conditions Tab

The following fields are common to multiple tabs on the Decomposition Rule editor:

Field	Use
Description	Edit the display name of the decomposition rule.
Namespace	Select any existing namespace or enter a unique namespace in which to include the decomposition rule. Design Studio uses the last saved namespace as the default.

Related Topics

Creating New Decomposition Rules

Working with Decomposition Rules

Decomposition Rule Editor Details Tab

Use the Details tab to identify the order item specification and orchestration fulfillment patterns to which the decomposition rule applies.

Field	Use
Order Item	Select the order item specification to which the decomposition rule applies. OSM evaluates every order item in the source order component against the conditions that you define for the decomposition rule. If an order item passes all specified conditions, OSM includes the order item in the target order component. Click: • **Select** to select from a list of all order items defined in the workspace. • **New** to create a new order item specification to associate with the decomposition rule. See "Creating New Order Item Specifications" for more information. • **Open** to open the specified order item in the "Order Item Specification Editor."

Field	Use
Fulfillment Pattern	(Optional) Specify one or multiple orchestration fulfillment patterns to limit the scope of the decomposition rule to those order items that match at least one of the orchestration fulfillment patterns listed. If you do not specify any orchestration fulfillment patterns, this decomposition rule applies to all order items in the source order component. Click: • **Select** to select from a list of all orchestration fulfillment patterns defined in the workspace. • **New** create a new orchestration fulfillment pattern to associate with the decomposition rule. See "Creating New Orchestration Fulfillment Patterns" for more information. • **Open** to open the specified orchestration fulfillment pattern in the "Orchestration Fulfillment Pattern Editor". • **Remove** to delete the entity from the list.

Related Topics

Decomposition Rule Editor

Creating New Decomposition Rules

Working with Decomposition Rules

Decomposition Rule Editor Source/Target Tab

Use the Source/Target tab to specify the order components against which OSM will evaluate the corresponding decomposition rule, and the target order component that receives the order items that pass the decomposition rule conditions.

Source Order Components

Field	Use
Base Components **Functions** **Search**	Use these fields to limit the number of order components that appear in the Source Order Components area. The **Source/Target** tab displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.
Source Order Components	Select the order components against which OSM evaluates the corresponding decomposition rule. OSM evaluates all order items in the specified order components. Click: • **New** to create a new source order component specification. See "Creating New Order Component Specifications" for more information. • **Open** to open the specified source order component specification in the "Order Component Specification Editor." If you specify no source order components, OSM evaluates all order items associated with the order. This scenario is valid only if the target order component is produced by an orchestration stage with no dependencies.

Target Order Components

Field	Use
Base Components **Functions** **Search**	Use these fields to limit the number of order components that appear in the Target Order Components area. The **Source/ Target** tab displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.
Target Order Components	Select the target order component that receives the order items that pass all conditions associated with the decomposition rule. Click: • **New** to create a new target order component specification. See "Creating New Order Component Specifications" for more information. • **Open** to open the specified target order component specification in the "Order Component Specification Editor."

Related Topics

Decomposition Rule Editor

Creating New Decomposition Rules

Working with Decomposition Rules

Decomposition Rule Editor Conditions Tab

Use the Conditions tab to specify the conditions that OSM applies against the order items in the source order components.

Field	Use
Decomposition Conditions	Specify the decomposition condition against which the order items in the source order components will be evaluated. OSM adds all order items that satisfy the specified conditions to the target order component. • Click **Select** to select from a list of the orchestration conditions defined for the order item specified in the **Details** tab. • Click **Add** to create a new orchestration condition to associate with the decomposition rule. • Click **Remove** to delete a condition from the list.
Condition Details	Enter an XQuery expression or a pointer to the decomposition rule. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. For example, the XQuery expression can determine a target system based on the value of an order item property. See "About Decomposition Rule Condition XQuery Expressions" for more information.

Related Topics

Decomposition Rule Editor

12

Working with Orchestration Product Specifications

Oracle Communications Order and Service Management (OSM) uses orchestration product specifications to group related commercial products that share certain attributes, and enable you to organize and reuse those attributes when you add new products to your catalog. When you assign a product to an orchestration product specification, it automatically inherits all the attributes defined in the orchestration product specification.

For example, consider that you have commercial products that customers can order called Web Gold, Web Silver, and Web Bronze. While the mailbox storage of these products may differ, they all can belong to an orchestration product specification called Internet and inherit all of the attributes that are common among Internet class products.

You do not create a new orchestration product specification in OSM, although OSM supports existing orchestration product specifications that were created in OSM. Instead, create a new conceptual model Product entity, which will then be mapped to a conceptual model fulfillment pattern, which is realized into an OSM orchestration fulfillment pattern. For more information about conceptual model products, see "About Products".

When modeling orchestration product specifications, refer to the following topics:

About Orchestration Product Specifications

You do not create a new orchestration product specification in OSM, although OSM supports existing orchestration product specifications that were created in OSM. Instead, create a new conceptual model Product entity. For more information about conceptual model products, see "About Products". The information below is generally true of both conceptual model products as used in OSM and OSM orchestration product specifications.

Some customer relationship management (CRM) systems, such as Oracle Siebel CRM, use orchestration product specifications to define a type or class of products (for example, DSL). Orchestration product specifications include dynamic attributes (characteristics) for a specific type of product. For example, DSL attributes might include *Up Speed*, *Down Speed*, *Quality of Service*, or *Service ID*.

Orchestration product specifications, of course, cannot be sold. Only products can be sold. When a marketing team decides to sell a new DSL product called *DSL Titanium*, for example, they can assign this new product to the DSL orchestration product specification, and include new attributes for this product so that a higher bandwidth or quality of service could be offered at a different price.

You map one or multiple product specifications to one fulfillment pattern. Fulfillment patterns are abstractions of product specifications. Mapping product specifications to existing fulfillment patterns enables you to introduce new products or classes of products with minimal fulfillment configuration.

Incoming customer orders contain order items that include product specification attributes as key/value pairs. For example, an order item may contain the DSL attribute Up Speed with a value of 1MB. Product specification attributes enable Design Studio to anticipate the structure of an order item and pass the attribute key/value pairs to downstream systems.

Product specifications have the ability to inherit attributes from other product specifications, and the product catalog may have hierarchies of product specifications defined. You might, for example, configure a number of attributes on a base product specification (parent product specification), then define multiple child classes that include all of the base attributes of the parent product specification, plus additional attributes.

Related Topics

Creating New Orchestration Product Specifications

Orchestration Product Specification Editor

Working with Orchestration Product Specifications

About Product Specification to Fulfillment Pattern Configuration Mapping

Configuring product specifications to fulfillment pattern mappings in Design Studio ensures that the OSM run-time server can properly fulfill order items on incoming customer orders. The information in this section applies both to existing OSM product specifications and to conceptual model products.

You can map to an existing fulfillment pattern or create and map to a new fulfillment pattern. Mapping products or product specifications to an existing fulfillment pattern is the simplest method for introducing new products or classes of products into your model, and enables you to introduce new products or classes with minimal configuration. You can also create and map to a new fulfillment pattern, or create and map to a new fulfillment pattern that inherits data from an existing fulfillment pattern. For example, you might create one base fulfillment pattern that includes attributes common among all fulfillment patterns, then extend that base specification each time you need to create a new fulfillment pattern.

Related Topics

Configuring Product-Specification-to-Fulfillment-Pattern Mapping

Working with Orchestration Product Specifications

Creating New Orchestration Product Specifications

The OSM orchestration product specification has been deprecated. To create a product in a conceptual model project, which is the replacement for the OSM orchestration product specification, see "About Products".

Related Topics

Orchestration Product Specification Editor

Working with Orchestration Product Specifications

Importing Product Specifications

The product specification import process now produces conceptual model products rather than OSM product specifications (now deprecated). For information on importing conceptual model products, see "Importing Products".

Related Topics

Orchestration Product Specification Editor

Working with Orchestration Product Specifications

Configuring Product-Specification-to-Fulfillment-Pattern Mapping

You configure the product-specification-to-fulfillment-pattern mapping to ensure that order items associated with those product specifications are fulfilled properly. When using Design Studio 8.0.0.0.0, the determination of the fulfillment pattern has been simplified as detailed in step four in the procedure given below. The target OSM server version must be 7.5.0 or later to take advantage of this simplification

To configure product-specification-to-fulfillment-pattern mapping:

1. In order to make the conceptual model content usable by OSM the following configuration needs to be completed:
   a. Create or import conceptual model products. See "About Products" for more information about conceptual model products.
   b. Create conceptual model fulfillment patterns. See "About Fulfillment Patterns" for more information about conceptual model fulfillment patterns.
   c. Map the conceptual model product to the conceptual model fulfillment pattern. See "Product Editor Properties Tab" for more information about creating this mapping.
   d. Create the OSM orchestration fulfillment pattern. See "Creating New Orchestration Fulfillment Patterns" for more information.
   e. Realize the conceptual model fulfillment pattern into the OSM orchestration fulfillment pattern. See "Realizing Conceptual Model Entities into Application Entities" for more information.
2. If you are changing an existing OSM orchestration product specification, do the following:
   a. Create the OSM orchestration fulfillment pattern. See "Creating New Orchestration Fulfillment Patterns" for more information.
   b. Open the OSM orchestration product specification and click **Select** next to the **Fulfillment Pattern** field to select the OSM orchestration fulfillment pattern.
3. In the Order Item Specification editor, specify which order item property contains the fulfillment pattern value.

   See "Order Item Specification Editor Order Item Properties Tab" for more information about configuring order item properties.
4. For the order item fulfillment pattern property, configure an XQuery expression that returns the **incoming product specification name** for the line item. When this property is populated with the product specification, then OSM can resolve the correct OSM fulfillment pattern based on the **productSpecMapping.xml** file.

> **Note:**
>
> During the build process, Design Studio generates the **productSpecMapping.xml** file in the resources/productSpecMapping folder of the OSM cartridge that contains the OSM fulfilment patterns.

See "About XQuery Expressions for Mapping Products and Fulfillment Patterns" for more information about the folder containing the mapping between the product or orchestration product specification and orchestration fulfillment pattern and sample XQuery expressions.

> **Note:**
>
> Returning the incoming product specification name instead of the fulfillment pattern name, is an important distinction when using OSM server 7.5.0 and Design Studio 8.0.0.0.0

See the "How to Simplify the Fulfillment Pattern Property on an OSM Order Item Specification (Doc ID 3000040.1)" KM article on My Oracle Support for more information on simplifying the fulfillment pattern property on an OSM order item specification.

Related Topics

Orchestration Product Specification Editor

Working with Orchestration Product Specifications

Using a Data Provider to Retrieve the Product Specification Mapping File

While configuring order item specifications, you indicate where to retrieve the product specification mapping file that determines what fulfillment pattern is applicable to each product or product specification. You define the location of the product specification mapping file in the XQuery expression of the order item property listed in the **Fulfillment Pattern Mapping Property** field in the **Property References** tab of the Order Item Specification editor.

You can use a data provider to retrieve the product or product specification mapping file by defining the data provider as a data instance in the appropriate order item specification property.

To use a data provider to retrieve the product or product specification mapping file:

1. In the Order Item Specification editor, select the property you have defined as the fulfillment pattern mapping property.
2. Select the **Instances** tab.
3. Define the data instance behavior that specifies the data provider that is to retrieve the product or product specification mapping file. See "Working with Data Providers" for more information about defining data providers.

Orchestration Product Specification Editor

Use the Orchestration Product Specification editor to edit OSM orchestration product specification configurations. You can add product specification inheritance, map the orchestration product specification to a fulfillment pattern, and describe the orchestration product specification characteristics.

Field	Use
Description	Edit the display name of the orchestration product specification.
Namespace	Select any existing namespace defined in the workspace, or enter a unique namespace in which to include the orchestration product specification. Design Studio uses the last saved namespace as the default value.
Extends	(Optional) Select an existing orchestration product specification to leverage the data and extend the functionality of that existing orchestration product specification. Do one of the following. Click: • **Select** to select from a list of all orchestration product specifications defined in the workspace. • **Open** to open the specified orchestration product specification in the Orchestration Product Specification editor.
Fulfillment Pattern	Select an orchestration fulfillment pattern to which the corresponding orchestration product specification maps. An orchestration product specification must map to a single orchestration fulfillment pattern. Do one of the following. Click: • **Select** to select from a list of all orchestration fulfillment patterns defined in the workspace. • **New** to create a new orchestration fulfillment pattern, if a suitable fulfillment pattern does not yet exist. • **Open** to open the specified fulfillment pattern in the Orchestration Fulfillment Pattern editor.
Effective Start Dates and **Effective End Dates**	Select **Immediately** if you want the orchestration product specification to be in effect immediately. Otherwise, select the date at which the orchestration product specification becomes effective. Select **No End Date** to indicate that the orchestration product specification will remain effective indefinitely. Otherwise, select the date at which the orchestration product specification will no longer be used.
Attributes	Edit, remove, or create new orchestration product specification attributes.

Related Topics

Creating New Orchestration Product Specifications

Working with Orchestration Product Specifications

13

Working with System Interaction Specifications

Oracle Communications Order and Service Management (OSM) uses System Interaction specifications to interact with external systems that utilize REST APIs.

When working with System Interaction specifications, refer to the following topics:

About System Interaction Specifications

A System Interaction specification enables OSM to interact with external systems that utilize REST APIs.

The System Interaction specifications can be TMForum REST APIs used in fulfillment (such as TMF700 for shipping or TMF 641 for provisioning), but can also be non-TMF APIs expressed as OpenAPI specifications.

A System Interaction is defined on the Order Component Specification Editor.

For conceptual information about the System Interaction specification, see *OSM Concepts* and *OSM Modeling Guide*.

Related Topics

Importing System Interaction Specifications

Importing System Interaction Specifications

To import a system interaction specification:

1. In the **Order Component Specification** editor, select the **System Interaction** tab.
2. In the **Specification File** field, click the **Import** button and then select the OpenAPI YAML file that describes the REST API of an external system.
3. In the **Target System** field, enter the name of the external system. This name must match the name of the target systems defined in the OSM CNTK project and instance specification files.

The **Version** field is populated automatically with the version number specified in the file.

Design Studio runs the following validations on the specification file you import:

- Validates that the version that is specified in `server:url` matches with the one in `info:version`. Ensure that these values match before importing the specification. The version is dictated by the external system.
- Validates the specification against the OpenAPI v3 document schema.

After successful validation, Design Studio creates the following files:

- **OpenAPI** yaml file stored in the **/openapi** folder.
- **AsyncAPI** yaml file stored in the **/asyncapi** folder.

These files are read-only and MUST not be altered under any circumstances.

A specification of a particular version can only be imported once. If you want to update an existing specification with the same version, delete the existing System Interaction entity and import it again.

Related Topics

About System Interaction Specifications

Order Component Specification Editor System Interaction Tab

Updating a System Interaction Entity

To update a System Interaction specification, use the Specification File dropdown in the Order Component Specification editor System Interaction tab, which lists all System Interaction Entities defined in the workspace.

When you select a specification file from the dropdown, the values of the associated System Interaction entity are copied to the System Interaction entity defined for the current Order Component Specification, but the specification file is not created again in the project.

To remove a System Interaction defined in an Order Component Specification, use the delete button. Selecting the **--select–** placeholder or clicking the Delete button deletes the System Interaction entity defined in that Order Component Specification along with its OpenApi and AsyncApi specification files in that project. If the deleted System Interaction entity refers to the OpenAPI and AsyncApi specification files from another project, then the specification files are not deleted.

For conceptual information about the System Interaction specification, see *OSM Concepts* and *OSM Modeling Guide*.

Related Topics

Importing System Interaction Specifications

Order Component Specification Editor System Interaction Tab

Using System Interactions in Automation

System Interactions are defined on OSM Order Components. Automation tasks can reference the operations and events defined within the System Interaction specification. To do so, the automation task must have a relationship to the specification file through the hierarchy of task-process-order component. The specification on an order component can be accessed from any automation task that is part of the process defined on the same Order Component.

The automated task should be associated with only one process and that process should be associated with only one order component.

To use System Interactions in automated tasks, in the Automated Task Details tab, select **System Interaction** in the **Transport** dropdown.

To update Automation properties for an automated task, use the System Interaction tab in the Properties view of the Automation tab.

See the topics about the following in the Modeling OSM Processes Online Help for details about the fields and tabs to use for using system interactions in automation:

- **Task Editor Details Tab** in Working with Tasks
- **Task Editor Automation Tab** in Working with Tasks

Related Topics

About System Interaction Specifications

Importing System Interaction Specifications

Order Component Specification Editor System Interaction Tab

14

Working with Hosted Specifications

Oracle Communications Order and Service Management (OSM) uses Hosted Specifications to import TMF OpenAPI specifications.

When working with hosted specifications, refer to the following topics:

- About Hosted Specifications
- Importing Hosted Specifications
- Hosted Specification Editor

About Hosted Specifications

A Hosted specification is one where this deployment of OSM serves the specification, persists and manages the key objects in the specification, and offers the operations described in the specification. OSM uses hosted specifications to host TM Forum (TMF) specifications for managing TMF-based product and service orders in OSM. Using hosted specifications, you import TMF specifications to support ordering using TMF Open APIs. OSM supports all operations described in the TMF **product ordering** and **service order** specifications.

You can use the TMF specification as-is or add more data to the specification (OSM extended specification) based on your business requirements.

For conceptual information about the Hosted specification and the OSM extended specification, see *OSM Concepts* and *OSM Modeling Guide*.

Related Topics

Importing Hosted Specifications

Hosted Specification Editor

Working with Hosted Specifications

Importing Hosted Specifications

To import a hosted specification:

1. From the **Studio Projects View** context menu, select **Import** and then select **Import Hosted Specification**.
2. In the **Specification Description** field, enter a description of the hosted specification.
3. In the **Specification file** field, click the **Browse** button and then select the OpenAPI YAML file that you want to import.
4. In the **Event Target System** field, enter a logical name for the external system that will receive northbound TMF event notifications. The name of the target system must match any of the target system definitions in the OSM CNTK specification files.
5. In the **Cartridge Name** field, enter a name to be used by Design Studio to generate TMF cartridges.

6. In the **Cartridge Version** field, enter the version to be used by Design Studio when generating TMF cartridges.
7. Click **Finish**.

Design Studio runs the following validations on the specification file you import:

- Validates the API Name in servers.url. The supported APIs are **productOrderingManagement** and **serviceOrdering**.
- When importing an OSM extended specification, it must be of the OSM release that aligns with the Design Studio version.
- Validates the specification against the OpenAPI 3.0 document schema.
- Validates whether the version of servers.url is 4 or greater, which is the minimum version supported by OSM.

After successful validation, Design Studio creates the following projects in the **Studio Projects** view:

- An OSM cartridge project: Uses the name entered in the **Cartridge Name** field in the import wizard. In the OSM project, you can see the hosted specification entity, which holds the details of hosted specification. This project contains the hosted specification, fulfillment mode, order life-cycle policy and role entities.
- A data model project: Uses the entered cartridge name appended with **_CDT**. This project contains the data schemas and data structure definitions associated with the imported specification.

Related Topics

About Hosted Specifications

Hosted Specification Editor

Working with Hosted Specifications

Hosted Specification Editor

Use the Hosted Specification editor to edit hosted specifications. You can update the event target system and the customer name JSON path for a selected hosted specification.

Field	Use
Specification Name	Displays the specification name extracted from the imported specification.
Specification Version	Displays the specification version extracted from the imported specification.
Specification Namespace	Displays the specification namespace extracted from the imported specification.
Specification File	Displays the specification file name extracted from the imported specification.
Event Target System	A logical name for the external system that receives the upstream TMF event notifications generated by OSM.

Field	Use
Customer Name JSON Path	A JSON path where $. represents the root TMF resource (productOrder or serviceOrder). The remaining path can point to any data element contained within the root resource schema. At runtime, the value on the incoming order payload (at that path) is used as the customer name in the OSM Fallout user interface.

Related Topics

Importing Hosted Specifications

About Hosted Specifications

Working with Hosted Specifications

15

Working with Orchestration Fulfillment Patterns

Oracle Communications Order and Service Management (OSM) uses orchestration fulfillment patterns to describe the structure of a set of products. Many conceptual model products or OSM orchestration product specifications can map to the same orchestration fulfillment pattern, which enables you to introduce new products without requiring new fulfillment patterns in Design Studio. Additionally, you can create base specifications that you reuse and extend to reduce the amount of modeling necessary to represent the types of products supported in your product catalog.

OSM must determine the orchestration fulfillment pattern for every order item on an incoming order. OSM acquires the orchestration fulfillment pattern from the property on the order item that is specified in the **Fulfillment Pattern Mapping** field in the **Property Mapping** tab in the Order Item Specification editor. OSM uses the orchestration fulfillment pattern to determine to which order components to add an order item and what dependencies exist between the order components.

OSM supports importing fulfillment patterns from the Oracle Siebel CRM product catalog through the use of an Enterprise Business Service (EBS). The EBS supports a query interface that enables modelers to specify a Siebel product class ID and retrieve its definition. During modeling, in Design Studio, you map the product or orchestration product specification definition to a fulfillment pattern. See "Configuring Product-Specification-to-Fulfillment-Pattern Mapping" for more information.

When working with orchestration fulfillment patterns, refer to the following topics:

- About Orchestration Fulfillment Pattern Dependencies
- Viewing Orchestration Fulfillment Pattern Dependencies
- Creating New Orchestration Fulfillment Patterns
- About Points of No Return
- Orchestration Fulfillment Pattern Editor

About Orchestration Fulfillment Pattern Dependencies

Orchestration fulfillment patterns can have dependencies across functions and products or orchestration product specifications.

Dependencies Across Functions

Dependencies can exist between different functions for an orchestration fulfillment pattern and can be based on completion. For example, the Billing function can be started only after the Provisioning function completes, and the SyncCustomer function, Collections function, and Marketing function can be started only after the Billing function completes. Orchestration fulfillment patterns can have dependencies between order components within the corresponding fulfillment pattern or between order components on other fulfillment patterns. For example, the fulfillment pattern determines to which order component an order item must be added.

Dependencies Across Products or Orchestration Product Specifications

Fulfillment patterns can have dependencies across many conceptual model products or OSM orchestration product specifications. For example, different sales products can be added to the same conceptual model product or OSM orchestration product specification, which can map to the same fulfillment pattern.

Viewing Orchestration Fulfillment Pattern Dependencies

You can view the dependencies for orchestration fulfillment patterns using the **Fulfillment Pattern Relation Graph Dependencies** tab.

The **Fulfillment Pattern Relation Graph Dependencies** tab shows dependencies that exist between fulfillment functions within an orchestration fulfillment pattern and between orchestration fulfillment patterns. It shows a design-time perspective of the dependencies in the same way as the Dependency Graph region in the **Orchestration Plan** tab in the Order Management web client shows a run-time representation of the dependencies. This tab shows a graphical representation of the dependencies between functions at design time. You can view the dependencies between functions at design time and compare that against dependencies between functions at run time. The **Fulfillment Pattern Relation Graph Dependencies** tab shows a flow-based view; a relationship-based view is shown by the **Fulfillment Pattern Relation Graph General** tab.

To view the orchestration fulfillment pattern dependencies:

1. In Design Studio, open the Orchestration Fulfillment Pattern editor for the fulfillment pattern you are interested in.
2. Select the **Orchestration Plan** tab.
3. Open the Relation Graph view unless it is already open.
4. Select the **Dependencies** tab in the Relation Graph view.

Related Topics

Fulfillment Pattern Relation Graph Dependencies Tab

Fulfillment Pattern Relation Graph Dependencies Tab

Use the **Fulfillment Pattern Relation Graph Dependencies** tab to copy an image of the graph into a document or an image file, show or hide outbound references, select a fulfillment mode, and view additional orchestration fulfillment patterns.

By default, the **Fulfillment Pattern Relation Graph Dependencies** tab shows in-bound dependencies (those defined directly in the editor). Outbound dependencies (those referenced by another orchestration fulfillment pattern) can be shown by selecting the **Outbound** option.

For example, consider an orchestration fulfillment pattern PS1 that defines the following dependencies between functions A, B, and C:

- Function A has to complete before function B can start
- Function B has to complete before function C can start

Consider an orchestration fulfillment pattern PS2 that defines the following dependencies between functions D, E, B, and F:

- Function D has to complete before function E can start

- Function B (which is defined on PS1) has to complete before function F can start

The relation graph for PS1 shows the dependencies between functions A, B, and C. Selecting the **Outbound** option in the **Fulfillment Pattern Relation Graph Dependencies** tab shows the dependencies between functions D, E, B, and F for PS2 (because PS1 has an outbound reference to PS2). You will see an additional connection between function B on PS1 and function F on PS2.

The **Fulfillment Pattern Relation Graph Dependencies** tab has the following options on the context menu:

Field	Use
Copy	Copies the view region to the clipboard and allows you to paste an image of the graph into a document or an image file.
Outbound	Displays or hides outbound references.
Fulfillment Mode	Lets you select a fulfillment mode. Changing fulfillment modes updates the diagram.
Fulfillment Pattern	Lets you view only those orchestration fulfillment patterns and dependencies in which you are interested. This field is accessible when **Outbound** is selected.
Zoom To	Increases the size of the objects to a predefined percentage value.

Creating New Orchestration Fulfillment Patterns

When you are required to represent new sets of products, you create new orchestration fulfillment patterns in Design Studio to model the fulfillment modes, order components, and dependencies required to fulfill order items associated with the corresponding orchestration fulfillment patterns.

To create new orchestration fulfillment patterns:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Orchestration Fulfillment Pattern**.

   The Fulfillment Pattern wizard is displayed.

2. In the **Project** field, select the project in which to save the orchestration fulfillment pattern.

3. (Optional) In the **Extends** field, select an existing orchestration fulfillment pattern if you want the new orchestration fulfillment pattern to extend the functionality of an existing orchestration fulfillment pattern.

   Click **Select** to open the Select Extends dialog box, where you can select an existing orchestration fulfillment pattern. If a suitable orchestration fulfillment pattern does not yet exist, click **New** in the Select Extends dialog box to create the orchestration fulfillment pattern. When finished, click **OK**. Your selection populates the corresponding **Extends** field in the Fulfillment Pattern wizard.

4. In the **Name** field, enter a name for the orchestration fulfillment pattern.

   The name must be unique among orchestration fulfillment pattern entity types within the same namespace.

5. In the **Namespace** field, select an existing namespace, or enter a unique namespace in which to include the orchestration fulfillment pattern.

   Design Studio uses the last saved namespace as the default.

6. (Optional) Select a location for the orchestration fulfillment pattern.

By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

a. Click the **Folder** field **Browse** button.

b. Navigate to the directory in which to save the entity.

c. Click **OK**.

7. Click **Finish**.

   Design Studio adds the new orchestration fulfillment pattern to the Studio Projects view and opens the new entity in the Orchestration Fulfillment Pattern editor.

Related Topics

Orchestration Fulfillment Pattern Editor

Working with Orchestration Fulfillment Patterns

About Points of No Return

There are two types of points of no return in OSM. Hard points of no return specify common fulfillment states at which processing of order amendments that affect the associated order component are either impossible or undesirable. Soft points of no return specify common fulfillment states at which order amendment processing is still feasible but involves some penalty.

When you create a point of no return, you create the following:

- Fulfillment states. These are required before configuring points of no return. See "About Configuring Fulfillment States" for more information.
- A point of no return value list on the Orchestration Fulfillment Pattern editor **Details** tab. Create a name for your point of no return and indicate whether it is a hard point of no return or not. See "Orchestration Fulfillment Pattern Editor Details Tab" for more information.
- Point of no return rules, in the Orchestration Fulfillment Pattern editor **Orchestration Plan** tab, for the point of no return values you created. Point of no return rules are specified at the order component level. Point of no return rules involve selecting one or more common fulfillment states to map to the specified point of no return value. See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for more information.
- One or more transition conditions in the Order Lifecycle Policy to check the point of no return value. See "Transition Condition for Checking a Hard Point of No Return" for information about defining this condition.

Orchestration Fulfillment Pattern Editor

Use the Orchestration Fulfillment Pattern editor to describe the structure of a group of related products in a product catalog and to configure the logical sequence of steps that must be performed to fulfill order items associated with the orchestration fulfillment pattern.

The following fields are common to all of the Orchestration Fulfillment Pattern editor tabs:

Field	Use
Description	Edit the display name of the orchestration fulfillment pattern.

Field	Use
Namespace	Select an existing namespace or enter a unique namespace in which to include the orchestration fulfillment pattern. Design Studio uses the last saved namespace as the default.

When configuring orchestration fulfillment patterns in the Orchestration Fulfillment Pattern editor, refer to the following topics:

- Orchestration Fulfillment Pattern Editor Details Tab
- Orchestration Fulfillment Pattern Editor Orchestration Plan Tab
- Orchestration Fulfillment Pattern Editor Realization Tab

Orchestration Fulfillment Pattern Editor Details Tab

Use the **Details** tab to select an existing orchestration fulfillment pattern on which to base the corresponding orchestration fulfillment pattern and to identify with which fulfillment modes the orchestration fulfillment pattern is associated.

> **Note:**
>
> See "Orchestration Fulfillment Pattern Editor" for information about fields that appear on all Orchestration Fulfillment Pattern editor tabs.

Field	Use
Extends	Select an existing specification if you want the new orchestration fulfillment pattern to inherit the functionality of an existing orchestration fulfillment pattern. Inheritance by extension dictates that the contents of this fulfillment pattern's orchestration plan is augmented with the sum contents of all ancestor fulfillment pattern orchestration plans. The child fulfillment pattern includes all order components and transitions associated with parent fulfillment patterns. Do any of the following: • Click **Select** to select from a list of all orchestration fulfillment patterns defined in the workspace. • Click **New** to create a new orchestration fulfillment pattern. • Click **Open** to open the specified orchestration fulfillment pattern in the Orchestration Fulfillment Pattern editor.
Default Fulfillment State Map	Select or create the fulfillment state map that should be used as a default when creating a new fulfillment state mapping from the order component list on the **Orchestration** tab. See "Orchestration Fulfillment Pattern Editor Order Components Subtab" for more information.

Field	Use
Fulfillment Modes	Associate the orchestration fulfillment pattern with one or more fulfillment modes to represent the types of orders on which this product will appear. You configure the order components, transitions, and process durations for each fulfillment mode on the Orchestration Fulfillment Pattern editor **Orchestration Plan** tab. Do any of the following: • Click **Select** to select from a list of all fulfillment modes defined in the workspace. • Click **New** to create a new fulfillment mode. • Click **Open** to open the specified fulfillment mode in the Fulfillment Mode editor. • Click **Remove** to delete a fulfillment mode from the list. See "Working with Fulfillment Modes" for more information.
Point of No Return values list	Displays the point of no return values created for this orchestration fulfillment pattern. It also displays point of no return values inherited from other fulfillment patterns in the hierarchy established by the **Extends** field. Inherited points of no return indicate the entity from which they are inherited and can only be edited in the source entity. Do any of the following: • Click **Add** to add an entry. You cannot create a new entry by selecting a blank row in the table. • Click **Remove** to remove the selected entry. • Select an entry in the list to edit it using the fields in the subtabs on the right. Use the **Details** subtab to select whether it is a hard point of no return and enter a description in the **Description** field. If **Hard Point of No Return** is not selected, it indicates a soft point of no return. If you are localizing OSM, use the list to the right of the **Description** field to set different values for different locales. Use the **Information** subtab to store any additional information about the point of no return that is required by your specific situation. If you are localizing OSM, use the list above the field to set different values for different locales.

Related Topics

Orchestration Fulfillment Pattern Editor

Orchestration Fulfillment Pattern Editor Orchestration Plan Tab

Orchestration Fulfillment Pattern Editor Orchestration Plan Tab

Use the **Orchestration Plan** tab to configure the order components, dependencies, and process durations for each fulfillment mode associated with the orchestration fulfillment pattern.

The following fields are common to multiple subtabs on the Orchestration Fulfillment Pattern editor **Orchestration Plan** tab:

> **Note:**
>
> See "Orchestration Fulfillment Pattern Editor" for information about fields that appear on all Orchestration Fulfillment Pattern editor tabs.

Field	Use
Fulfillment Mode	A fulfillment mode denotes whether the order is intended for delivery to fulfillment systems, for testing purposes only, to request a target fulfillment system, and so forth. An orchestration fulfillment pattern can be associated with multiple fulfillment modes, each configured differently. Click the **Fulfillment Mode** menu to access additional fulfillment modes associated with the orchestration fulfillment pattern, and configure the order components, transitions, and process durations for every fulfillment mode listed in the menu. See "Working with Fulfillment Modes" for more information.

When modeling orchestration fulfillment patterns, refer to the following topics for additional information:

- Orchestration Fulfillment Pattern Editor Order Components Subtab
- Orchestration Fulfillment Pattern Editor Dependencies Subtab
- Orchestration Fulfillment Pattern Editor Duration Subtab

Orchestration Fulfillment Pattern Editor Order Components Subtab

Use the **Order Components** subtab to select all of the order components to which an order item associated with the corresponding orchestration fulfillment pattern can be added when OSM creates an orchestration plan, to specify the time OSM can expect to process a single order item for the associated order component, to include any order items that do not share the same fulfillment pattern, and to add order items to an order component only when the condition evaluates to **true**.

Order Components

Use the Order Components area to select all of the order components to which an order item associated with the corresponding fulfillment pattern can be added when OSM creates an orchestration plan.

Field	Use
Base Component **Functions** **Search**	Use these fields to limit the number of order components that appear in the Order Components area. The **Order Components** subtab displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.

Field	Use
Order components list	Displays the order components that meet the criteria selected in the **Base Component**, **Functions**, and **Search** fields. A check preceding the order component indicates that it is an order component processed by the orchestration fulfillment pattern. You can right-click in this box and select any of the following functions: • **Create New Order Component:** Select this to create a new order component as a child of the selected order component. See "Creating New Order Component Specifications" for more information. • **Create New Root Order Component**: Select this to create a new order component that is not a child of another order component. See "Creating New Order Component Specifications" for more information. • **Create Fulfillment State Mapping in Default Map**: Select this to create a new fulfillment state mapping in the fulfillment state map entered in the **Default Fulfillment State Map** field on the **Details** tab. See "Creating New Fulfillment State Maps" for more information. • **Create Fulfillment State Mapping**: Select this to create a new fulfillment state mapping in a fulfillment state map of your choice. When you select it, an dialog box is displayed that allows you to select a fulfillment state map to contain the new mapping. See "Creating New Fulfillment State Maps" for more information. When you select one of the options to create a new fulfillment state mapping, the orchestration fulfillment pattern and fulfillment mode criteria are entered by default in the new mapping. If all of the following conditions are met, the new mapping will also contain the order item, orchestration sequence, orchestration stage, and order component criteria. • The selected order component is produced by exactly one orchestration stage. • The orchestration stage is produced by a single orchestration sequence. • The orchestration sequence specifies an order item in its **Order Item** field. You can make any changes you like to the default data values provided in the mapping.
Open **New**	Select any order component in the list and click **New** to create and add a new order component to the list. Click **Open** to open the order component in the Order Component Specification editor.

Selected Order Components

For each order component you select in the Order Components area, you can also define its processing duration, whether the order items included in order components are associated by additional properties (such as component IDs), and the conditions that dictate whether the order items are added to the order component.

OSM uses either the orchestration fulfillment pattern selected order component duration or the order component specification duration, whichever is longer, to determine the start times for order components. See "Order Component Specification Editor Duration Tab" for more information.

If **Use Calculated Start Date** is enabled, the order component will not start until the calculated start date is reached. If the current date and time are greater than calculated start date, then the calculated start date is ignored.

If **Use Calculated Start Date** is disabled, the order component starts as soon as all preceding dependencies are resolved.

The calculated start dates are only used if there is sufficient time to meet the date. Otherwise, the order component begins immediately.

See "Order Component Specification Editor Details Tab" for more information about **Use Calculated Start Date**.

OSM uses several configured durations to calculate the start times for order components. See *OSM Modeling Guide* for more information.

Field	Use
Duration subtab	Specify the time OSM can expect to process a single order item for the associated order component for this orchestration fulfillment pattern. The **Optimistic Processing Duration** area contains information about the minimum expected duration. The **Most Likely Processing Duration** area contains information about the most likely expected duration. The **Pessimistic Processing Duration** area contains information about the maximum expected duration. For each of the areas, do one of the following: • Define a specific duration in the **Duration** fields. • In the **Duration Expression** field, enter an XQuery expression or a pointer to a file or URI to determine the processing duration based on a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Orchestration Fulfillment Pattern Component Duration XQuery Expressions" for more information.

Field	Use
Order Item Associations subtab	Include order items that do not share the same orchestration fulfillment pattern in the order component. Select an order component in the Order Components area and do any of the following: • Click **Add** to enter a name for a new order item association. • Click **Select** to select a previously created order item association to associate with the corresponding order component. • Click **Remove** to delete an order item association from the list. You can select the following options in the **Order Item Associations** subtab: • Fulfillment Pattern • Matching Order Component ID • Property Correlation By default, **Fulfillment Pattern** is selected. You can also group order items into an order component when those order items share a common component ID or property correlation. If you select **Property Correlation**, enter an XQuery expression to select the order item property. See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" and "About Associating Order Items Using Property Correlations XQuery Expressions" for more information.
Conditions subtab	Provide a condition so that OSM adds order items to an order component only when the condition evaluates to **true**. Select an order component in the Order Components area and click **Add** to open the Add Condition dialog box, where you can specify a name for a new condition. Click **Select** to open the Select a Condition dialog box, where you can select a previously created condition to associate with the corresponding order component. Select any condition and click **Remove** to delete it from the list. Select any condition from the **Conditions** list and enter the XQuery expression to run against the order item. See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" and "About Order Item Specification Condition XQuery Expressions" for more information.
Point of No Return subtab	Associate a point of no return with one or more common fulfillment states for the currently selected order component. Select a point of no return from the list, and then select each of the fulfillment states to which that point of no return should apply. Note that a point of no return must be associated with each fulfillment state that can cause it. Selecting a parent fulfillment state indicates that a point of no return will also be set for the children of that fulfillment state. The list of points of no return is populated with the values listed in the Orchestration Fulfillment Pattern editor **Details** tab.

Related Topics

Orchestration Fulfillment Pattern Editor Dependencies Subtab

Order Component Specification Editor Duration Tab

Orchestration Fulfillment Pattern Editor Dependencies Subtab

Use the **Dependencies** subtab to configure dependencies (the logical sequence between order components) within the corresponding orchestration fulfillment pattern or between order components on other orchestration fulfillment patterns and order components on the corresponding orchestration fulfillment pattern.

Field	Use
Dependencies area	Lists the dependencies that have been defined for the orchestration fulfillment pattern for the selected **Fulfillment Mode**. Do any of the following: • Click **New** to add a new row to the table. Once you have created the new row, configure it by selecting the row and entering values in the tabs in the **Selected Dependency** area. • Click **Remove** to remove a row from the table.
Inherited From	Denotes the source of the dependency when this orchestration fulfillment pattern inherits from a parent fulfillment pattern. The parent fulfillment pattern is defined in the **Extends** field in the Fulfillment Pattern editor **Details** tab.
From Fulfillment Pattern **From Component** **To Component**	Configure these values on the **From/To Components** Subtab in the **Selected Dependency** area.

From/To Components Subtab

Use the **From/To Components** subtab in the **Selected Dependency** area to specify the order components to which the dependency applies.

> **Note:**
>
> You must select a row in the **Dependencies** table before you can define the fields on this subtab.

Field	Use
From Fulfillment Pattern	Click **Select** or **New** to configure a dependency between order components on a different orchestration fulfillment pattern with order components on the corresponding orchestration fulfillment pattern. Click **Open** to review the fulfillment pattern in the Orchestration Fulfillment Pattern editor.
From Order Component	Specify the source order component. Select an order component and click **Open** to review the order component in the Order Component Specification editor. Click **New** to create a new order component to add to the **From Order Component** list.

Field	Use
To Order Component	Specify the target order component that starts after all conditions associated with the dependent order items for the source order component have been met. Select an order component and click **Open** to review the order component in the Order Component Specification editor. Click **New** to create a new order component to add to the **To Order Component** list.

Order Item Dependency Subtab

Use the **Order Item Dependency** subtab in the **Selected Dependency** area to specify which order items in the source and target order components have dependencies.

> **Note:**
>
> You must select a row in the **Dependencies** table before you can define the fields on this tab.

By default, an order item in the **To Component** field in the **Dependencies** table has a dependency when the same order item exists in the **From Component** field. However, you can use order item dependencies if you want to create a dependency on two different order items. For example, order items in a shipping order component may not be identical to order items in an install order component, but a dependency between those order items exists.

Field	Use
Order Item	This is the default option.
Fulfillment Pattern	Select this option to set a dependency based on orchestration fulfillment pattern.
Property Correlation	Select this option to set a dependency based on order item property correlation. If you select this option, enter an XQuery expression to select the order item property that order items in the **To Order Component** field must share with order items in the **From Order Component** field. **Note**: See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" and "About Order Item Dependency Property Correlation XQuery Expressions" for more information.

Wait Condition Subtab

Use the **Wait Condition** subtab in the **Selected Dependency** area to specify the conditions that must be satisfied before the transition can occur.

> **Note:**
>
> You must select a row in the **Dependencies** table before you can define the fields on this subtab.

Field	Use
Task State Completed	Select this option to specify that the transition should wait until the subprocess task associated with the order component reaches the completed state.
Data Change Notification	Select this option to specify that a notification is to be generated when a data value changes if that data value change passes the condition expression. This field indicates that the dependency target order component cannot process until the value you select for the data change notification property of the order item changes. For example, you can configure a data change notification on a status field in the order template so that if the status changes to Failed, OSM generates a fallout notification to an administrator. When you select **Data Change Notification**, you also define the following fields: • In the **Order Item** field, select the order item specification upon which the dependency target order component must wait. • In the **Data Change Notification Property** field, select the order item property that contains the data change notification property value upon which the dependency target order component must wait. See "Order Item Specification Editor Order Item Properties Tab" for more information about defining order item properties. • In the **Relative Path** field, you can specify the location of the data change notification property. • In the Data Change Condition Expression area, you can configure an XQuery expression to denote that the transition wait until the property contains a specific value. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Order Data Change Wait Condition XQuery Expressions" for more information about this XQuery field.
Wait Delay area	Use the Wait Delay area to configure a delay that supplements the information defined in the **Wait For Condition** area. You configure the Wait Delay against either the **Task State** or the **Data Change Notification** wait condition options. After the conditions defined in the Wait For Condition area are met, OSM further delays the processing of the corresponding target order component using the information defined here. The wait delay setting defers the start of the waiting order component after the dependencies have been resolved. For example, you can use this to defer the resolution of the condition until two days after a task completion. Do one of the following: • Select **Duration** and use the **Duration** fields to specify the amount of time to wait after the conditions defined in the Wait For Condition area is met. • Select **Date Time Expression** and use the **Duration Expression** field to use XQuery to set the date and time for the transition to occur.
Duration fields	Use these fields to set a static amount of time to delay. Only available if the **Duration** option is selected.

Field	Use
Duration Expression	Create an XQuery expression in the **Duration Expression** field to run against a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. If the **Duration** option is selected, write the XQuery expression to return a string in the xs:duration format (for example,"PT15M"). See "About Wait Delay Duration XQuery Expressions" for more information. If the **Date Time Expression** option is selected, write the XQuery expression to return a string in the xs:dateTime format (for example, "2012-09-15T15:30:00-4:00"). See "About Wait Delay Date and Time XQuery Expressions" for more information.

Orchestration Fulfillment Pattern Editor Duration Subtab

For cartridges that target OSM server version 7.2.0 or earlier, use the **Duration** subtab to specify the minimum process duration for the associated orchestration fulfillment pattern.

> **Note:**
>
> OSM uses several configured durations to calculate the start times for order components. See *OSM Modeling Guide* for more information.

For cartridges that target OSM server version 7.2.2 or later, the **Duration** subtab has no effect on orchestration fulfillment patterns. If you configure any values and deploy the cartridge to OSM server 7.2.2 or later, you will get a warning that the configured values on the **Duration** subtab are ignored.

Field	Use
Duration	Define a specific duration in the **Duration** field, or create an XQuery expression in the **Duration Expression** field to run against a specific order item property. This is used when the calculated duration for an order item based on order component durations and dependencies is less than the minimum processing duration value for the orchestration fulfillment pattern that the order item maps to. You can override the orchestration fulfillment pattern process duration by specifying the processing duration for individual order components.
Duration Expression	Enter an XQuery expression or a pointer to determine the processing duration based on a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Orchestration Fulfillment Pattern Duration XQuery Expressions" for more information.

Related Topics

Orchestration Fulfillment Pattern Editor Order Components Subtab

Order Component Specification Editor Duration Tab

Orchestration Fulfillment Pattern Editor Realization Tab

Use the **Realization** tab to select the conceptual model fulfillment pattern that is realized by this OSM orchestration fulfillment pattern.

> **Note:**
>
> See "Orchestration Fulfillment Pattern Editor" for information about fields that appear on all Orchestration Fulfillment Pattern editor tabs.

Field	Use
Realization	Specifies the conceptual model fulfillment pattern that is realized by this orchestration fulfillment pattern. See "About Fulfillment Patterns" for more information about conceptual model fulfillment patterns. Do any of the following: • Click **Select** to select a conceptual model fulfillment pattern to be realized by this orchestration fulfillment pattern. • Click **Open** to open the editor for the selected conceptual model fulfillment pattern.

Related Topics

Orchestration Fulfillment Pattern Editor

16

Working with Orchestration Dependencies

Orchestration dependencies are dependencies that exist between order items within a set of order components that are not based on fulfillment patterns. When Oracle Communications Order and Service Management (OSM) receives a customer order from an external system, it determines all dependencies that exist among the order items on the order.

When modeling orchestration dependencies, refer to the following topics:

- Creating New Orchestration Dependencies
- Orchestration Dependency Editor

Creating New Orchestration Dependencies

To create new orchestration dependencies:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Orchestration Dependency**.

   The Orchestration Dependency wizard is displayed.
2. In the **Project** field, select the project in which to save the orchestration dependency.
3. In the **Name** field, enter a name for the orchestration dependency.

   The name must be unique among orchestration dependency entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace, or enter a unique namespace in which to include the orchestration dependency.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the orchestration dependency.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

   Design Studio adds the new orchestration dependency to the Studio Projects view and opens the new entity in the Orchestration Dependency editor.

Related Topics

Orchestration Dependency Editor

Working with Orchestration Dependencies

Orchestration Dependency Editor

Use the Orchestration Dependency editor to configure dependencies that exist between order items within a set of order components that are not based on fulfillment patterns.

When modeling orchestration dependencies, refer to the following topics:

- Orchestration Dependency Editor Details Tab
- Orchestration Dependency Editor Wait for Condition Tab
- Orchestration Dependency Editor Order Item Dependencies Tab

The following fields are common to multiple tabs in the Orchestration Dependency editor:

Field	Use
Description	Edit the display name of the orchestration dependency.
Namespace	Select an existing namespace or enter a unique namespace in which to include the orchestration dependency. Design Studio uses the last saved namespace as the default value.

Orchestration Dependency Editor Details Tab

Use the **Details** tab to specify the order components that have dependencies.

Dependency Source

Use the Dependency Source area to define the source order components upon which the Dependency Target order components will have dependencies.

Field	Use
Base Component **Functions** **Search**	Use these fields to limit the number of order components that appear in the Dependency Source area. The Dependency Source area displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.
Dependency Source	Specify the order components that contain the order items on which the order items defined in the **Dependency Target** field depend. • Click **New** to create a new order component specification. See "Creating New Order Component Specifications" for more information. • Click **Open** to open the corresponding specification in the Order Component Specification editor.

Dependency Target

Field	Use
Base Component **Functions** **Search**	Use these fields to limit the number of order components that appear in the Dependency Source area. The Dependency Source area displays all order component specifications defined in the workspace. Use **Base Component** to limit the display to a single root-level order component and its dependencies. Use **Functions** to limit the scope to executable order components (order components associated with a process). Enter a value in the **Search** field to search for a specific order component.
Dependency Target	Specify the order components that cannot run until the order items included in order components specified in the **Dependency Source** field have met their specified conditions. • Click **New** to create a new order component specification. See "Creating New Order Component Specifications" for more information. • Click **Open** to open the corresponding specification in the Order Component Specification editor.

Selected Order Component

Select a checked order component in the Order Source area and define an XQuery condition that must evaluate to true for the dependency to exist. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls.

Orchestration Dependency Editor Wait for Condition Tab

Use the **Wait For Condition** tab to specify the conditions that must be satisfied before an order component with dependencies on another order component can run.

Wait For Condition

Use the Wait for Condition area to specify the conditions that must be satisfied before the transition can occur.

Field	Use
Task State Completed	Select this option to specify that the dependency target order component cannot run until the subprocess task associated with the dependency source order component reaches the completed state.

Field	Use
Data Change Notification	Select this option to indicate that the dependency target order component cannot run until the value you select for the data change notification property of the order item changes. For example, you can configure a data change notification on a status field in the order template so that if the status changes to Failed, OSM generates a fallout notification to an administrator. When you select **Data Change Notification**, you also define the following fields: • In the **Order Item** field, select the order item specification upon which the dependency target order component must wait. • In the **Data Change Notification Property** field, select the order item property that contains the data change notification property value upon which the dependency target order component must wait. See "Order Item Specification Editor Order Item Properties Tab" for more information about defining order item properties. • In the **Relative Path** field, you can specify the location of the data change notification property. • In the Data Change Condition Expression area, you can configure an XQuery expression to denote that the transition wait until the property contains a specific value. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Order Data Change Wait Condition XQuery Expressions" for more information about this XQuery field.

Wait Delay

Use the Wait Delay area to configure a delay in addition to the information defined in the Wait For Condition area. You configure the wait delay against either the **Task State** or the **Data Change Notification** wait condition options. After the conditions specified in the Wait for Condition area are met, OSM delays the processing of the corresponding target order component using the information defined here.

The wait delay setting defers the start of the waiting order component until *after* the dependencies have been resolved. For example, you can use this to defer the resolution of the condition until two days after a task completes.

> **Note:**
>
> If order items have Wait Delay making it impossible to complete the order component on the requested delivery date, **Use Calculated Start Date** is ignored. See "Order Component Specification Editor Details Tab" for more information.

Field	Use
Wait Delay area	In the Wait Delay area, do one of the following: • Select **Duration** and use the **Duration** fields to specify the amount of time to wait after the conditions defined in the Wait For Condition area is met. • Select **Date Time Expression** and use the **Duration Expression** field to use XQuery to set the date and time for the transition to occur.

Field	Use
Duration fields	Use these fields to set a static amount of time to delay. Only available if the **Duration** option is selected.
Duration Expression	Create an XQuery expression in the **Duration Expression** field to run against a specific order item property. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. If the **Duration** option is selected, write the XQuery expression to return a string in the xs:duration format (for example,"PT15M"). See "About Wait Delay Duration XQuery Expressions" for more information. If the **Date Time Expression** option is selected, write the XQuery expression to return a string in the xs:dateTime format (for example, "2012-09-15T15:30:00-4:00"). See "About Wait Delay Date and Time XQuery Expressions" for more information.

Related Topics

Orchestration Dependency Editor

Creating New Orchestration Dependencies

Orchestration Dependency Editor Order Item Dependencies Tab

Use the **Order Item Dependencies** tab to define the property order items must share to create a dependency.

By default, an order item in the **To Component** field has a dependency when the same order item exists in the **From Component** field. However, you can use order item dependencies if you want to create a dependency on two different order items. For example, order items in a shipping order component may not be identical to order items in an install order component, but a dependency between those order items exists.

Field	Use
Order Item	The order item dependency default value is **Order Item**. To create order item dependencies between two different order items based on a different attribute, select a different option in this tab.
Fulfillment Pattern	Select to create a dependency on two different order items that share the same orchestration fulfillment pattern.
Property Correlation	Select to create a dependency on two different order items that share the same property correlation. If you select this option, enter an XQuery expression to select the order item property that order items in the dependency target order component must share with order items in the dependency source order component. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Order Item Dependency Property Correlation XQuery Expressions" for more information about this XQuery field.

Related Topics

Orchestration Dependency Editor

Creating New Orchestration Dependencies

17

Working with Fulfillment State Maps

Oracle Communications Order and Service Management (OSM) uses fulfillment state maps to contain common fulfillment states and mappings of those states to external fulfillment states. These mapping definitions are used in order fulfillment state composition rules and order item fulfillment state composition rules and point of no return rules.

Fulfillment state mappings consist of a mapped fulfillment state, an external fulfillment state, and various criteria to determine when that mapped fulfillment state is applied. At a minimum, you must specify fulfillment pattern, order item, orchestration sequence, orchestration stage, and order component.

When working with fulfillment state maps, refer to the following topics:

- About Fulfillment State Maps
- About Configuring Fulfillment States
- Creating New Fulfillment State Maps
- Fulfillment State Map Editor

About Fulfillment State Maps

Fulfillment state maps are containers for common fulfillment states and fulfillment state mappings. Common fulfillment states are user-defined values that are used to provide consistent state information about various entities. Fulfillment state *mappings* are the entities that contain the actual mapping information, and fulfillment state *maps* are just containers for the information. These containers are logical groupings you set up for convenience. Functionally, it does not matter whether you have one or many fulfillment state maps. Each common fulfillment state is available to all of the fulfillment state mappings in the workspace, regardless of which fulfillment state map it is configured in. This means that each common fulfillment state needs to be unique in the workspace, regardless of which fulfillment state map it is located in.

Common fulfillment states have two functions:

- They are used as the result of the fulfillment state mappings. When they are used this way, they are referred to as mapped fulfillment states.
- They are also used as the result of the composition rules. When they are used this way, they are referred to as composite fulfillment states. (For more information about composition rules, see "About Order Item Fulfillment State Composition Rule Sets" and "About Order Fulfillment State Composition Rule Sets".)

However they are going to be used, they are configured in a single list using the Fulfillment State Maps editor. You do not need to know how the common fulfillment state will be used when you configure it, and the same common fulfillment state can be used for both purposes at the same time.

After the common fulfillment states have been created, you create the mappings. The purpose of a fulfillment state mapping is to map an external fulfillment state to a common fulfillment state. However, each mapping must also specify further conditions to indicate when it should be applied: a single orchestration fulfillment pattern, order item, and orchestration sequence,

with a single set of orchestration stage and order component combinations. There are no wild cards, which may lead to a large number of mappings. In addition, you can further restrict the application of the mapping by adding any of the following: fulfillment mode, property/property value combinations, and current fulfillment state of the order component.

Related Topics

About Configuring Fulfillment States

Creating New Fulfillment State Maps

Fulfillment State Map Editor

About Configuring Fulfillment States

Fulfillment state maps are a key part of configuring fulfillment states. The fulfillment state management configuration consists of the following:

1. External fulfillment states: External fulfillment states relate to the status information sent to OSM by fulfillment systems. An external fulfillment state defined on one order component is available on that component and on any order component that extends that order component. The external fulfillment state is an input to the fulfillment state mappings. See "Order Component Specification Editor External Fulfillment States Tab" for more information.
2. Fulfillment State Maps: This entity provides the normalization of the external system's states into mapped fulfillment states. This is also where common fulfillment states are created. See "About Fulfillment State Maps" for more information.
3. Order Item Fulfillment State Composition Rule Sets: This entity provides the composition of these mapped fulfillment states into composite fulfillment states reflecting fulfillment states for order items. See "About Order Item Fulfillment State Composition Rule Sets" for more information.
4. Order Fulfillment State Composition Rule Sets: This entity provides the same function as the Order Item Fulfillment State Composition Rule Sets, but at the order level instead of the order-item level. See "About Order Fulfillment State Composition Rule Sets" for more information.

> **Note:**
>
> If you are adding fulfillment state processing to cartridges that were created in a pre-7.2 version of OSM, you also need to add data elements to the Data Dictionary for fulfillment state. See the discussion of modeling order template structures for fulfillment states in the *OSM Developer's Guide*.

In addition to providing order item and order fulfillment state information to upstream systems, fulfillment states may also be used to restrict processing of order amendments from the upstream system. This is done using the point of no return processing in the orchestration fulfillment pattern. See "About Points of No Return" for more information.

Creating New Fulfillment State Maps

You create fulfillment state maps to associate external fulfillment states with common fulfillment states. This fulfillment state map contains the mapping definitions that are used to determine

fulfillment states that are used in order fulfillment state composition rules, order item fulfillment state composition rules, and point of no return rules.

To create a new fulfillment state map:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Fulfillment State Map**.

   The Fulfillment State Map wizard is displayed.

2. In the **Project** field, select the project in which to save the fulfillment state map.
3. In the **Name** field, enter a name for the fulfillment state map.

   The name must be unique among fulfillment state map entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the fulfillment state map.

   Design Studio uses the last saved namespace as the default.

5. (Optional) Select a location for the fulfillment state map.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.
   b. Navigate to the directory in which to save the entity.
   c. Click **OK**.

6. (Optional) Select the check box for **Create default states**. This option creates the following set of fulfillment states in your fulfillment state map:

   - canceled
   - completed
   - failed
   - inProgress
   - pending

   > **Note:**
   >
   > Fulfillment states can only be defined once per workspace, so if any project in your workspace already has these fulfillment states defined, you should not select the check box.

7. Click **Finish**.

   Design Studio adds the new fulfillment state map to the Studio Projects view and opens the new entity in the Fulfillment State Map editor.

Related Topics

About Fulfillment State Maps

Fulfillment State Map Editor

Fulfillment State Map Editor

Use the Fulfillment State Map editor to create both common fulfillment states and the mappings that use them.

The following fields are common to all Fulfillment State Map editor tabs:

Field	Use
Description	Edit the display name of the fulfillment state map.
Namespace	Select an existing namespace or enter a unique namespace in which to include the fulfillment state map. Design Studio uses the last saved namespace as the default.

Many fields in this editor follow the following conventions:

- Field name as a link: If the name of the field is a link, clicking it performs one of two actions:
  - If there is an entry in the field, clicking the name of the field opens the item referenced in the field. For example, if the field name is **Fulfillment Pattern** and the entry in the field is **Product.Base**, clicking the label opens the **Product.Base** orchestration fulfillment pattern.
  - If there is no entry in the field, clicking the name of the field opens the wizard to create an item. For example, if the field name is **Order Item**, clicking the label opens the wizard to create a new order item specification.
- Clear button: This button is shaped like an X and is located after the field. If it is possible to clear the field, the X is red. If it is not possible to clear the field (for example, if it is already empty), the X is gray.
- **Select**: This button opens a dialog box with a list of the relevant entities. Select a value to enter it into the field.

When configuring fulfillment states and mappings in the Fulfillment State Map editor, refer to the following topics:

- Fulfillment State Map Mappings Tab
- Fulfillment State Map States Tab

Fulfillment State Map Mappings Tab

Use the **Mappings** tab to create and maintain mappings between an external fulfillment state and common project fulfillment states that are used in order and order item composition rules and point of no return rules. The mappings are based on criteria the criteria used to generate it. These criteria include: orchestration fulfillment pattern, order item, orchestration sequence, orchestration stage, and order component.

> **Note:**
>
> See "Fulfillment State Map Editor" for information about fields that appear on all of the Fulfillment State Map editor tabs and for information about field conventions.

Field	Use
Filters	Expand this section and select one or more values to constrain the mappings shown in the fulfillment state mapping table that follows. If more than one filter is specified, only entries in the table that meet all of the criteria are displayed. Note that the filter fields recognize inheritance. For example, if an orchestration fulfillment pattern of **basePS** is chosen in the filter, orchestration fulfillment patterns that extend **basePS** are also included in the resulting list.
Fulfillment state mapping table	Displays a list of the fulfillment state mappings that have been defined for the fulfillment state map, subject to the specified filter criteria. Do any of the following: • Select an entry in the table to edit it using the fields on this tab. • Click **Duplicate** to create a duplicate of the selected entry. Use this command as a shortcut to quickly create another mapping that is similar to the original entry. • Click **Remove** to remove the selected entry. • Select **Add** to add an entry. You cannot create a new entry by selecting a blank row in the table.
Mapped Fulfillment State	Enter a fulfillment state to be assigned to the mapping's specified order item for the mapping's specified order component.
Processing State	Enter a processing state to be mapped to the fulfillment state.
Fulfillment Pattern	Enter an orchestration fulfillment pattern. This mapping only affects order components defined on orchestration plans associated with the specified orchestration fulfillment pattern.
Fulfillment Mode	(Optional) Select a fulfillment mode to restrict this mapping to order components defined on orchestration plans associated with the specified orchestration fulfillment pattern/fulfillment mode combination. This field is disabled until an orchestration fulfillment pattern is selected. Unlike most other fields, a new fulfillment mode cannot be created directly from the field name's hyperlink.
Order Item	Enter the order item specification for which the mapped fulfillment state is to be set.
Properties	(Optional) Select one or more properties of the selected order item specification if you want to restrict the mapping to order items with the specified property values.
Property Value	Select a property from the **Properties** list to enter a value in the **Property Value** field. If there is no property defined and selected, you cannot add a property value. However, if you have entered a property, you must define a value for it or the property will be ignored at run time.
Sequence	Enter the orchestration sequence that produces the orchestration stage that in turn produces the order components to which the mapping applies.
Orchestration Stage	Select one or more orchestration stages that produce the order components to which the mapping applies. This field is disabled until an orchestration sequence is selected.
Order Component	Select a stage in the **Orchestration Stage** list to select an order component that is either produced by that stage or extends an order component that is produced by that stage. The mapping only applies to this component and its descendants. Additionally, at least one of these components or its hierarchical ancestor must define an external fulfillment state. This field is disabled until an orchestration stage is selected. A new order component cannot be created from the field name's link.

Field	Use
External Fulfillment State	Enter the external fulfillment state to which this mapping applies.
Current Fulfillment State	Enter a common fulfillment state (optional) for the mapping to apply only if the order item already has a specific fulfillment state.

Related Topics

Fulfillment State Map Editor

Fulfillment State Map States Tab

Use the **States** tab to define the fulfillment states that are used in mappings, composition rules, and point of no return rules.

> **Note:**
>
> See "Fulfillment State Map Editor" for information about fields that appear on all of the Fulfillment State Map editor tabs and for information about field conventions.

Field	Use
Fulfillment States	Displays the fulfillment state definitions that can be used in fulfillment state mappings, order fulfillment state composition rules, order item fulfillment state composition rules, and point of no return rules. Do any of the following: • Select an existing fulfillment state to edit it. • Click **Add** to add a new fulfillment state. If an existing fulfillment state is selected, the new fulfillment state is added as a child of the selected state. If a child fulfillment state is selected, this button is disabled because fulfillment states are only allowed one level of descendants. • Click **Remove** to remove the selected fulfillment states. • Right-click a selected fulfillment and select **Rename** to rename the selected fulfillment state.
Details subtab	Edit the display name of the fulfillment state. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Information subtab	Enter any additional information about the fulfillment state that is required by your specific situation. If you are localizing OSM, use the list above the field to set different values for different locales.

Related Topics

Fulfillment State Map Editor

18

Working with Order Item Fulfillment State Composition Rule Sets

Oracle Communications Order and Service Management (OSM) uses order item fulfillment state composition rule sets to determine composite fulfillment states for order items. The rule sets contain rules and conditions that determine the overall fulfillment state of an order item based on the composite fulfillment states of its child order items and the mapped fulfillment states of its order components.

When working with order item fulfillment state composition rule sets, see the following topics:

- Order Item Fulfillment State Composition Rule Set Editor
- Creating New Order Fulfillment State Composition Rule Sets
- Order Item Fulfillment State Composition Rule Set Editor

About Order Item Fulfillment State Composition Rule Sets

Fulfillment state composition rules for order items are defined in order item composition rule sets. For order items, there is at most a single fulfillment state rule set per orchestration fulfillment pattern and order item specification combination.

A fulfillment state composition rule specifies the composite fulfillment state for the order item when all of the conditions are met. If there are separate situations that can result in the same fulfillment state (for example, use this composite state if either any of the input states is completed, or if none of the states is failed), create additional rules that evaluate to the same composite fulfillment state.

Related Topics

Creating New Order Item Fulfillment State Composition Rule Sets

Order Item Fulfillment State Composition Rule Set Editor

Creating New Order Item Fulfillment State Composition Rule Sets

You create new order item fulfillment state composition rule sets in Design Studio to define a composite fulfillment state based on the fulfillment states of lower-level order items.

To create new order item fulfillment state composition rule sets:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Order Item Fulfillment State Composition Rule Set**.

   The Order Item Fulfillment State Composition Rule Set wizard is displayed.
2. In the **Project** field, select the project in which to save the composition rule set.
3. In the **Name** field, enter a name for the composition rule set.

The name must be unique among order item fulfillment state composition rule set entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the composition rule set.

   Design Studio uses the last saved namespace as the default.

5. (Optional) Select a location for the composition rule set.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new composition rule set to the Studio Projects view and opens the new entity in the Order Item Fulfillment State Composition Rule Set editor.

Related Topics

About Order Item Fulfillment State Composition Rule Sets

Order Item Fulfillment State Composition Rule Set Editor

Order Item Fulfillment State Composition Rule Set Editor

Use the Order Item Fulfillment State Composition Rule Set editor to define the composition rules that apply to a particular order item.

Field	Use
Description	Edit the display name of the order item fulfillment state composition rule set.
Namespace	Select an existing namespace or enter a unique namespace in which to include the order item fulfillment state composition rule. Design Studio uses the last saved namespace as the default.
Fulfillment Pattern	Select the orchestration fulfillment pattern to which the rules in the rule set apply. The orchestration fulfillment pattern hierarchy is honored here. If a rule is defined for a base orchestration fulfillment pattern, it will apply to the fulfillment patterns that extend it, unless a rule is defined for the child fulfillment pattern specifically.
Order Item	Select the order item to which the rules in the rule set apply.

Field	Use
Composition rule and condition list	The rules and conditions are evaluated in the order listed: use the **Move Up** and **Move Down** commands to ensure that higher priority rules and more restrictive conditions are listed first. The root-level items in this list are the rules, and the child items are the conditions. Do any of the following: • Select an existing rule or condition to edit it. • Click **Add Rule** to create a new order item fulfillment state composition rule. • Click **Add Condition** to create a new order item fulfillment state composition condition under the selected rule. • Click **Remove** to remove the selected rules and conditions. • Right-click a selected rule and select **Rename** to rename the rule or condition. • Right-click a selected rule or condition and select **Move Up** to move the selected rule or condition higher in the list. • Right-click a selected rule or condition and select **Move Down** to move the selected rule or condition lower in the list. See the following for more information about configuring composition rules: • Order Item Fulfillment State Rule Details Subtab • Order Item Fulfillment State Rule Information Subtab

Related Topics

Order Item Fulfillment State Composition Rule Set Editor

Creating New Order Item Fulfillment State Composition Rule Sets

Order Item Fulfillment State Rule Details Subtab

Use the **Rule Details** subtab to specify the composite fulfillment state to be assigned to the associated order item when the selected rule's conditions are met. If no rule is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the composition rule. If you are localizing OSM, use the drop-down list to the right of the field to set different values for different locales.
Composite State	Select the fulfillment state to be assigned to the composition rule's order item when all of the rule's composition conditions are met.

See the following for more information about configuring rules and conditions on the **Rule Details** subtab:

- Order Item Fulfillment State Condition Details Subtab
- Order Item Fulfillment State Condition Information Subtab

Related Topics

Order Item Fulfillment State Composition Rule Set Editor

Order Item Fulfillment State Condition Details Subtab

Use the **Condition Details** subtab to specify the criteria that must be met for the selected condition to be evaluated. If no condition is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the condition. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Type	Select **Any** when the condition requires at least one of the input order item's fulfillment states to match one of the selected fulfillment states. Select **All** when the condition requires all of the input order item's fulfillment states to match one of the selected fulfillment states. Select **None** when the condition requires that none of the input order item's fulfillment states match any of the selected fulfillment states.
Fulfillment State tree	Select one or more fulfillment states for matching to the rule's input fulfillment states. If a parent fulfillment state is selected, it includes the child fulfillment states also.
Property Values	Select one or more order item properties, and then enter a value for each. For the condition to be met, the order item must contain the specified property and that property must have the specified value. If both property values and fulfillment states are defined on the same condition, both the property value and fulfillment state conditions must be met. Do any of the following: • Click **Select** to select from a list of the properties for the order item. • Click **Open** to open the appropriate order item specification with the property selected. • Select a property from the list and click **Remove** to remove the property from the list.

Related Topics

Order Item Fulfillment State Rule Details Subtab

Order Item Fulfillment State Condition Information Subtab

Use the **Condition Information** subtab to capture additional information about the condition.

Field	Use
Information	Enter any additional information about the condition that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Order Item Fulfillment State Rule Details Subtab

Order Item Fulfillment State Rule Information Subtab

Use the **Rule Information** subtab to capture additional information about the rule.

Field	Use
Information	Enter any additional information about the composition rule that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Order Item Fulfillment State Composition Rule Set Editor

19

Working with Order Fulfillment State Composition Rule Sets

Oracle Communications Order and Service Management (OSM) uses order fulfillment state composition rule sets to determine composite fulfillment states for orders. The rule sets contain rules and conditions that determine the overall fulfillment state of an order based on the composite fulfillment states of the root-level order items.

When working with order fulfillment state composition rule sets, see the following topics:

- About Order Fulfillment State Composition Rule Sets
- Creating New Order Fulfillment State Composition Rule Sets
- Order Fulfillment State Composition Rule Set Editor

About Order Fulfillment State Composition Rule Sets

Fulfillment state composition rules for the order are defined in order composition rule sets. For orders, there is at most a single fulfillment state rule set per order definition.

A fulfillment state composition rule specifies the composite fulfillment state for the order item when all of the conditions are met. If there are separate situations that can result in the same fulfillment state (for example, use this composite state if either any of the input states is completed, or if none of the states is failed), create additional rules that evaluate to the same composite fulfillment state.

Related Topics

Creating New Order Fulfillment State Composition Rule Sets

Order Fulfillment State Composition Rule Set Editor

Creating New Order Fulfillment State Composition Rule Sets

You create new order fulfillment state composition rule sets in Design Studio to define a composite fulfillment state based on the fulfillment states of the order items.

To create new order fulfillment state composition rule sets:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Order Fulfillment State Composition Rule Set**.

   The Order Fulfillment State Composition Rule Set wizard is displayed.
2. In the **Project** field, select the project in which to save the composition rule set.
3. In the **Name** field, enter a name for the composition rule set.

   The name must be unique among order fulfillment state composition rule set entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the composition rule set.

Design Studio uses the last saved namespace as the default.

5. (Optional) Select a location for the composition rule set.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new composition rule set to the Studio Projects view and opens the new entity in the Order Fulfillment State Composition Rule Set editor.

Related Topics

About Order Fulfillment State Composition Rule Sets

Order Fulfillment State Composition Rule Set Editor

Order Fulfillment State Composition Rule Set Editor

Use the Order Fulfillment State Composition Rule Set editor to define the composition rules that apply to a particular order.

Field	Use
Description	Edit the display name of the order fulfillment state composition rule set.
Namespace	Select an existing namespace or enter a unique namespace in which to include the order fulfillment state composition rule. Design Studio uses the last saved namespace as the default.
Order	Select the order to which the rules in the rule set apply.
Composition rule and condition list	The rules and conditions are evaluated in the order listed: use the **Move Up** and **Move Down** commands to ensure that higher priority rules and more restrictive conditions are listed first. The root-level items in this list are the rules, and the child items are the conditions. Do any of the following: • Select an existing rule or condition to edit it. • Click **Add Rule** to create a new order fulfillment state composition rule. • Click **Add Condition** to create a new order fulfillment state composition condition under the selected rule. • Click **Remove** to remove the selected rules and conditions. • Right-click a selected rule and select **Rename** to rename the rule or condition. • Right-click a selected rule or condition and select **Move Up** to move the selected rule or condition higher in the list. • Right-click a selected rule or condition and select **Move Down** to move the selected rule or condition lower in the list. See the following for more information about configuring composition rules: • Order Fulfillment State Rule Details Subtab • Order Fulfillment State Rule Information Subtab

Related Topics

About Order Fulfillment State Composition Rule Sets

Creating New Order Fulfillment State Composition Rule Sets

Order Fulfillment State Rule Details Subtab

Use the **Rule Details** subtab to specify the composite fulfillment state to be assigned to the associated order when the selected rule's conditions are met. If no rule is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the composition rule. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Composite State	Select the fulfillment state to be assigned to the composition rule's order when all of the rule's composition conditions are met.

See the following for more information about configuring rules and conditions on the **Rule Details** subtab:

- Order Item Fulfillment State Condition Details Subtab
- Order Fulfillment State Condition Information Subtab

Related Topics

Order Fulfillment State Composition Rule Set Editor

Order Item Fulfillment State Condition Details Subtab

Use the **Condition Details** subtab to specify the criteria that must be met for the selected condition to be evaluated. If no condition is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the condition. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Type	Select **Any** when the condition requires at least one of the input order item's fulfillment states to match one of the selected fulfillment states. Select **All** when the condition requires all of the input order item's fulfillment states to match one of the selected fulfillment states. Select **None** when the condition requires that none of the input order item's fulfillment states match any of the selected fulfillment states.
Fulfillment State tree	Select one or more fulfillment states for matching to the rule's input fulfillment states. If a parent fulfillment state is selected, it includes the child fulfillment states also.

Field	Use
Property Values	Select one or more order item properties, and then enter a value for each. For the condition to be met, the order item must contain the specified property and that property must have the specified value. If both property values and fulfillment states are defined on the same condition, both the property value and fulfillment state conditions must be met. Do any of the following: • Click **Select** to select from a list of the properties for the order item. • Click **Open** to open the appropriate order item specification with the property selected. • Select a property from the list and click **Remove** to remove the property from the list.

Related Topics

Order Item Fulfillment State Rule Details Subtab

Order Fulfillment State Condition Information Subtab

Use the **Condition Information** subtab to capture additional information about the condition.

Field	Use
Information	Enter any additional information about the condition that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Order Item Fulfillment State Rule Details Subtab

Order Fulfillment State Rule Information Subtab

Use the **Rule Information** subtab to capture additional information about the rule.

Field	Use
Information	Enter any additional information about the composition rule that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Order Item Fulfillment State Composition Rule Set Editor

20

Working with Composite Cartridge Views

Composite cartridge views can define task views, which enable you to define additional task data and views for generic tasks, such as creation tasks and query tasks, which are typically stored in sealed cartridges. After the data is defined, you can contribute it to a composite cartridge build so that the sealed cartridges can access it.

When working with Oracle Communications Order and Service Management (OSM) composite cartridge views, see the following topics:

About Composite Cartridge Views

When you extend an existing OSM solution, for example, by adding a new fulfillment function, the tasks of the new function typically require additional task data to be added. For example, a new Shipping function might require a Shipping ID and Start Date in addition to the control data that is generated for the function.

Once you have added the required data to the task, you can define the task data and any required behaviors using the tabs of the Composite Cartridge View editor. Oracle recommends that you create the composite cartridge view in the component cartridge that is hosting the new component; for example, the component cartridge that hosts the new fulfillment function.

Once the data is defined, you can contribute it to required tasks, such as creation tasks and query tasks, using the **Task Data Contribution** tab of the Composite Cartridge editor. See "Composite Cartridge Editor Task Data Contribution Tab" for more information.

During the composite cartridge build, the additional task data and views are added into the build artifact, within the context of the solution.

After you have added the task data, you can view the entire data set for an order or for a task by using the order or tasks's **Composite Data View** tab.

Related Topics

Creating New Cartridge Composite Views

Adding Task Data to a Composite Cartridge View

Contributing Task Data to a Composite Cartridge

Composite Cartridge View Editor

Creating New Cartridge Composite Views

You create Cartridge Composite Views when you want to add additional data to required tasks, such as creation tasks and query tasks, which are typically stored in sealed cartridges.

Oracle recommends that you create the view in the same component cartridge that hosts the extended functionality; for example, a new fulfillment function.

To create new cartridge composite views:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Order Management**, then select **Composite Cartridge View**.

   The Composite Cartridge View wizard is displayed.

2. In the **Project Name** field, select the project in which the composite cartridge view will be saved.

3. (Optional) In the **Extends** field, select an existing view to extend the task data and behavior of an existing task or composite cartridge view.

   Click **Select** to open the Select Extends dialog box, and do one of the following:

   - Select an existing view.
   - If a suitable view does not yet exist, click **New** in the Select Extends dialog box to create the view.

   When finished, click **OK**. Your selection populates the corresponding **Extends** field in the Composite Cartridge View wizard.

4. In the **Order** field, select an order to associate with the composite cartridge view.

5. In the **Name** field, enter a name for the composite cartridge view.

   The name must be unique among composite cartridge views within the same namespace.

6. (Optional) Select a location for the composite cartridge view.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

7. Click **Finish**.

   Design Studio adds the new composite cartridge view to the Studio Projects view and opens the new entity in the Composite Cartridge View editor.

Related Topics

Adding Task Data to a Composite Cartridge View

Contributing Task Data to a Composite Cartridge

Working with Composite Cartridge Views

Adding Task Data to a Composite Cartridge View

You can add task data to a composite cartridge view in a variety of ways; for example, by adding the data to the task directly, by selecting existing data from the Order Template or Data Dictionary, or by inheriting data from an extension task.

To add task data to a composite view:

1. In the Composite Cartridge View editor, click the **Task Data** tab and model the data as described in "Defining Task Data".
2. (Optional) Define behaviors for manual tasks as described in "Creating New Behaviors".
3. (Optional) In the **Details** tab, select or create a task from which to extend this task.

> **Note:**
>
> When creating a composite cartridge view for a creation task, Oracle recommends that you extend from the base task. Creation tasks must have visibility to all of the data required to create the order.

4. If an order has not already been associated with the task, select an order, or create a new order.

Related Topics

Working with Tasks

Working with Behaviors

Contributing Task Data to a Composite Cartridge

Task data that has been added to a composite cartridge view can be contributed to required tasks, such as creation tasks and query tasks, using the **Task Data Contribution** tab of the Composite Cartridge editor.

To contribute task data to required tasks:

1. Create a new composite cartridge view.

   See "Creating New Cartridge Composite Views" for more information.
2. Model the data using the tabs of the Composite Cartridge View editor.

   See "Adding Task Data to a Composite Cartridge View" for more information.
3. Ensure that the component cartridge that contains the composite cartridge view has been added as a project dependency.

   See "Project Editor Dependency Tab" for more information.
4. In the Composite Cartridge editor, click the **Task Data Contribution** tab.
5. Click **Add** in either the Task Data Contribution area or the Query Task Data Contribution area depending on the type of task to which you are contributing.
6. In the Task Data Contribution Selection dialog box, add the desired combination of order, process, task, and view for the task data contribution.

Similarly, in the Query Task Data Contribution Selection dialog box, add the desired combination of order, role, task, and view for the task data contribution.

Click **Select** to select an entity from the selection list and click **OK**. If a suitable entity does not yet exist, click **New** to create a new entity using the wizard. Click **Open** to open the corresponding editor for the entity. When finished, click **OK**.

> **Tip:**
>
> You can define multiple task data contributions for a required task. For example, a creation task could get a contribution from each fulfillment function in a solution (SyncCustomerCreationView, BillingCreationView, ProvisioningCreationView, and so on).

The task data defined in the composite cartridge view is added to the required tasks at build time, within the context of the solution. The original task names and entity references are not changed.

Composite Cartridge View Editor

Use the Composite Cartridge View editor to model additional task data to contribute to required tasks, such as creation tasks and query tasks, which are typically contained in sealed cartridges.

The Composite Cartridge View editor uses the same **Task Data** and **Behaviors** tabs as the Task editor. For details about the fields on each tab, see the following topics:

- Task Editor Task Data Tab
- Task Editor Behaviors Tab
- Composite Cartridge View Editor Details Tab

Composite Cartridge View Editor Details Tab

Use the Composite Cartridge View editor **Details** tab to extend the task definition.

Field	Use
Extends	Select or create a task from which this task will inherit data. Using task inheritance, you can leverage existing task data when building new, similar tasks. To remove a task extension, you must manually clear this field. See "About Task Extensions and Inheritance" for more information.
Order	Contains the order to associate with this task. If the correct order was not already associated when the task was created, do one of the following: • Click **Select** to choose an order from the selection box. • Click **New** to create a new Order entity. • Click **Open** to view in the Order editor the order with which you want to associate the task. The order associated with a task determines the overall data set that will be available to the task when you model the task data.

21

Working with Composite Cartridge Projects

An Oracle Communications Order and Service Management (OSM) composite cartridge is a type of cartridge that references other cartridges, called component cartridges, within a single logical scope. You use a composite cartridge to assemble an OSM solution from a collection of component cartridges. When you deploy the composite cartridge into the run-time environment, all of the component cartridges that are referenced in the composite cartridge are deployed as a solution in a single action.

Composite cartridge projects may contain any number of component cartridges, but not other composite cartridges. You cannot create entities directly in a composite cartridge project; rather, you create the entities in the component cartridges that are referenced in the composite cartridge. You use component cartridges and composite cartridges together to develop, deploy, and maintain OSM solutions.

When working with OSM composite cartridge projects, see the following topics:

About Composite Cartridge Projects

You use composite cartridge projects to extend the functionality of existing component cartridges, without having to unseal the component cartridge and modify the contents directly.

For example, consider a typical central order management solution which consists of the following component cartridges:

- Base Cartridge v1.0
- Billing Cartridge v1.0
- Provisioning Cartridge v1.0
- Fulfillment Pattern Cartridge v1.0
- Product Mapping Cartridge v1.0
- Composite Cartridge v.1.0

If you wanted to extend this solution by adding a new Shipping fulfillment function, the new solution would consist of the following cartridges:

- (New) Shipping Cartridge v1.0:

A new Shipping cartridge is created to host the new fulfillment function, decomposition rules, processes and tasks, and any additional task data required by the new fulfillment function.

- (Sealed) Base Cartridge v1.0:

  The sealed base cartridge that serves as the foundation for the original solution remains sealed and unchanged. It contains such entities as the order, order item specification, order recognition rules, and required tasks for the order. It is deployed as part of the solution.

- (Unchanged) Billing Cartridge v1.0:

  The billing cartridge is unchanged but is deployed as part of the solution.

- (Unchanged) Provisioning Cartridge v1.0:

  The provisioning cartridge is unchanged but is deployed as part of the solution.

- (Updated) Fulfillment Pattern Cartridge v2.0:

  The v1.0 fulfillment pattern cartridge is updated to include the new shipping dependencies along with the billing and provisioning dependencies. The cartridge is versioned and deployed as part of the solution. This cartridge would generally not be sealed, as orchestration fulfillment patterns are not usually reusable components of the solution.

- (Unchanged) Product Mapping Cartridge v1.0:

  The product mapping cartridge is unchanged but is deployed as part of the solution.

- (Updated) Composite Cartridge v2.0:

  A new composite cartridge is created to reference all other cartridges in the solution; specifically, Shipping Cartridge v1.0, Base Cartridge v1.0, Billing Cartridge v1.0, Provisioning Cartridge v1.0, and Fulfillment Pattern Cartridge v2.0.

In this way, you can extend any cartridge that has been deployed to an OSM run-time environment, including Order-To-Activate cartridges in the Oracle Communications Order to Cash Integration Pack for OSM.

See *OSM Developer's Guide* for cartridge development guidelines and solution-versioning best practices.

Related Topics

Working with Composite Cartridge Projects

Composite Cartridge Editor

Extending Component Cartridges

Creating New Composite Cartridge Projects

You create an OSM composite cartridge project the same way you create any other type of Design Studio project.

To develop an OSM composite cartridge in Design Studio, you must first create an OSM composite project. See "Creating New Cartridge Projects" for more information.

Related Topics

About Composite Cartridge Projects

Composite Cartridge Editor

Working with Composite Cartridge Projects

Building and Packaging Composite Cartridge Projects

OSM composite cartridge projects are built and packaged in much the same way as other Design Studio cartridge projects. However, there are some differences you should be aware of.

Aggregating Non-Orchestration Entities

In OSM non-orchestration cartridges, entities are bound in scope at runtime to the cartridge in which they are created. It is not possible for non-orchestration entities to reach across cartridge boundaries to resolve references. For example, a process in one cartridge cannot refer to tasks or data defined in another deployed cartridge.

In contrast, OSM orchestration entities in one cartridge can reference entities with the same namespace in other cartridges in the same workspace. For example, an orchestration fulfillment pattern in one cartridge can reference order components that have been defined and deployed in another cartridge.

At build and packaging time, a composite cartridge acts as a aggregator of all non-orchestration entities defined across all of the component cartridges in the solution. These entities are automatically packaged into a new component cartridge with the same name and namespace as the composite cartridge.

Building and Packaging Composite Cartridges

When you build and package a composite cartridge, it is packaged as a single PAR file which contains:

- All non-orchestration entities aggregated and packaged into the composite cartridge
- A PAR file for each component cartridge referenced in the composite cartridge

Defer Packaging

Typically, Design Studio automatically packages your projects during incremental builds. You may choose to defer packaging for large projects when packaging slows the incremental build process and becomes cumbersome. If you defer packaging, the composite cartridge will not be packaged until you deploy the cartridge. See "Defining Packaging Preferences" for more information.

When you perform a clean build in Design Studio, the composite cartridge and all of the referenced component cartridges are packaged.

Optimize Deploy

Another option you may choose is Optimize Deploy. Using Optimize Deploy on a composite cartridge deploys only the entities in a component cartridge that have changed since the last build. See "Deploying Cartridge Projects with Optimize Deploy" for more information.

Related Topics

Building and Packaging Projects

Packaging and Deploying OSM Cartridges

Deploying Composite Cartridge Projects

In Design Studio, OSM composite cartridges are displayed in the Environment Perspective cartridge list the same way that other cartridges do.

When a composite cartridge is deployed, it includes all of the OSM non-orchestration entities and all component cartridges referenced in the composite cartridge, if they are either changed or not currently deployed.

Related Topics

Deploying Cartridge Projects from the Environment Perspective

Cartridge Management View

Studio Environment Editor

Undeploying Composite Cartridges with Shared Component Cartridges

By default, when you undeploy a composite cartridge with some component cartridges that are being used by other solutions in the run-time environment, the shared component cartridges remain deployed, and the composite cartridge and component cartridges that are not shared by other solutions are undeployed.

> **Note:**
>
> When multiple versions of a composite cartridge are running in a Production environment, you can undeploy an earlier version without impacting the currently-running versions.

Forcing the Undeployment of Shared Component Cartridges

To force the undeployment of shared component cartridges within a composite cartridge, use the **undeploy_shared_cartridge** variable in the Composite Cartridge editor **Cartridge Management Variable** tab. This variable forces the undeployment of the entire composite cartridge even if there are shared component cartridges contained within it.

For example, consider a scenario where two versions of a composite cartridge (1.0.0 and 2.0.0) are deployed in the run-time environment and share some component cartridges. If you set the **undeploy_shared_cartridge** variable of composite cartridge 1.0.0 to true then all of the component cartridges, including shared component cartridges, are undeployed. However, this forces composite cartridge version 2.0.0 to become invalid because its dependent cartridges no longer exist in the system.

> **Note:**
>
> Forcing the undeployment of shared component cartridges within a composite cartridge could render some solutions non-functional because their dependent component cartridges no longer exist in the run-time environment. This option should only be used in troubleshooting or run-time environment clean-up situations.

Undeploying Component Cartridges from a Composite Cartridge During Development

During the development of a composite cartridge, you may need to undeploy a component cartridge used by the solution. By default, component cartridges that are used by solutions in the run-time environment cannot be undeployed.

To force the undeployment of one or more component cartridges used by a solution, you can set the **allow_undeploy_component_cartridge_in_solution** parameter in the **oms-config.xml** file to **true**.

> **Note:**
>
> Forcing the undeployment of component cartridges that are used in a composite cartridge could render the solution non-functional. This option should only be used in troubleshooting or run-time environment clean-up situations.

Related Topics

Project Editor Cartridge Management Variables Tab

Cartridge Management View

Composite Cartridge Editor

Use the Composite Cartridge editor to configure composite cartridge projects. When you create a composite cartridge, you assemble all the component cartridges that make up the solution into a single archive file and deploy the file to the OSM run-time environment.

To access the Composite Cartridge editor, click any composite cartridge entity in the Studio Projects view. The editor contains a number of different tabs. For details about the fields on each tab, see the following topics:

- Project Editor Properties Tab
- Project Editor Copyright Tab
- Project Editor Dependency Tab
- Project Editor Tag Tab
- Project Editor Packaging Tab
- Project Editor Locations Tab
- Project Editor Model Variables Tab
- Project Editor Cartridge Management Variables Tab
- Composite Cartridge Editor Manifest Tab

- Composite Cartridge Editor Task Data Contribution Tab

Composite Cartridge Editor Manifest Tab

Use the **Manifest** tab to manage entity-level dependencies in a composite cartridge. Design Studio uses the following top-level entity types to determine all entity-level dependencies in a composite cartridge project:

- Order
- Fulfillment Pattern
- Order Recognition Rule

By default, all entities from all component cartridges that are referenced in the composite cartridge are included in manifest.

> **Note:**
>
> Design Studio populates the entity list based on project-level dependencies defined in the **Dependency** tab. If you have not defined any project dependencies, the entity list will be empty. See "Project Editor Dependency Tab" for more information.

Required Entities

Field	Use
Entity Type	Displays the top-level entity types that define the scope of the composite cartridge: Order, Fulfillment Pattern, and Order Recognition Rule. Select an entity type to display the required entities from component cartridges that are referenced in the composite cartridge. By default, all entities in all referenced component cartridges are included in the solution scope.
Entity List	Displays the required entities for the selected entity type. If the cartridge is sealed or read-only, the entity list is read-only. To exclude entities from the list, deselect the **Include all from project** check box, click **Select**, and from the selection list select only those entities that you want to include in the solution. For example, you may choose to exclude a new orchestration fulfillment pattern that is still in development and untested. Once the list is populated, you can click **Remove** to remove entities or **Open** to open the corresponding editor. **Note:** Excluding an entity in the **Manifest** tab removes it from the validation process only; packaging is not impacted. If the entity is referenced in the run-time environment, it will be available.
Include all from project	By default, the check box is selected. The entity list is automatically populated with the entities from all referenced component cartridges. To customize the entity list, deselect the check box and click **Select** to select one or more entities from the selection list.

Related Topics

Order and Service Management Project Editor

Composite Cartridge Editor Task Data Contribution Tab

Use the **Task Data Contribution** tab to define task data contributions to required tasks, such as creation tasks and query tasks, based on additional data modeled in the Composite Cartridge View editor.

When the composite cartridge builds, the task data and corresponding views will be available to the required tasks within the context of the solution.

Field	Use
Task Data Contribution	Displays a particular order, process, and task combination to which additional task data will be contributed. Additional task data is modeled in a Composite Cartridge View. (See "Adding Task Data to a Composite Cartridge View" for more information.) Click **Add** to create a new entry in the Task Data Contribution list using the Task Data Contribution Selection dialog box. In the Task Data Contribution Selection dialog box fields, do any of the following: • Click **Select** to select each entity in succession from the selection list. • Click **New** to create a new entity if a suitable one is not available. • Select **Open** to configure a selected entity in the corresponding editor. When you are finished in the Task Data Contribution Selection dialog box, click **OK**. Click **Remove** to delete an existing task data contribution from the list.
Query Task Data Contribution	Displays a particular order, role, and query task combination to which additional task data (modeled in a Composite Cartridge view) will be contributed. See "Adding Task Data to a Composite Cartridge View" for more information. Click **Add** to create a new entry in the Query Task Data Contribution list using the Query Task Data Contribution Selection dialog box. In the Query Task Data Contribution Selection dialog box fields, do any of the following: • Click **Select** to select each entity in succession from the selection list. • Click **New** to create a new entity if a suitable one is not available. • Select **Open** to configure a selected entity in the corresponding editor. When you are finished in the Query Task Data Contribution Selection dialog box, click **OK**. Click **Remove** to delete an existing query task data contribution from the list.

Related Topics

Contributing Task Data to a Composite Cartridge

Order and Service Management Project Editor

Extending Component Cartridges

Cartridge extensibility is the ability to extend the functionality of sealed cartridges by building on to them, rather than unsealing them and modifying the contents directly.

This type of extensibility applies to any OSM component cartridge, including productized cartridges.

To understand the features and functionality that enable cartridge extensibility, see the following topics:

- Working with Composite Cartridge Projects
- Working with Composite Cartridge Views

When working with OSM cartridge extensibility, see the following topics:

- Adding a New Fulfillment Function
- Adding a New Fulfillment Mode
- Adding a New Product
- Adding a New Service

Adding a New Fulfillment Function

This topic describes design steps to add a new fulfillment function.

To add a new fulfillment function for a new service offering:

1. Import all of the component cartridges to be included in the solution into Design Studio.

   For example, this might include a sealed base cartridge, Billing cartridge, Provisioning cartridge, Fulfillment Pattern cartridge, and Product Mapping cartridge.

2. Create a new Order and Service Management project to host the new fulfillment function.

   See "Creating New Cartridge Projects" for more information.

3. In the Order and Service Management Project editor **Properties** tab, deselect the **Standalone** check box.

   This allows the cartridge to be referenced in the composite cartridge as part of the solution (rather than as a standalone cartridge with no dependencies).

4. Delete the Order entity.

5. Create a base task for the new function that will serve as the base from which to extend all other new tasks.

   See "Creating New Tasks" and "About Task Extensions and Inheritance" for more information.

6. Create any other tasks required by the new function by extending from the base task created in the previous step.

7. Create a new process or extend from a process that will run when fulfilling the new function.

   See "Creating New Processes" for more information.

8. Create a new fulfillment function.

See "Adding New Functional Order Components" for detailed instructions.

9. Select an orchestration fulfillment pattern that maps to the new function or create a new orchestration fulfillment pattern.
10. (Optional) Create a new fulfillment system for the fulfillment function.

    Often, when you introduce a new fulfillment function, you will also require a new fulfillment system. See "Adding New Fulfillment Systems" for more information.
11. Create a new decomposition rule that maps from the fulfillment function to the fulfillment system.

    See "Creating New Decomposition Rules" for detailed information.
12. If you do not already have a composite cartridge to use, create a new Order and Service Management Composite project.

    See "Creating New Cartridge Projects" for more information.
13. Add the necessary cartridges, including the cartridge you created for the new fulfillment function, to the Dependency tab for the composite cartridge.

    See "Project Editor Dependency Tab" for more information.
14. Check entity-level dependencies.
15. Create a composite cartridge view in the cartridges containing the orders that will use the new fulfillment function.

    See "Creating New Cartridge Composite Views" for more information.
16. Add the data for the new tasks you created to the composite cartridge view.

    See "Adding Task Data to a Composite Cartridge View" for more information.
17. Create a task data contribution to extend the existing sales order creation task and order query task.

    See "Contributing Task Data to a Composite Cartridge" for more information.
18. Package and deploy the composite cartridge.

    See "Packaging and Deploying Orchestration Cartridges" for more information.

Adding a New Fulfillment Mode

This topic describes design steps to add new fulfillment mode.

To add new fulfillment mode:

1. Import all of the component cartridges to be included in the solution into Design Studio.

   For example, this might include a sealed base cartridge, Billing cartridge, Provisioning cartridge, Fulfillment Pattern cartridge, and Product Mapping cartridge.
2. Create a new Order and Service Management project to host the new fulfillment function.

   See "Creating New Cartridge Projects" for more information.
3. In the Order and Service Management Project editor **Properties** tab, deselect the **Standalone** check box.

   This allows the cartridge to be referenced in the composite cartridge as part of the solution (rather than as a standalone cartridge with no dependencies).
4. Add a new fulfillment mode.

   See "Creating New Fulfillment Modes" for more information.

5. Create a new recognition rule that recognizes the new fulfillment mode inside either the new cartridge or an unsealed existing cartridge.

   See "Creating New Recognition Rules" for more information.

6. Add the new fulfillment mode to the base orchestration fulfillment pattern or to any applicable orchestration fulfillment patterns.

   See "Orchestration Fulfillment Pattern Editor Details Tab" for more information.

7. Model the orchestration plan for the new fulfillment mode for all of the affected orchestration fulfillment patterns.

   See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for more information.

8. If you do not already have a composite cartridge to use, create a new Order and Service Management Composite project.

   See "Creating New Cartridge Projects" for more information.

9. Add the necessary cartridges, including the cartridge you created for the new fulfillment function, to the Dependency tab for the composite cartridge.

   See "Project Editor Dependency Tab" for more information.

10. Ensure that the appropriate orders, fulfillment patterns, and recognition rules are included in the manifest for the solution cartridge.

    See "Composite Cartridge Editor Manifest Tab" for more information.

11. Package and deploy the composite cartridge.

    See "Packaging and Deploying Orchestration Cartridges" for more information.

Adding a New Product

This topic describes design steps to add a new product.

To add a new product:

1. Import all of the component cartridges to be included in the solution into Design Studio.

   For example, this might include a sealed base cartridge, Billing cartridge, Provisioning cartridge, Fulfillment Pattern cartridge, and Product Mapping cartridge.

2. Create or select an Order and Service Management project to host any new OSM entities.

   See "Creating New Cartridge Projects" for more information.

3. In the Order and Service Management Project editor **Properties** tab, ensure that the **Standalone** check box is not selected.

   This allows the cartridge to be referenced in the composite cartridge as part of the solution (rather than as a standalone cartridge with no dependencies).

4. Create or select a common model project to contain common model entities.

   See "Creating New Cartridge Projects" for more information.

5. Ensure that an appropriate container relationship is defined for the common model project you are going to use.

   See "Project Editor Properties Tab" for more information.

6. Create or import the new product into the common model project.

   See "Importing Products" or "Product Editor" for more information.

7. (Optional) Associate the new product with a domain in the conceptual model.

See "Domain Editor" for more information.

8. (Optional) Create an order item parameter binding for the new product if you are planning to use the distributed order template or order transformation manager.

   See "Working with Order Item Parameter Bindings" for more information.

9. Map the new product to an existing fulfillment pattern or create a new fulfillment pattern and map it to the new product in the common model project.

   See "Product Editor Properties Tab" for more information.

10. If you created a new fulfillment pattern in the common model project, create a new OSM orchestration fulfillment pattern and set it to realize the common model fulfillment pattern you created.

    See "Creating New Orchestration Fulfillment Patterns" and "Orchestration Fulfillment Pattern Editor Realization Tab" for more information.

11. If you added a new orchestration fulfillment pattern, model the orchestration for the new orchestration fulfillment pattern.

    See "Orchestration Fulfillment Pattern Editor" and "Decomposition Rule Editor Details Tab" for more information.

12. If you do not already have a composite cartridge to use, create a new Order and Service Management Composite project.

    See "Creating New Cartridge Projects" for more information.

13. Add the necessary cartridges, including any OSM cartridge you created, to the Dependency tab for the composite cartridge.

    See "Project Editor Dependency Tab" for more information.

14. Ensure that the appropriate orders, fulfillment patterns, and recognition rules are included in the manifest for the solution cartridge.

    See "Composite Cartridge Editor Manifest Tab" for more information.

15. Package and deploy the composite cartridge.

    See "Packaging and Deploying Orchestration Cartridges" for more information.

Adding a New Service

Describes design steps to implement triple-play solution (broadband, VoIP, TV).

1. Import all of the component cartridges to be included in the solution into Design Studio.

   For example, this might include a sealed base cartridge, Billing cartridge, Provisioning cartridge, Fulfillment Pattern cartridge, and Product Mapping cartridge.

2. Create or select an Order and Service Management project to host any new OSM entities.

   See "Creating New Cartridge Projects" for more information.

3. In the Order and Service Management Project editor **Properties** tab, ensure that the **Standalone** check box is not selected.

   This allows the cartridge to be referenced in the composite cartridge as part of the solution (rather than as a standalone cartridge with no dependencies).

4. Create or select a common model project to contain common model entities.

   See "Creating New Cartridge Projects" for more information.

5. Ensure that an appropriate container relationship is defined for the common model project you are going to use.

   See "Project Editor Properties Tab" for more information.

6. Add a new orchestration fulfillment pattern that extends from the base orchestration fulfillment pattern.

   See "Creating New Orchestration Fulfillment Patterns" for more information.

7. Model the orchestration plan for the new orchestration fulfillment pattern.

   See "Orchestration Fulfillment Pattern Editor Orchestration Plan Tab" for more information.

8. If the new service requires communicating with one or more new external systems, create order component specifications for the new systems.

   See "Creating New Order Component Specifications" for more information.

9. If you have created new order component specifications, ensure the **Order Component Executable** check box is deselected.

10. Create or import the new product into the common model project.

    See "Importing Products" or "Product Editor" for more information.

11. (Optional) Associate the new product with a domain in the conceptual model.

    See "Domain Editor" for more information.

12. (Optional) Create an order item parameter binding for the new product if you are planning to use the distributed order template or order transformation manager.

    See "Working with Order Item Parameter Bindings" for more information.

13. Map the new product to the appropriate fulfillment pattern in the common model project.

    See "Product Editor Properties Tab" for more information.

14. Add decomposition rules for the new service.

    See "Creating New Decomposition Rules" for more information.

15. If you do not already have a composite cartridge to use, create a new Order and Service Management Composite project.

    See "Creating New Cartridge Projects" for more information.

16. Add the necessary cartridges, including any OSM cartridge you created, to the Dependency tab for the composite cartridge.

    See "Project Editor Dependency Tab" for more information.

17. Ensure that the appropriate orders, fulfillment patterns, and recognition rules are included in the manifest for the solution cartridge.

    See "Composite Cartridge Editor Manifest Tab" for more information.

18. Package and deploy the composite cartridge.

    See "Packaging and Deploying Orchestration Cartridges" for more information.

22

Working with Order Item Parameter Bindings

Order item parameter bindings are used in many areas of Oracle Communications Order and Service Management (OSM) to ensure that order item parameters can be synchronized with conceptual model entities. Order item parameter bindings are especially necessary when using the order transformation manager feature of OSM.

You use order item parameter bindings to map incoming order item parameters to parameters on a conceptual model entity. You can create multiple order item parameter bindings for different types of parameters on the incoming order. For example, you could create one binding for parameters that are stored as name-value pairs and another binding for more strongly typed parameters, where the element name is itself the name of the parameter.

For more information about conceptual model entities, see "Working with Conceptual Models". For more information about the order transformation manager feature, see *OSM Modeling Guide*.

When working with order item parameter bindings, see the following topics:

- Creating New Order Item Parameter Bindings
- Order Item Parameter Binding Editor

Creating New Order Item Parameter Bindings

Order item parameter bindings define how OSM maps incoming order item parameters to parameters on a conceptual model entity.

You can create order item parameter bindings manually by using the procedure below, or you can use the supplied **Order Item Parameter Binding** design pattern. This design patterns is located in the Model Project section of the list of design patterns. If you launch the design pattern from the context menu of an appropriate conceptual model entity, the name of that entity will be populated in the design pattern. Otherwise you will have the opportunity to select the name of the conceptual model entity to use. The design pattern assists you in your configuration activities by creating bindings with sample XQuery files. It also provides a cheat sheet with information about next steps to take. For more information about the capabilities of the design pattern, run the design pattern.

To create new order item parameter bindings manually:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, then select **Order Transformation**, and then select **Order Item Parameter Binding**.

   The Order Item Parameter Binding wizard is displayed.

2. In the **Project Name** field, select the project in which to save the order item parameter binding.

3. In the **Name** field, enter a name for the order item parameter binding.

   The name must be unique among order item parameter binding entity types within the same namespace.

4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the order item parameter binding.

5. (Optional) Select a location for the order item parameter binding.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.

6. Click **Finish**.

   Design Studio adds the new order item parameter binding to the selected project and opens the new entity in the Order Item Parameter Binding editor.

7. In the **Properties** tab, select the order item and conceptual model entity to which the binding will apply.

8. Use the **Parameter Bindings** tab to define the specific binding information.

Related Topics

Order Item Parameter Binding Editor

Order Item Parameter Binding Editor

Use the Order Item Parameter Binding editor to map order item parameters to parameters on a conceptual model entity.

When modeling order item parameter bindings, see the following topics:

- Order Item Parameter Binding Editor Properties Tab
- Order Item Parameter Binding Editor Parameter Bindings Tab

The following fields are common to multiple tabs in the Order Item Parameter Binding editor:

Field	Use
Description	Edit the display name of the order item parameter binding.
Namespace	Select an existing namespace or enter a unique namespace in which to include the order item parameter binding.

Related Topics

Creating New Order Item Parameter Bindings

Order Item Parameter Binding Editor Properties Tab

Use the **Properties** tab to define the entities to use for the parameter bindings.

> **Note:**
>
> See "Order Item Parameter Binding Editor" for information about fields that appear on multiple Order Item Parameter Binding editor tabs.

Field	Use
Order Item Specification	Click **Select** to select an order item from the list of all of the order items in the workspace. The order items in this list are the order items defined in the same project as the order item parameter binding and the order items in any project on which this project depends. See "Managing Project Dependencies" for more information about project dependencies.
Conceptual Model Entity	Click **Select** to select a conceptual model entity that contains the parameters to which order line items will be mapped.

Related Topics

Order Item Parameter Binding Editor

Order Item Parameter Binding Editor Parameter Bindings Tab

Use the **Parameter Bindings** tab to define expressions that will map the incoming order item parameters to parameters defined for the conceptual model entity selected on the **Properties** tab.

> **Note:**
>
> See "Order Item Parameter Binding Editor" for information about fields that appear on multiple Order Item Parameter Binding editor tabs.

Field	Use
Bindings	Displays a list of the bindings contained in the current order item parameter binding. Do any of the following: • Click **Add** to add a new binding. • Select an existing binding to edit the binding using the fields in the **Binding Details** area. • Select an existing binding and click **Remove** to remove the binding.
Binding	Enter the name of the binding. You may want to have the name reflect the scope of the binding. The binding can map all of the parameters for the product, or you can define multiple bindings, each with its own XQuery.
Binding Expression	Enter the XQuery expression to perform the parameter binding. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "Order Item Parameter Binding XQuery Expressions" for more information about this XQuery field.

Related Topics

Order Item Parameter Binding Editor

23

Working with Transformation Sequences

Transformation sequences are part of the order transformation manager feature of Oracle Communications Order and Service Management (OSM). Transformation sequences enable you to define the transformation stages. Transformation stages define the source (input) and target (output) order items and the relationship between them for each step of the transformation. For more information about the order transformation manager feature, see *OSM Modeling Guide*.

When working with transformation sequences, see the following topics:

Creating New Transformation Sequences

Transformation sequences enable you to define transformation stages, which define the source and target order items and the relationship between them for each step of the transformation.

To create new transformation sequences:

1. From the **Studio** menu, select **New**, then select **Order and Service Management**, then select **Order Transformation**, and then select **Transformation Sequence**.

   The Transformation Sequence wizard is displayed.
2. In the **Project** field, select the project in which to save the transformation sequence.
3. In the **Name** field, enter a name for the transformation sequence.

   The name must be unique among transformation sequence entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the transformation sequence.
5. (Optional) Select a location for the transformation sequence.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

   Design Studio adds the new transformation sequence to the selected project and opens the new entity in the Transformation Sequence editor.
7. In the **Properties** tab of the Transformation Sequence editor, select input and output order items for the transformation sequence.

8. Use the **Dependencies** tab to configure the transformation stages in the transformation sequence.

Related Topics

Transformation Sequence Editor

Configuring Non-Sequential Stage Transitions

By default, the transformation stages are run in sequence, beginning with the first stage in the list of transformation stages in the **Dependencies** tab. However, you can configure multiple stages to run in parallel.

> **Note:**
>
> Because the first transformation stage creates the transformed order items (rather than enriching the data), the first transformation stage in the sequence cannot run in parallel with another stage.

To configure non-sequential stage transitions:

1. Open the editor for the transformation sequence you would like to modify and click the **Dependencies** tab.
2. Right-click the transformation sequence and select **Change to Non-Sequential**.

   Default transitions are added in the tree view. Each stage (except the last) now has a default transition to the stage following it. If you do not do any more configuration, the stages will run sequentially as before.
3. If you would like to remove a transition, expand the tree view for the stages, right-click the name of any transition that you do not need, and select **Remove**.
4. If you would like to add a transition, right-click the name of a transformation stage, select **Add Transition**, and select the name of the transformation stage that you would like to run immediately after the transformation stage you right-clicked.

Configuring Non-Sequential Stage Transitions: an Example

Assume that you have a transformation sequence called TransSeq with four stages. If you would like the transformation sequence to start with Stage1, continue to Stage2, and then run Stage3a and Stage3b in parallel, configure your transformation sequence in the following order:

- TransSeq
  - Stage1
  - Stage2
  - Stage3a
  - Stage3b

To configure the last two stages to run in parallel, right-click **TransSeq** and select **Change to Non-Sequential**. Right-click the default transition defined for Stage3a, which is named **Stage3a => Stage3b**, and select **Remove**. Then right-click **Stage2**, select **Add Transition**, and select **Stage3b**. Stage3a will run immediately after Stage2 because of the default

transition that was added for Stage2, and Stage3b will now run in parallel with it because of the transition you have just added from Stage2 to Stage3b.

Related Topics

Transformation Sequence Editor

Transformation Sequence Editor

Use the Transformation Sequence editor to define the source and target order items and the relationship between them for each step of the transformation.

When modeling transformation sequences, see the following topics:

- Transformation Sequence Editor Properties Tab
- Transformation Sequence Editor Dependencies Tab

The following fields are common to multiple tabs in the Transformation Sequence editor:

Field	Use
Description	Edit the display name of the transformation sequence.
Namespace	Select an existing namespace or enter a unique namespace in which to include the transformation sequence.

Related Topics

Creating New Transformation Sequences

Transformation Sequence Editor Properties Tab

Use the **Properties** tab to define the order items that are used for the transformation sequence.

> **Note:**
>
> See "Transformation Sequence Editor" for information about fields that appear on multiple Transformation Sequence editor tabs.

Field	Use
Output Order Item	Click **Select** to select an order item specification for the order items to result from the transformation.
Input Order Item	Click **Select** to select an order item specification for the order items to be transformed.

Related Topics

Transformation Sequence Editor

Transformation Sequence Editor Dependencies Tab

Use the **Dependencies** tab to configure the transformation stages in the transformation sequence.

When modeling transformation stages, refer to the following topics:

- Transformation Sequence Editor Dependencies Tab Transformation Model Subtab
- Transformation Sequence Editor Dependencies Tab Stage Detail Subtab
- Transformation Sequence Editor Dependencies Order Item Context Subtab
- Transformation Sequence Editor Dependencies Tab Related Order Items Selector Subtab
- Transformation Sequence Editor Dependencies Tab Relationship Type Details Subtab
- Transformation Sequence Editor Dependencies Tab Stage Condition Subtab

> **Note:**
>
> See "Transformation Sequence Editor" for information about fields that appear on multiple Transformation Sequence editor tabs.

Field	Use
Tree area	The tree area contains the transformation sequence and each of the transformation stages. Do any of the following: • Right-click the name of the transformation sequence and select **Add New Transformation Stage** to add a new stage. • Select a transformation stage from this list to configure it using the fields in the subtabs. • Right-click the name of a transformation stage and select **Move Up** to move the transformation stage earlier in the sequence. • Right-click the name of a transformation stage and select **Move Down** to move the transformation stage later in the sequence. • Right-click the name of the transformation sequence or stage and select **Change to Non-Sequential** to allow custom sequencing of the transformation stages for the sequence. See "Configuring Non-Sequential Stage Transitions" for more information about using this option. • Right-click the name of a transformation stage and select Add Transition to configure a non-sequential stage transition. This option is only available if the transformation sequence is using non-sequential stage transitions. See "Configuring Non-Sequential Stage Transitions" for more information about using this option. • Right-click the name of a transformation stage and select **Remove** to delete the stage.

Related Topics

Transformation Sequence Editor

Transformation Sequence Editor Dependencies Tab Transformation Model Subtab

Use the **Transformation Sequence** subtab of the **Dependencies** tab to define the name of the transformation sequence and to determine whether the stages run in parallel or sequentially. This subtab is visible when you have a transformation sequence selected in the tree area.

Field	Use
Name	Enter the name of the transformation sequence.
Sequential Execution	Select this option to allow custom sequencing of the transformation stages. See "Configuring Non-Sequential Stage Transitions" for more information about using this option.

Related Topics

Transformation Sequence Editor Dependencies Tab

Transformation Sequence Editor Dependencies Tab Stage Detail Subtab

Use the **Stage Detail** subtab of the **Dependencies** tab to configure details of the transformation stage. This subtab is visible when you have a transformation stage selected in the tree area.

Field	Use
Name	Enter the name of the transformation stage.
Allow Overrides	Select this option to set a value that indicates that the results of previous transformation stages may be overridden. **Note:** This option should not be selected for the first transformation stage in the transformation sequence.

Related Topics

Transformation Sequence Editor Dependencies Tab

Transformation Sequence Editor Dependencies Order Item Context Subtab

This subtab is visible when you have a transformation stage selected in the tree area. Use the **Order Item Context** subtab of the **Dependencies** tab to configure details of the transformation stage.

Field	Use
Reuse Previous Stage Context	Select this option to indicate that the current transformation stage should not recalculate the order item context but should use the order item context of the previous stage. **Note:** This option should not be selected for the first transformation stage in the transformation sequence.

Field	Use
Order Item Context	Do one of the following: • Select **Simple** to configure the order item context using a simple user process. In the **Property** field, select an order item property belonging to the order item specification selected on the **Properties** tab. Then enter a value in the **Value** field to determine what property is used to determine whether an order item is a context order item. The order item is considered a context order item if the value of the specified order item property is equal to the value in the **Value** field. • Select **Advanced** to configure the order item context using an XQuery expression. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Order Item Context XQuery Expressions" for more information about this Query field.

Related Topics

Transformation Sequence Editor Dependencies Tab

Transformation Sequence Editor Dependencies Tab Related Order Items Selector Subtab

This subtab is visible when you have a transformation stage selected in the tree area. Use the **Related Order Items Selector** subtab of the **Dependencies** tab to configure details of the order items to which the transformation stage applies. This subtab is visible when you have a transformation stage selected in the tree area.

Field	Use
Related Order Item	Do one of the following: • Select **None** to not configure any related order items. This is recommended for the first transformation stage. • Select **Simple** to configure the related order item using a simple user process. In the **Direction** field, select a way to search for related order items. The possible values are Ancestors, Siblings, and Dependents. In the **Hierarchy** field select the order item hierarchy that to indicate the hierarchy that is used to determine the related order items. • Select **Advanced** to configure the related order item using an XQuery expression. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Related Order Item Selector XQuery Expressions" for more information about this XQuery field.

Related Topics

Transformation Sequence Editor Dependencies Tab

Transformation Sequence Editor Dependencies Tab Relationship Type Details Subtab

This subtab is visible when you have a transformation stage selected in the tree area. Use the **Relationship Type Details** subtab of the **Dependencies** tab to configure details of the Java classes that implement the transformation stage. This subtab is visible when you have a transformation stage selected in the tree area.

Field	Use
Relationship Types	This table shows a list of the relationships that are available to the transformation stage. Do any of the following: • Click **Add** to add a new relationship type to the table. A dialog box is presented where you can enter the information for the relationship using the **Relationship Type** and **Implementation** fields, or you can click **OK** in the dialog box without adding anything and enter the information using the fields in the editor. • Click **Remove** to remove a relationship from the table. • Click an existing row in the table to edit the information for that row using the **Relationship Type** and **Implementation** fields.
Relationship Type	Click **Select** to select a relationship type to be implemented by the transformation stage.
Implementation	Do one of the following: • Click **Select** to select a Java class from the classes in the workspace that implement the required relationship handler interfaces for the selected relationship type for this transformation stage. • If the **Implementation** field is empty, click the name of the field to display the New Java Class wizard, with the appropriate interface for the relationship implementation prepopulated.

Related Topics

Transformation Sequence Editor Dependencies Tab

Transformation Sequence Editor Dependencies Tab Stage Condition Subtab

Use the **Stage Condition** subtab of the **Dependencies** tab to configure how OSM determines whether the stage should be run. This subtab is visible when you have a transformation stage selected in the tree area.

Field	Use
XQuery	The **XQuery** tab enables you to configure XQuery-based rules to determine whether the transformation stage will run. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Stage Condition XQuery Expressions" for more information about this XQuery field.

Field	Use
Instances	Define a Data Instance behavior to obtain data that is not included in the order data and to make that data available to the XQuery expression. Click **Add** to add a Data Instance behavior. Select the Data Instance behavior, and click **Properties** to configure the Data Instance behavior. See "Defining Data Instance Behavior Properties" for more information.
Information	Describe the intended use of the XQuery expression. For example, you might describe the functionality of a complex expression or provide instructions on its use.

Related Topics

Transformation Sequence Editor Dependencies Tab

24

Working with Transformation Managers

Transformation managers are part of the order transformation manager feature of Oracle Communications Order and Service Management (OSM). You use transformation managers to select the transformation sequences for the service domains within a provider function. For more information about the order transformation manager feature, see *OSM Modeling Guide*.

When working with transformation managers, see the following topics:

- Creating New Transformation Managers
- Transformation Manager Editor

Creating New Transformation Managers

You create transformation managers to bind transformation sequences with each domain associated with a specified provider function.

To create new transformation managers:

1. From the **Studio** menu, select **New**, then select **Order and Service Management**, then select **Order Transformation**, and then select **Transformation Manager**.

   The Transformation Manager wizard is displayed.
2. In the **Project Name** field, select the project in which to save the transformation manager.
3. In the **Name** field, enter a name for the transformation manager.

   The name must be unique among transformation manager entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the transformation manager.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the transformation manager.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field, or select a location different from the system-provided default. To select a different location:

   a. Click the Folder field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

   Design Studio adds the new transformation manager to the selected project and opens the new entity in the Transformation Manager editor.

Related Topics

Transformation Manager Editor

About Domains

About Provider Functions

Transformation Manager Editor

Use the Transformation Manager editor to define the transformation sequence for each service domain in a provider function.

Field	Use
Description	Edit the display name of the transformation manager.
Namespace	Select an existing namespace or enter a unique namespace in which to include the transformation manager.
Provider Function	Click **Select** to select the provider function that this transformation manager realizes.
Transformation Sequence Bindings area	Displays the transformation sequence used for each domain associated with provider function. Do any of the following: • Click **Add** to add domain and transformation sequence bindings. • Select a row in the table to change the values for the binding using the **Service Domain** and **Transformation Sequence** fields. • Select a row and click **Remove** to remove the selected transformation sequence binding.
Domain	Click **Select** to select a domain to bind to a transformation sequence.
Transformation Sequence	Click **Select** to select a transformation sequence to bind to the service domain.

Related Topics

Creating New Transformation Managers

25

Working with Mapping Rules

Mapping rules are part of the order transformation manager feature of Oracle Communications Order and Service Management (OSM). Mapping rules enable you to map original order items to transformed order items and ensure that the correct data elements from the customer order are added to the transformed order items. For more information about the order transformation manager feature, see *OSM Modeling Guide*.

When working with mapping rules, see the following topics:

Creating New Mapping Rules

Mapping rules define the entity and data element mappings between two entities that have a named relationship.

To create new mapping rules:

1. From the **Studio** menu, select **New**, then select **Order and Service Management**, then select **Order Transformation**, and then select **Mapping Rule**.

   The Mapping Rule wizard is displayed.
2. In the **Project Name** field, select the project in which to save the mapping rule.
3. In the **Name** field, enter a name for the mapping rule.

   The name must be unique among mapping rule entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the mapping rule.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the mapping rule.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

Design Studio adds the new Mapping Rule entity to the selected project and opens the new entity in the Mapping Rule editor.

7. Configure the properties for the mapping in the **Properties** tab.

Related Topics

Mapping Rule Editor

Creating New Mappings in a Mapping Rule

You create new mappings in a mapping rule to define the correct transformed order items and their data elements. There are two methods of creating new mappings. You can define a new mapping between any two parameters manually. In addition, if the names of the source and target data elements are the same, you can create new mappings by inference.

> **Note:**
>
> If you update an order either to add a parameter (which includes providing a value to a parameter that did not previously have one) or to delete a parameter (which includes setting the value of a parameter to null), the OSM order transformation manager will not propagate the change in either the forward or reverse direction. To avoid the need to add or delete values, you can set values (such as NOT_SET) as defaults on the conceptual model entity or in the order item parameter binding and ensure that a parameter always has a value.

To create a new mapping manually:

1. Open a mapping rule and click the **Mapping** tab.
2. In the Named Relationships area, click **Add**.

   The Create Named Relationships Element dialog box is displayed.
3. Select the relationship you would like to use and click **OK**.

   The list of relationships displayed is determined by the provider function and domain that you selected in the **Properties** tab when creating the mapping rule.
4. In the Named Relationships area, select a relationship.

   After you have selected a relationship, in the Named Relationships area, in the **Mapping** subtab, the source entity of the relationship and its data elements are displayed on the left. The target entity and its data elements are displayed on the right.
5. Do any of the following:
   - To map an entity to an entity, drag the entity on the left to the entity on the right.
   - To map an entity to a data element, expand the data element tree for the target entity and drag the source entity to the appropriate data element on the target entity.
   - To map a data element to another data element, expand the data element trees for both the source and target entities and drag the source data element to the target data element.

To create a single new mapping by inference:

1. Open a mapping rule and click the **Mapping** tab.

2. Right-click the target of the desired mapping and select **Infer Mapping for Selection**.

   If a data element with the same name (case-sensitive) exists on the source entity, a mapping is created between the two data elements and a dialog box, indicating that a mapping was created, is displayed. The type of mapping created depends on the data types of the source and target data elements. See "Configuring a Data-Element-to-Data-Element Mapping" for information about the default types of mappings between data elements.

   If no appropriate data element on the source element is found, a dialog box, indicating that no mapping was created, is displayed.

3. Click **OK** in the dialog box.

To create multiple new mappings by inference:

1. Open a mapping rule and click the **Mapping** tab.
2. Right-click any element of the target entity and select **Infer All Mappings**.

   If any data elements with the same name (case-sensitive) exist on both the source and target entities, mappings are created between the data elements, and a dialog box, indicating how many mappings were created, is displayed. The type of mappings created depends on the data types of the source and target data elements. See "Configuring a Data-Element-to-Data-Element Mapping" for information about the default types of mappings between data elements.

   If no appropriate data elements were found, a dialog box is displayed indicating that no mapping was created.

3. Click **OK** in the dialog box.

Related Topics

Mapping Rule Editor

Configuring Mappings

Configure mappings you have created to determine how data values are transferred from the source to the target of the mapping. Different types of mappings are available depending on whether the source and target are entities or data elements. The available mapping types are:

- Copy value (data-element-to-data-element mappings): This type of mapping copies the value of the source parameter directly to the target parameter.
- Value transformation (entity-to-data-element mappings and data-element-to-data-element mappings): This type of mapping enables you to set up a list of value mappings, where a source value is mapped to a target value.
- Unit of measure (data-element-to-data-element mappings): This type of mapping enables you to perform conversions between numeric values having different units of measure. The units of measure must be defined on the data elements for this type of mapping to work. For more information about setting up units of measure, see "Conceptual Model Unit of Measure Editor". For more information about setting units of measure for individual data elements, see "Details or Attributes Tab".
- Advanced (all mapping types): This type of mapping enables you to define the mapping using XQuery.

In addition to the procedures below, you can change the type of a mapping from one to another and perform other actions relating to the mapping by right-clicking the target of the mapping and selecting an appropriate item from the context menu.

When configuring mappings, see the following topics:

- Configuring an Entity-to-Entity Mapping
- Configuring an Entity-to-Data-Element Mapping
- Configuring a Data-Element-to-Data-Element Mapping

Related Topics

Configuring Bi-Directional Mapping

Configuring Multi-Instance Structure Key Mapping

Mapping Rule Editor

Configuring an Entity-to-Entity Mapping

The types of mapping between entities are:

- Advanced mapping

To configure an entity-to-entity advanced mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the named relationship for the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.

   In the Mapping Rule Item area, **Advanced** is selected.
4. In the Mapping Rule Item area, enter information about the XQuery expression you would like to use. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Entity-to-Entity Advanced Mapping XQuery Expressions" for more information about this XQuery field.
5. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

Related Topics

Configuring Mappings

Configuring an Entity-to-Data-Element Mapping

The types of mapping between entities and data elements are:

- Value transformation mapping
- Advanced mapping

To configure an entity-to-data-element value transformation mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Value Map**.
5. In the **Configuration** subtab, Transform Value Selection area, click **Add**.
6. In the **Output Value** field, enter a value and click **OK**.

7. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

To configure an entity-to-data-element advanced mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Advanced**.
5. In the Mapping Rule Item area, enter information about the XQuery expression you would like to use. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Entity-to-Entity Advanced Mapping XQuery Expressions" for more information about this XQuery field.
6. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

Related Topics

Configuring Mappings

Configuring a Data-Element-to-Data-Element Mapping

The types of mapping between data elements are:

- Copy value mapping
- Value transformation mapping
- Unit-of-measure mapping
- Advanced mapping

When you create a mapping between data elements, the default mapping type created depends on the data elements being mapped:

- If the source and target elements are of the same type (except if one or both has units of measure defined for it) or if the target element is of string type, the default mapping is copy value mapping.
- If either the source or target data element has enumerations defined for it, the default mapping type is value transformation mapping.
- If both the source and target have units defined from the same Units of Measure common model entity, the default mapping is unit-of-measure mapping.
- If the source and target elements are of incompatible types (for example only one has units of measure defined or the source is dateTime and the target is Boolean), the default mapping type is an advanced mapping.

To configure a data-element-to-data-element copy value mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Copy Value**.
5. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

To configure a data-element-to-data-element value transformation mapping:

1. Open a mapping rule and click the **Mapping** tab.

2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Value Map**.
5. For each incoming value that you wish to transform, do the following:
    a. In the **Configuration** subtab, Transform Value Selection area, click **Add**.
    b. Enter values in the **Input Value** and **Output Value** fields and click **OK**.
6. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

To configure a data-element-to-data-element unit-of-measure mapping:

In general, there is no need to configure unit-of-measure mappings. If you create a mapping between two data elements with compatible units of measure defined for them, when you create a mapping between them, it is created as a unit-of-measure mapping by default. If you create a mapping between any other types of data elements, the unit-of-measure mapping is not available. However, if you have a mapping between compatible data elements that you have changed to another type of mapping, and you want to reconfigure it as a unit-of-measure mapping, you can use the following procedure.

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Unit Of Measure**.

    The units defined for the data elements are automatically populated in the **Target Unit** and **Source Unit** fields and cannot be changed.

To configure a data-element-to-data-element advanced mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, click the target of the mapping you would like to configure.
4. In the Mapping Rule Item area, select **Advanced**.
5. In the Mapping Rule Item area, enter information about the XQuery expression you would like to use. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Entity-to-Entity Advanced Mapping XQuery Expressions" for more information about this XQuery field.
6. Configure the information in the **Details**, **Condition**, and **Actions** subtabs as necessary.

Related Topics

Configuring Mappings

Configuring Bi-Directional Mapping

Bi-directional mapping enables you to propagate data from the transformed order item back to the original order item. This function is supported only for parameter data elements, not for order item properties.

> **Note:**
>
> If you update an order to either add a parameter (which includes providing a value to a parameter that did not previously have one) or delete a parameter (which includes setting the value of a parameter to null), the OSM order transformation manager will not propagate the change in either the forward or reverse direction. To avoid the need to add or delete values, you can set values (such as NOT_SET) as defaults on the conceptual model entity or in the order item parameter binding and ensure that a parameter always has a value.

To configure bi-directional mapping:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select a mapping.
3. In the **Details** subtab, select **Supports Bi-directional Mapping**.
4. In the **Mapping** subtab, select the target of any data-element-to-data-element advanced-type mapping between parameters.

   The **Bi-Directional Mapping** subtab is now displayed in the Mapping Rule Item area.
5. In the **XQuery** field in the **Bi-Directional Mapping** subtab, provide XQuery information for performing the data mapping from the transformed order item to the original order item. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Reverse Mapping XQuery Expressions" for more information about this XQuery field.

Related Topics

Configuring Mappings

Mapping Rule Editor

Configuring Multi-Instance Structure Key Mapping

Multi-instance data mapping is available when the target of a data-element-to-data-element mapping is a multi-instance structure. The key uniquely identifies a specific instance of the structure. Key values must be unique in an order item.

To configure the key for a multi-instance structure:

1. Open a mapping rule and click the **Mapping** tab.
2. In the **Entries** table, select the mapping you would like to configure.
3. In the **Mapping** subtab, select a mapping target that is part of a multi-instance structure.

   The **Multi-Instance** and **Multi-Instance Expression** subtabs are displayed in the Mapping Rule Item area.
4. Do one of the following:
   - To use an order item property as the key, in the **Multi-Instance** subtab, in the **Source Property Key** field, select an order item property to use as the key element.
   - To use the element defined as the structure key in the data schema, in the **Multi-Instance** subtab, select the **Use Default Key** option.

- Do select the key using an XQuery expression, in the **Multi-Instance Expression** subtab, provide XQuery information to determine the key. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Multi-Instance XQuery Expressions" for more information about this XQuery field.

Related Topics

Configuring Mappings

Mapping Rule Editor

Mapping Rule Editor

Use the Mapping Rule editor to map entities and data elements for transformed order items.

When modeling mapping rules, see the following topics:

- Mapping Rule Editor Properties Tab
- Mapping Rule Editor Mapping Tab

The following fields are common to multiple tabs in the Mapping Rule editor:

Field	Use
Description	Edit the display name of the mapping rule.
Namespace	Select an existing namespace or enter a unique namespace in which to include the mapping rule.

Related Topics

Creating New Mapping Rules

Creating New Mappings in a Mapping Rule

Configuring Mappings

Mapping Rule Editor Properties Tab

Use the **Properties** tab to define the properties for the mapping rule.

> **Note:**
>
> See "Mapping Rule Editor" for information about fields that appear on multiple Mapping Rule editor tabs.

Field	Use
Provider Function	Click **Select** to select a provider function from a list of the provider functions defined in your workspace. The provider function that you select here is used in determining the list of relationships that are available to add on the **Mapping** tab. See "About Provider Functions" for more information about defining provider functions.

Field	Use
Domain	Click **Select** to select a domain from a list of the domains defined in your workspace. The domain that you select here is used in determining the list of relationships that are available to add on the **Mapping** tab. See "About Domains" for more information about defining domains.
Output Order Item	Click **Select** to select the order item specification that you wish to use for the result of the transformation. All order item specifications in the workspace are available for selection.
Input Order Item	Click **Select** to select the order item specification that you wish to use for the source of the transformation. All order item specifications in the workspace are available for selection.

Related Topics

Mapping Rule Editor

Mapping Rule Editor Mapping Tab

Use the **Mapping** tab to configure the mapping of entities and data elements from source to target order items. For more information about mapping types, see "Configuring Mappings".

When modeling mappings in mapping rules, see the following topics:

- Mapping Rule Editor
- Mapping Rule Editor Mapping Tab Details Subtab
- Mapping Rule Editor Mapping Tab Condition Subtab
- Mapping Rule Editor Mapping Tab Fulfillment State Filters Subtab
- Mapping Rule Editor Mapping Tab Actions Subtab

> **Note:**
>
> See "Mapping Rule Editor" for information about fields that appear on multiple Mapping Rule editor tabs.

Field	Use
Filters area	Use the **Input**, **Output**, and **Relationship Type** fields to filter the mappings displayed in the **Entries** table.
Entries table	The Entries table displays the relationships that have been configured for the mapping rule. Do any of the following: • Click **Add** to add a mapping to the table. • Select a row in the table to configure the mapping. See "Configuring Mappings" for more information. • Select a row and click **Remove** to remove the selected mapping.

Related Topics

Mapping Rule Editor

Mapping Rule Editor Mapping Tab Mapping Subtab

Use the **Mapping** subtab to map entities and data elements to the appropriate values on the transformed order item.

Field	Use
Input	This tree contains the properties of the original order item and the parameters on the source entity.
Output	This tree contains the properties of the transformed order item and the parameters on the target entity.
Mapping Rule Item area	This area contains the specifics for the individual mappings. For more information about the contents of this section for the different mapping types, see "Configuring Mappings".

Related Topics

Mapping Rule Editor Mapping Tab

Mapping Rule Editor Mapping Tab Details Subtab

Use the **Details** subtab to configure details about the selected named relationship.

Field	Use
Use for Order Item Selection	Select this option to use this relationship to select the order items for the transformation.
Supports Bi-directional Mapping	Select this option to propagate data that is changed on the transformed order item during order processing back to the original order item.
Named Relationship	This field contains the name of the relationship between the input and output entities. See "Product Editor Derivation Tab" for more information about creating this relationship in a conceptual model product. See "Conceptual Model Editor Components Tab" for information about creating this relationship for other types of conceptual model entities.
Input Conceptual Model Entity	Displays the name of the conceptual model entity that is related to the original order item.
Input Action	Select an action to use for the input entity. This action provides additional data elements for the source of the mapping.
Output Conceptual Model Entity	Displays the name of the conceptual model entity associated with the transformed order item.
Output Action	Select an action to use for the output entity. This action provides additional data elements for the target of the mapping.

Related Topics

Mapping Rule Editor Mapping Tab

Mapping Rule Editor Mapping Tab Condition Subtab

Use the **Condition** subtab to define a condition under which the mapping rule is used.

Field	Use
XQuery field	Enter an XQuery expression or a pointer for a condition for the mapping rule to be used. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Mapping Condition XQuery Expressions" for more information about this XQuery field.

Related Topics

Mapping Rule Editor Mapping Tab

Mapping Rule Editor Mapping Tab Fulfillment State Filters Subtab

Use the **Fulfillment State Filters** subtab to configure which external fulfillment states will be propagated for order items associated with the selected named relationship.

Field	Use
Fulfillment State Filters area	The table in this area contains the order component specifications whose external fulfillment states should be propagated. Fulfillment states for any order component that is not in this table will *not* be propagated. Do any of the following: • Click **Add** to add an order component to the table. In the resulting window, click **Select** to select an order component. If you do not see the order component in the resulting window, deselect the **Filter Project Dependencies** check box. By default, when you add a new order component to the table, all of its external fulfillment states are added. • Select a row in the table to configure the external fulfillment states for that order component. • Select a row and click **Remove** to remove the selected order component.
Fulfillment State Filter area	This area enables you to select specific external fulfillment states to propagate. External fulfillment states that are not included in this area will *not* be propagated. Select an order component in the Fulfillment State Filters area and do any of the following: • Click **Add** (and then deselect the **Filter Project Dependencies** check box if necessary) to select a new external fulfillment state to be propagated. • Click **Open** to open the referenced order component specification. • Select an external fulfillment state and click **Remove** to remove the selected external fulfillment state from the list of fulfillment states to be propagated for the selected named relationship.

Related Topics

Mapping Rule Editor Mapping Tab

Mapping Rule Editor Mapping Tab Actions Subtab

Use the **Actions** subtab to set action mappings for the selected named relationship. These mappings override any mappings that are set on the relationship itself.

Field	Use
Action Codes	Enter the action codes that apply to this mapping. Do any of the following: • Click **Select** and select an existing action code. • Click **Add** to create a new action code and use it for this mapping. • Select an action code and click **Open** to open the action code in the Action Code editor. • Select an action code and click **Remove** to remove it from the list.
Use Relationship Action Map	Select this option to inherit action mapping information from the named relationship. This is selected by default.
Action Mapping Type	Deselect **Use Relationship Action Map** to access these options. Do one of the following: • Select **Simple** to define the action mappings using the Action Maps and Action Map Details areas. • Select **Advanced** to define the action mappings using an XQuery field. Enter an XQuery to define the action mappings. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Action Mapping XQuery Expressions" for more information about this XQuery field.
Action Maps area	This area appears when you select the **Simple** action mapping type. The table in this area displays the action maps defined for this mapping rule. Do any of the following: • Click **Add** to add a new entry to the table. • Select an action map in the table and click **Remove** to remove that entry from the table.
Output Action Code area	This area appears when you select the **Simple** action mapping type. Select the desired action code to set the transformed order item.
Input Action Code area	This area appears when you select the **Simple** action mapping type. Select the action code for the original order item.
Current Action Code area	(Optional) This area appears when you select the **Simple** action mapping type. Use this field to produce a different output action code depending on the current action code of the transformed order item.

Related Topics

Mapping Rule Editor Mapping Tab

26

Working with Transformed Order Item Fulfillment State Composition Rule Sets

Oracle Communications Order and Service Management (OSM) uses transformed order item fulfillment state composition rule sets to determine composite fulfillment states for transformed order items. The rule sets contain rules and conditions that determine the overall fulfillment state of a transformed order item based on the composite fulfillment states of its child order items and the mapped fulfillment states of its order components. The rule sets also enable you to set fulfillment states based on the fulfillment states of the original order items that generated the transformed order item.

When working with transformed order item fulfillment state composition rule sets, see the following topics:

- About Transformed Order Item Fulfillment State Composition Rule Sets
- Creating New Transformed Order Item Fulfillment State Composition Rule Sets
- Transformed Order Item Fulfillment State Composition Rule Set Editor

About Transformed Order Item Fulfillment State Composition Rule Sets

Fulfillment state composition rules for transformed order items are defined in transformed order item composition rule sets. As with regular order items, you can have only one fulfillment state rule set per orchestration fulfillment pattern and order item specification combination.

A fulfillment state composition rule specifies the composite fulfillment state for the transformed order item when all of the conditions are met. If there are separate situations that can result in the same fulfillment state (for example, use this composite state if either any of the input states is completed, or if none of the states is failed), create additional rules that evaluate to the same composite fulfillment state.

The difference between these rule sets and the order item fulfillment state composition rule sets is that these rule sets enable you to account for the fact that a particular order item might have relationships with multiple transformed order items, all of which might affect the fulfillment state. Rules will calculate the state of an order item based on its associated transformed order items taking into consideration:

- The order item specification of the transformed order items
- The order component associated with the order item
- The type of relationship that exists between a source and transformed order item
- The conceptual model entity associated with the source order item.

Related Topics

Creating New Transformed Order Item Fulfillment State Composition Rule Sets

Transformed Order Item Fulfillment State Composition Rule Set Editor

Creating New Transformed Order Item Fulfillment State Composition Rule Sets

You create new transformed order item fulfillment state composition rule sets in Design Studio to define a composite fulfillment state based on the fulfillment states of lower-level order items.

To create new transformed order item fulfillment state composition rule sets:

1. From the **Studio** menu, select **New**, select **Order and Service Management**, select **Orchestration**, then select **Transformed Order Item Fulfillment State Composition Rule Set**.

   The Transformed Order Item Fulfillment State Composition Rule Set wizard is displayed.
2. In the **Project** field, select the project in which to save the composition rule set.
3. In the **Name** field, enter a name for the composition rule set.

   The name must be unique among transformed order item fulfillment state composition rule set entity types within the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the fulfillment state map.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the composition rule set.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or select a location different from the system-provided default. To select a different location:

   a. Click the **Folder** field **Browse** button.

   b. Navigate to the directory in which to save the entity.

   c. Click **OK**.
6. Click **Finish**.

   Design Studio adds the new composition rule set to the Studio Projects view and opens the new entity in the Transformed Order Item Fulfillment State Composition Rule Set editor.

Related Topics

About Transformed Order Item Fulfillment State Composition Rule Sets

Transformed Order Item Fulfillment State Composition Rule Set Editor

Transformed Order Item Fulfillment State Composition Rule Set Editor

Use the Transformed Order Item Fulfillment State Composition Rule Set editor to define the composition rules that apply to a particular transformed order item.

Field	Use
Description	Edit the display name of the transformed order item fulfillment state composition rule set.

Field	Use
Namespace	Select an existing namespace or enter a unique namespace in which to include the transformed order item fulfillment state composition rule. Design Studio uses the last saved namespace as the default.
Fulfillment Pattern	Select the orchestration fulfillment pattern to which the rules in the rule set apply. The orchestration fulfillment pattern hierarchy is honored here. If a rule is defined for a base fulfillment pattern, it will apply to the fulfillment patterns that extend it, unless a rule is defined for the child fulfillment pattern specifically.
Order Item	Select the order item to which the rules in the rule set apply.
Composition rule and condition list	The rules and conditions are evaluated in the order listed: use the **Move Up** and **Move Down** commands to ensure that higher priority rules and more restrictive conditions are listed first. The root-level items in this list are the rules, and the child items are the conditions. Do any of the following: • Select an existing rule or condition to edit it. • Click **Add Rule** to create a new transformed order item fulfillment state composition rule. • Click **Add Condition** to create a new transformed order item fulfillment state composition condition under the selected rule. • Click **Remove** to remove the selected rules and conditions. • Right-click a selected rule and select **Rename** to rename the rule or condition. • Right-click a selected rule or condition and select **Move Up** to move the selected rule or condition higher in the list. • Right-click a selected rule or condition and select **Move Down** to move the selected rule or condition lower in the list. See the following for more information about configuring composition rules: • Transformed Order Item Fulfillment State Rule Details Subtab • Transformed Order Item Fulfillment State Rule Information Subtab • Transformed Order Item Fulfillment State Source Order Item Subtab

Related Topics

About Transformed Order Item Fulfillment State Composition Rule Sets

Creating New Transformed Order Item Fulfillment State Composition Rule Sets

Transformed Order Item Fulfillment State Rule Details Subtab

Use the **Rule Details** subtab to specify the composite fulfillment state to be assigned to the associated order item when the selected rule's conditions are met. If no rule is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the composition rule. If you are localizing OSM, use the drop-down list to the right of the field to set different values for different locales.
Composite State	Select the fulfillment state to be assigned to the composition rule's order item when all of the rule's composition conditions are met.

See the following for more information about configuring rules and conditions on the **Rule Details** subtab:

- Transformed Order Item Fulfillment State Transformed Order Items Subtab
- Transformed Order Item Fulfillment State Condition Details Subtab
- Transformed Order Item Fulfillment State Condition Information Subtab

Related Topics

Transformed Order Item Fulfillment State Composition Rule Set Editor

Transformed Order Item Fulfillment State Transformed Order Items Subtab

Use the **Transformed Order Items** subtab to specify the criteria that must be met for the selected condition to be evaluated. If no condition is selected, these fields are disabled.

Field	Use
Order Item	Click **Select** to select the order item specification that contains the transformed order items.
Order Components	This field enables you to constrain the order components for which this condition is applied. If order components are defined here, the condition will be applied only for those order components. If no order component is defined, the rule will be applied for all order components. Do any of the following: • Click **Select** to select from a list of the order components defined in the workspace. • Click **Add** to create a new order component. • Click **Open** to open the specified order component in the Order Component Specification editor. • Select an order component from the list and click **Remove** to remove the order component from the list.
Relationship	This field enables you to determine whether the condition is applied based on the relationship between the original and transformed order items. If no relationship is configured, the condition will be applied regardless of the relationship. This list works in conjunction with the **Order Item Relationship Exists** check box. Do any of the following: • Click **Select** to select from a list of the relationship types used in conjunction with the transformed order items. • Click **Add** to create a new relationship type. • Click **Open** to open the specified relationship type in the Relationship Type editor. • Select a relationship type from the list and click **Remove** to remove the relationship type from the list.
Order Item Relationship Exists	Select this check box to indicate that one of the relationship types in the **Relationship** field must be defined between the original and transformed order items in order for the condition to be evaluated. Deselect this check box to indicate that the condition should be evaluated only if *none* of the relationship types defined in the **Relationship** field is defined between the original and transformed order items.

Related Topics

Transformed Order Item Fulfillment State Rule Details Subtab

Transformed Order Item Fulfillment State Condition Details Subtab

Use the **Condition Details** subtab to specify the criteria that must be met for the selected condition to be evaluated. If no condition is selected, these fields are disabled.

Field	Use
Description	Edit the display name of the condition. If you are localizing OSM, use the list to the right of the field to set different values for different locales.
Type	Select **Any** when the condition requires at least one of the input order item's fulfillment states to match one of the selected fulfillment states. Select **All** when the condition requires all of the input order item's fulfillment states to match one of the selected fulfillment states. Select **None** when the condition requires that none of the input order item's fulfillment states match any of the selected fulfillment states.
Fulfillment State tree	Select one or more fulfillment states for matching to the rule's input fulfillment states. If a parent fulfillment state is selected, it includes the child fulfillment states also.
Property Values	Select one or more order item properties, and then enter a value for each. For the condition to be met, the order item must contain the specified property and that property must have the specified value. If both property values and fulfillment states are defined on the same condition, both the property value and fulfillment state conditions must be met. Do any of the following: • Click **Select** to select from a list of the properties for the order item. • Click **Open** to open the appropriate order item specification with the property selected. • Select a property from the list and click **Remove** to remove the property from the list.

Related Topics

Transformed Order Item Fulfillment State Rule Details Subtab

Transformed Order Item Fulfillment State Condition Information Subtab

Use the **Condition Information** subtab to capture additional information about the condition.

Field	Use
Information	Enter any additional information about the condition that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Transformed Order Item Fulfillment State Rule Details Subtab

Transformed Order Item Fulfillment State Rule Information Subtab

Use the **Rule Information** subtab to capture additional information about the rule.

Field	Use
Information	Enter any additional information about the composition rule that is required by your specific situation. If you are localizing OSM, use the list to the right of the field to set different values for different locales.

Related Topics

Transformed Order Item Fulfillment State Composition Rule Set Editor

Transformed Order Item Fulfillment State Source Order Item Subtab

Use the **Source Order Item** subtab to filter the conditions to be evaluated based on the conceptual model entity associated with the source order item.

Field	Use
Simple	Select this option to filter the composition rules based on the conceptual model entities configured in the **Conceptual Model Entity** field.
Advanced	Select this option to use the expression in the **XQuery** field to filter the composition rules.
Conceptual Model Entity	This field enables you to define the conceptual model entities that must be associated with the order item for the condition to be evaluated. This field is only available if you have selected the **Simple** option.
XQuery	This field enables you to determine the relevant conceptual model entities using an XQuery expression. This field is only available if you have selected the **Advanced** option. See "About Creating XQuery Expressions with Design Studio" for more information about entering information into XQuery controls. See "About Transformed Order Item Fulfillment State XQuery Expressions" for more information about this particular XQuery field.

Related Topics

Transformed Order Item Fulfillment State Composition Rule Set Editor

27

Working with Order Lifecycle Managers

You use order lifecycle managers to support the optional Oracle Communications Order and Service Management (OSM) Order Lifecycle Management user interface (UI). Design Studio Order Lifecycle Manager entities define properties that are used in the Order Lifecycle Management UI to provide header information. They also enable you to map the fulfillment states in your solution to the standard order lifecycle manager states (referred to in the UI as order fulfillment states). When you use this UI, every order you model in Design Studio must be associated with an order lifecycle manager.

When working with order lifecycle managers, see the following topics:

- Creating New Order Lifecycle Managers
- Order Lifecycle Manager Editor

Creating New Order Lifecycle Managers

If you are using the Order Lifecycle Management UI, you must create a new order lifecycle manager in Design Studio for each order. You also map fulfillment states to order lifecycle manager states (referred to in the UI as order fulfillment states). The state mappings are system-wide and can all be located in the same order lifecycle manager as the configuration for one of the orders, or you can create a separate order lifecycle manager that contains only the fulfillment state mappings.

To create an order lifecycle manager:

1. From the **Studio** menu, select **New**, then select **Order and Service Management**, then select **Orchestration**, and then select **Order Lifecycle Manager**.

   The Order Lifecycle Manager wizard is displayed.
2. In the **Project** field, select the project in which to save the order lifecycle manager.
3. In the **Name** field, enter a name for the order lifecycle manager.

   The name must be unique among Order Lifecycle Manager entities in the same namespace.
4. In the **Namespace** field, select an existing namespace or enter a unique namespace in which to include the order lifecycle manager.

   Design Studio uses the last saved namespace as the default.
5. (Optional) Select a location for the order lifecycle manager.

   By default, Design Studio saves the entity to your default workspace location. You can enter a folder name in the **Folder** field or click Browse to select a different location.
6. Click **Finish**.

   Design Studio adds the new order lifecycle manager to the Studio Projects view and opens the new entity in the Order Lifecycle Manager editor.
7. Do one or both of the following:
   - To configure the properties that appear for an order in the Order Lifecycle Management UI, use the **Properties** tab.

- To configure mappings between fulfillment states and order lifecycle manager states, use the **Fulfillment State Mapping**.

> **Note:**
>
> Ensure that the mapping for a particular fulfillment state is only configured in one order lifecycle manager.

Related Topics

Order Lifecycle Manager Editor

Order Lifecycle Manager Editor

Use the Order Lifecycle Manager editor to configure the Order Lifecycle Manager entities that support the optional OSM Order Lifecycle Management UI.

The following fields are common to all of the Order Lifecycle Manager editor tabs:

Field	Use
Description	Edit the display name of the order lifecycle manager.

When configuring order lifecycle managers in the Order Lifecycle Manager editor, see the following topics:

- Order Lifecycle Manager Editor Properties Tab
- Order Lifecycle Manager Editor Fulfillment State Mapping Tab

Order Lifecycle Manager Editor Properties Tab

Use the **Properties** tab to set the properties and presentation-level information, such as Customer Name and Customer ID, for the Order Lifecycle Management UI. You can define, per role and view, where in the order the values for those fields come from. In the Order Lifecycle Management UI, users belonging to a specified role see values for Customer Name and Customer ID as extracted from fields with the associated view.

> **Note:**
>
> See "Order Lifecycle Manager Editor" for information about fields that appear on both Fulfillment Pattern editor tabs.

Field	Use
Order	Click **Select** and select the order to which this order lifecycle manager applies. This field is mandatory if you are defining an order summary binding (that is, if you are defining properties for an order) in this order lifecycle manager. If you are only using this order lifecycle manager to define the state mappings, it is optional. If you do not select a value for this field, the configuration of this entity applies to all orders.

Field	Use
Roles	Specify the roles that require access to the order. Do any of the following: • Click **Add** to select an existing role and view or add a new one. • Click **Remove** to remove a role from the list.
Role	Displays the role selected in the Roles table. Click **Select** to select a different role.
View	The view associated with the role selected in the Roles table. Click **Select** to select a different view.
Order Summary Headers area	Select the data elements that contain the header information to use in the Order Lifecycle Management UI. All data elements you select must be in the view specified in the **View** field. Users with the role selected in the Roles table will see the values obtained from these data elements. Select a role in the Roles table and do any of the following: • Click **Select** after the **Customer Name** field to select a data element from the order template to use as the customer name. • Click **Select** after the **Customer ID** field to select a data element from the order template to use as the customer ID. • Click **Select** after the **External Order ID** field to select a data element from the order template to use as the order ID.
Additional Properties	(Optional) Specify additional properties to send to the Order Lifecycle Management UI. Do any of the following: • Click **Add** to define a property name and select a data element from the appropriate view. • Click **Remove** to remove a property from the list.
Name	Displays the property selected in the Additional Properties table. Click **Select** to select a different property.
Path	Displays the path associated with the data element to use. The data element must be in the view displayed in the **View** field. Click **Select** to select a different data element.

Related Topics

Creating New Order Lifecycle Managers

Order Lifecycle Manager Editor

Order Lifecycle Manager Editor Fulfillment State Mapping Tab

Order Lifecycle Manager Editor Fulfillment State Mapping Tab

Use the **Fulfillment State Mapping** tab to map the fulfillment states for the order and the standard states displayed in the Order Lifecycle Management UI. See "Working with Fulfillment State Maps" and *OSM Concepts* for more information about fulfillment states.

> **Note:**
>
> See "Order Lifecycle Manager Editor" for information about fields that appear on both Fulfillment Pattern editor tabs.

Field	Use
Fulfillment State Mapping area	Specify the mappings between the fulfillment states configured on the order and the standard Order Lifecycle Manager states. Create a mapping by dragging a fulfillment state from the Fulfillment States list to the appropriate state in the Order Lifecycle Manager States list. If you want to remove mappings, do one of the following: • Right-click a fulfillment state and select **Remove Mapping** to remove the mapping for that fulfillment state. • Right-click an order lifecycle manager state and select **Remove Mappings** to remove all of the mappings for that order lifecycle manager state. If you have seen an error in the Problems view saying that you have invalid mappings, click any (mapped or unmapped) state in either list, right-click the state and select **Remove Invalid Mappings**. A mapping could become invalid if, for example, the corresponding fulfillment state was deleted. To see the state to which a given state is mapped, do one of the following: • Right-click a mapped state in the Fulfillment States list and select **Show Target Elements for Mapping** to highlight the element in the Order Lifecycle Manager States list to which the selected state is mapped. • Right-click a mapped state in the Order Lifecycle Manager States list and select **Show Source Elements for Mapping** to highlight the elements in the Fulfillment States list to which the selected state is mapped.

Related Topics

Creating New Order Lifecycle Managers

Order Lifecycle Manager Editor

Order Lifecycle Manager Editor Properties Tab

28

Packaging and Deploying Orchestration Cartridges

The following procedure describes the steps required to package an Oracle Communications Order and Service Management (OSM) orchestration cartridge and deploy it to the run-time environment:

1. Package the cartridge.

   Before you can deploy to the OSM run-time environment, you must determine which entities, libraries, and resources to include in the cartridge PAR file. See Packaging and Deploying Orchestration Cartridges "Packaging Order and Service Management Cartridges" for more information.

2. From the Design Studio main menu, select **Studio**, then **Show Environment Perspective**.

   An Environment perspective is a collection of views that enable you to create and manage the attributes associated with your environment. These attributes are organized within an Environment project. You must create at least one Environment project. See "Creating Environment Projects" for more information.

3. Select the Design Studio environment to which you want to connect.

   Design Studio environments contain the specific OSM server connection information. You must create at least one Design Studio environment. See "Working with Environment Projects" for more information.

4. Deploy the cartridge.

   See "Packaging and Deploying OSM Cartridges" for more information.

   > **Note:**
   >
   > You can also use the Optimized Deploy feature to deploy cartridges to a run-time environment. See "Deploying Cartridge Projects with Optimize Deploy" for more information.

5. Test the cartridge model.

   See "Testing OSM Cartridge Models" for more information.

A

XQuery Examples

You use XQuery expressions in various locations to implement key aspects of the Oracle Communications Order and Service Management (OSM) orchestration functionality. For information about these XQuery expressions, refer to the following topics:

- General XQuery Information
- Order Recognition Rule XQuery Expressions
- Decomposition XQuery Expressions
- Dependency XQuery Expressions
- Order Transformation Manager XQuery Expressions

General XQuery Information

This topic contains general or reference information about XQuery that applies the same in different situations.

When working with XQuery expressions, see the following topics:

- About Creating XQuery Expressions with Design Studio
- OSM XQuery Functions
- Referencing Items from a Distributed Order Template in XQuery Expressions

About Creating XQuery Expressions with Design Studio

In general, the way you enter XQuery information into editors in Design Studio is the same, regardless of the editor. The XQuery control in Design Studio generally has three tabs: XQuery, Instances, and Information. Following are general instructions for entering XQuery information into each of these tabs in Design Studio.

Using the XQuery Tab

The **XQuery** tab allows you to configure XQuery-based rules or elements, or identify the source of the XQuery-based rules or elements. Select one of the following options:

- Select **None** if the XQuery configuration is optional and not configured. When you select this option, Design Studio disables the remaining options in the subtabs.
- Select **Expression** and enter the XQuery expression in the corresponding text box. Click **Edit** to open the Edit XQuery dialog box, which displays the configured XQuery expression in a larger and resizable text box. You can edit the expression in the Edit XQuery dialog box and click **OK** to save your changes, or click **Cancel** to dismiss the dialog box without saving the changes.

> **Note:**
>
> Design Studio provides XQuery validation on basic syntax and semantics, and denotes errors with Problem markers.

- Select **File** to denote that the XQuery configuration is located in a file saved to the project **resources** directory. This option enables you to write your XQuery expressions using any XQuery editing application you have installed in your Eclipse environment. See the Eclipse online Help topic *Associating editors with file types* for more information.

  Click **Select** to open the Select XQuery File dialog box, which displays all XQuery files contained in the project **resources** directory. Select the appropriate XQuery file and click **OK**.

- Select **URI** to denote that the XQuery configuration is located in a remote URI location. For example, you might enter:

  http://osm_server/AIARecognitionRule.xqy

Click **Properties** to open the Properties view, where you can define the following information for the XQuery:

- Annotation: The optional XML annotation element allows you to provide information about the XQuery. Enter information (for example, HTML-formatted information) for external systems into the **Annotation <appinfo>** field. Enter information for human users into the **Annotation <documentation>** field.
- Language: When you work with multiple languages, you can select a different language for displaying the description and annotation. For more information, see "Defining Language Preferences" in *Modeling OSM Processes* Help.

Using the Instances Tab

You can define a Data Instance behavior to obtain data that is not included in the order data and make that data available to the rule. Click **Add** to add a Data Instance behavior. Select the Data Instance behavior and click **Properties** to configure the Data Instance behavior.

For more information, see "Defining Data Instance Behavior Properties" in *Modeling OSM Processes* Help.

Using the Notes Tab

Use this tab if you want to describe the intended use of the rule. For example, you might describe the functionality of a complex rule or provide instructions on its use.

OSM XQuery Functions

OSM-specific XQuery functions are available to you when writing XQuery expressions. These XQuery functions are contained in classes that you can declare in the prolog of your XQuery expression.

To see specifics about the functions available, install the OSM SDK and extract the OSM Javadocs from the *OSM_home*/**SDK/osm7.***w.x.y.z*-**javadocs.zip** file (where *OSM_home* is the directory in which the OSM software is installed and *w.x.y.z* represents the specific version numbers for OSM). See *OSM Installation Guide* for more information about installing the OSM SDK.

The specific classes that contain XQuery functions you might use are:

- OrchestrationXQueryFunctions: This class contains XQuery functions that are used in OSM Orchestration. To declare this class, put the following declaration in the prolog of your XQuery expression:

```
declare namespace osmfn =
"java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQueryFunctions";
```

- XQueryFunctions: This class contains XQuery functions that are used in the order transformation manager. To declare this class, put the following declaration in the prolog of your XQuery expression:

```
declare namespace otmfn =
"java:oracle.communications.ordermanagement.orchestration.transformation.XQueryFunctions.";
```

Referencing Items from a Distributed Order Template in XQuery Expressions

The distributed order template is an option you can set on an order item specification to modify the method used to store order item data. For more general information about the distributed order template, see *OSM Modeling Guide*.

When using a distributed order template, any XQuery expressions that reference order item data must be in a particular format.

For any order item that is not a transformed order item, you must include the namespace of the order item specification. Following is an example of an XQuery reference to the **lineItemID** property on the **InputOrderItem** order item with the namespace **http://ex_input.com**:

```
/ControlData/OrderItem[@type='{http://ex_input.com}InputOrderItem']/lineItemID
```

For transformed order items, the format depends on the source of the data for the transformed order item. Data that is defined in the order item specification itself will use the namespace for the order item specification, the same way that data would be referenced for an input order item. Following is an example of an XQuery reference to the **lineItemID** property on the **OutputOrderItem** order item with the namespace **http://ex_output.com**:

```
/ControlData/OrderItem[@type='{http://ex_output.com}OutputOrderItem']/lineItemID
```

Data that has been derived from a common model entity, for example an action, will use a different format. In the following situation:

- Order item name: **OutputOrderItem**
- Order item namespace: **http://ex_output.com**
- Conceptual model entity (in this case an Action) name: **SA_Add_Internet**
- Conceptual model cartridge name: **Model_Broadband**
- Conceptual model cartridge version: **1.0.0.0.0**
- Parameter name on SA_Add_Internet: **serviceLevel**

The reference would look like this:

```
/ControlData/OrderItem[@type='{http://ex_output.com}OutputOrderItem']/
dynamicParams[@type='{Model_Broadband/1.0.0.0.0}SA_Add_InternetType']/serviceLevel
```

Note that the type for the parameters contained in the conceptual model entity has the string "Type" appended to the name of the entity. Thus, the type contains **SA_Add_InternetType** rather than just SA_Add_Internet.

Order Recognition Rule XQuery Expressions

The following topics provide reference information about order recognition rule XQuery expressions:

- About Recognition Rule XQuery Expressions
- About Validation Rule XQuery Expressions
- About Order Priority XQuery Expressions
- About Order Reference XQuery Expressions
- About Order Data Rule XQuery Expressions

About Recognition Rule XQuery Expressions

This topic describes how to use the Order Recognition Rule editor Recognition Rule area **XQuery** tab to write an expression that specifies a customer order and associates it with an OSM target order type. The XQuery has the following characteristics:

- Context: The input document for the Recognition Rule **XQuery** is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: You can declare the namespace for the customer order if you want to use the contents of the order as part of the recognition rule or you can omit the declaration if you only want to check the incoming customer order namespace. For example:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  ```

- Body: You must match the namespace you want to select for order processing with the namespace of the incoming customer order. For example, the following expression retrieves the namespace URI from the incoming customer order (**fn:namespace-uri(.)**) and compares it with this URI: **'http://xmlns.oracle.com/InputMessage**':

  ```
  fn:namespace-uri(.) = 'http://xmlns.oracle.com/InputMessage'
  ```

  If you have declared a namespace in the prolog, you can also check to see if specific values exist in the order. For example, you can use the **fn:exists** function to check that an element exists. Or you can use a comparison expression such as **=** (equal to) or **!=** (not equal to) to compare a value in the customer order with a value in the XQuery.

> **Tip:**
>
> Recognition rules are global entities within OSM, meaning that they can apply to any CreateOrder operation. Configure the relevancy settings and the recognition rule carefully to avoid having an incoming customer order recognized by a recognition rule that you do not intend. For more information about relevancy, see *OSM Modeling Guide*.

For example, in a simple scenario, the XQuery is based on a namespace:

```
fn:namespace-uri(.) = 'http://xmlns.oracle.com/InputMessage'
```

The input message XML file includes the following line, which matches the namespace specified in the recognition rule:

```
<im:order xmlns:im="http://xmlns.oracle.com/InputMessage"
```

The XQuery expression returns a Boolean expression, for example, `fn:true()` or `fn:false()`

The following example searches in a specific type of order:

```
fn:namespace-uri(.) = 'http://xmlns.oracle.com/communications/sce/dictionary/
CentralOMManagedServices-Orchestration/CustomerSalesOrder'
```

In a more complicated scenario, you might create an XQuery expression that looks for a specific namespace and also interrogates the data within the incoming customer order. The following example shows a recognition rule that recognizes an order based on the following criteria:

- Namespace
- Value of the **typeCode** data element in the incoming customer order. In this case, the value must be OSM-BDB. This indicates an OSM business-to-business order.
- The value of the **FulfillmentModeCode** data element in the incoming customer order. In this case, the value can be DELIVER, CANCEL, or TSQ.

```
declare namespace provord=";http://xmlns.oracle.com/EnterpriseObjects/Core/EBO/ProvisioningOrder/V1";;
declare namespace corecom=";http://xmlns.oracle.com/EnterpriseObjects/Core/Common/V2";;
fn:namespace-uri(.) = 'http://xmlns.oracle.com/EnterpriseObjects/Core/EBO/ProvisioningOrder/V1'
and
fn:exists(../provord:ProcessProvisioningOrderEBM/provord:DataArea/provord:ProcessProvisioningOrder/
corecom:Identification/corecom:BusinessComponentID)
and
../provord:ProcessProvisioningOrderEBM/provord:DataArea/provord:ProcessProvisioningOrder/
provord:TypeCode/text() = 'OSM-BDB'
and
(
../provord:ProcessProvisioningOrderEBM/provord:DataArea/provord:ProcessProvisioningOrder/
provord:FulfillmentModeCode/text() = 'DELIVER'
or
../provord:ProcessProvisioningOrderEBM/provord:DataArea/provord:ProcessProvisioningOrder/
provord:FulfillmentModeCode/text() = 'CANCEL'
or
../provord:ProcessProvisioningOrderEBM/provord:DataArea/provord:ProcessProvisioningOrder/
provord:FulfillmentModeCode/text() = 'TSQ'
)
```

For more information about order recognition rules see *OSM Modeling Guide*.

About Validation Rule XQuery Expressions

This topic describes how to use the Order Recognition Rule editor Validation Rule area **XQuery** tab to write an expression that specifies nodes in the incoming customer order that must evaluate to true to accept the customer order into the system. The XQuery has the following characteristics:

- Context: The input document for the Validation Rule **XQuery** is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: The input document for the Validation Rule **XQuery** is the customer order. You can declare the customer order namespace in the XQuery prolog. For example:

```
declare namespace im="http://xmlns.oracle.com/InputMessage";
```

- Body: The validation rule must specify customer order parameters or parameter values to evaluate to **true** for the validation to be successful. If the validation fails, the expression should return an error message.

  In addition, if the Validation Rule fails, OSM automatically creates the order and sets the order state to Failed. The inbound message and validation failure output are attached to the order for reference. You can display and manage the order failure in the Order Management web client.

The following sample XQuery checks for the existence of a sender ID:

```
if (fn:exists(./header/c:Sender/c:ID) and ./header/c:Sender/c:ID != '')
  then (true())
     else concat("SEVERE", "Message Header should contain Sender ID", header/Sender/ID")
```

The following sample XQuery checks for correct values in the **typeCode** data element in the incoming customer order:

```
if (fn:exists($orderLine/im:ItemReference/im:TypeCode)
    and
    $orderLine/im:ItemReference/im:TypeCode != '')
then
    (
    if ($orderLine/im:ItemReference/im:TypeCode = "PRODUCT" or
        $orderLine/im:ItemReference/im:TypeCode = "OFFER" or
        $orderLine/im:ItemReference/im:TypeCode = "BUNDLE") then ()
    else
       local:reportIssue("ERROR", "Product Type should be one of: PRODUCT, OFFER, BUNDLE",
             $lineNum, "ProcessProvisioningOrderEBM/DataArea/ProcessProvisioningOrder/
                        ProvisioningOrderLine/ItemReference/TypeCode")
)
```

Given this XQuery sample, the following part of a customer order would evaluate to true because the typeCode element value is **BUNDLE**.

```
<!-- FIXED BUNDLE - BUNDLE -->
<im:salesOrderLine>
      <im:lineId>2</im:lineId>
  <im:promotionalSalesOrderLineReference>1</im:promotionalSalesOrderLineReference>
      <im:serviceId></im:serviceId>
      <im:requestedDeliveryDate>2001-12-31T12:00:00</im:requestedDeliveryDate>
      <im:serviceActionCode>Add</im:serviceActionCode>
                  <im:itemReference>
                         <im:name>Fixed Bundle</im:name>
                         <im:typeCode>BUNDLE</im:typeCode>
                         <im:specificationGroup />
      </im:itemReference>
</im:salesOrderLine>
```

For more information about validation rules see *OSM Modeling Guide*.

About Order Priority XQuery Expressions

This topic describes how to use the Order Recognition Rule editor Order Priority area **XQuery** tab to write an expression that specifies an element value in the incoming customer order that identifies the order priority. The XQuery has the following characteristics:

- Context: The input document for the Order Priority XQuery is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.

- Prolog: You can declare the customer order namespace in the XQuery prolog. For example:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  ```

- Body: The Order Priority body must specify the node that contains the order priority value.

For more information about creating order priority XQuery expressions in the order recognition rule and about creating order priority ranges for an order type, see *OSM Modeling Guide*.

About Order Reference XQuery Expressions

This topic describes how to use the Order Recognition Rule editor Order Reference area **XQuery** tab to write an expression that specifies an element value in the incoming customer order that identifies the order reference. The XQuery has the following characteristics:

- Context: The input document for the Order Reference XQuery is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: You can declare the customer order namespace in the XQuery prolog. For example:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  ```

- Body: The Order Reference body must specify the node that contains the order reference value.

The following example shows a transformation rule XQuery expression that retrieves the order reference number (as a string) from the **numSalesOrder** field in the incoming customer order:

```
declare namespace im="http://xmlns.oracle.com/InputMessage";
let $order := ../im:order
return
$order/im:numSalesOrder/text()
```

For more information about order reference, see *OSM Modeling Guide*.

About Order Data Rule XQuery Expressions

This topic describes how to use the Order Recognition Rule editor Order Data Rule area **XQuery** tab to write an expression that specifies nodes in the incoming customer order that must be used in the creation task. The XQuery has the following characteristics:

- Context: The input document for the Order Data Rule XQuery is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: You can declare the customer order namespace in the XQuery prolog. For example:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  ```

- Body: The Order Data Rule body must map the customer order element values to the corresponding creation task Task Data values.

  The following example shows the fields in an incoming customer order:

  ```
  <im:customerAddress>
       <im:locationType>Street</im:locationType>
       <im:nameLocation>Jangadeiros</im:nameLocation>
       <im:number>48</im:number>
       <im:typeCompl>floor</im:typeCompl>
       <im:numCompl>6</im:numCompl>
       <im:district>Ipanema</im:district>
  ```

```
     <im:codeLocation>5000</im:codeLocation>
     <im:city>Rio de Janeiro</im:city>
     <im:state>RJ</im:state>
     <im:referencePoint>Gen. Osorio Square</im:referencePoint>
     <im:areaCode>22420-010</im:areaCode>
     <im:typeAddress>Building</im:typeAddress>
</im:customerAddress>
```

Following is a sample order data in a creation task. In the example, the following data is contained in a CustomerDetails element:

- locationType
- nameLocation
- number
- typeCompl
- numCompl
- district
- codeLocation
- city
- state
- referencePoint
- areaCode
- typeAddress

The following XQuery expression specifies a variable for the location of the customerAddress node in the customer order that can then be used to map customerAddress child element values to CustomerDetails task data elements:

```
let $details := $customer/mes:customerAddress
```

The following XQuery expression performs this mapping:

```
return<_root>
<CustomerDetails>
     <locationType>{$details/im:locationType/text()}</locationType>
     <nameLocation>{$details/im:nameLocation/text()}</nameLocation>
     <number>{$details/im:number/text()}</number>
     <typeCompl>{$details/im:typeCompl/text()}</typeCompl>
     <numCompl>{$details/im:numCompl/text()}</numCompl>
     <district>{$details/im:district/text()}</district>
     <codeLocation>{$details/im:codeLocation/text()}</codeLocation>
     <city>{$details/im:city/text()}</city>
     <state>{$details/im:state/text()}</state>
     <referencePoint>{$details/im:referencePoint/text()}</referencePoint>
     <areaCode>{$details/im:areaCode/text()}</areaCode>
     <typeAddress>{$details/im:typeAddress/text()}</typeAddress>
</CustomerDetails>
</_root>
```

In the following example, the XQuery expression returns the <*_root*> portion of the order. The ControlData portion of the order is populated by the system during the generation of the orchestration plan.

```
declare namespace cso="http://xmlns.oracle.com/communications/sce/dictionary/
CentralOMManagedServices-Orchestration/CustomerSalesOrder";
let $customer :=  //cso:CustomerAccount
```

```
return
<_root>
<OrderHeader>
<AccountIdentifier>{$customer/cso:AccountID/text()}</AccountIdentifier>
</OrderHeader>
</_root>
```

For more information about order data rules, see *OSM Modeling Guide*.

Decomposition XQuery Expressions

This topic includes information about order recognition rule XQuery expressions related to order decomposition:

- About Orchestration Sequence XQuery Expressions
- About Order Item Specification XQuery Expressions
- About Orchestration Fulfillment Pattern Order Component XQuery Expressions
- About Decomposition Rule Condition XQuery Expressions
- About Component Specification Custom Component ID XQuery Expressions
- About Component Specification Duration XQuery Expressions
- About Orchestration Fulfillment Pattern Duration XQuery Expressions (deprecated)
- About Orchestration Fulfillment Pattern Component Duration XQuery Expressions

About Orchestration Sequence XQuery Expressions

The Orchestration Sequence editor provides the following areas to define XQuery expressions related to order decomposition:

- About Order Sequence Order Item Selector XQuery Expressions
- About Order Sequence Fulfillment Mode XQuery Expressions

About Order Sequence Order Item Selector XQuery Expressions

This topic describes how to use the Orchestration Sequence editor Order Item Selector area **XQuery** tab to write an expression that specifies which node-set to use from the customer order as order items and has the following characteristics:

- Context: The input document for the Order Item Selector XQuery is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: You can declare the customer order namespace in the XQuery prolog.
- Body: The XQuery body must specify the customer order node-sets that OSM then uses as order items.

The following example shows an order item selector XQuery where the **<salesOrderLine>** node-set is specified. OSM can now use the data in the **<salesOrderLine>** node-set in the incoming customer order in the order items. There can only be one node-set selected per sequence.

```
declare namespace im="http://xmlns.oracle.com/InputMessage";
.//im:salesOrderLine
```

About Order Sequence Fulfillment Mode XQuery Expressions

This topic describes how to use the Orchestration Sequence editor Fulfillment Mode area **XQuery** tab to write an expression that specifies the fulfillment mode for the orchestration sequence from a customer order element and has the following characteristics:

- Context: The input document for the Fulfillment Mode Expression area XQuery is the customer order. For more information about typical customer order structures, see *OSM Modeling Guide*.
- Prolog: The input document for the Fulfillment Mode Expression area XQuery is the incoming customer order. You must declare the customer order namespace in the XQuery prolog.
- Body: The XQuery body must specify the fulfillment mode.

Typically, the fulfillment mode is specified in the order header. For example:

```
<im:FulfillmentModeCode>Deliver</im:FulfillmentModeCode>
```

In the following example, the XQuery looks in the incoming customer order (SalesOrder) for the **<FulfillmentModeCode>** element. It returns the text contained in that element.

```
declare namespace
im="http://xmlns.oracle.com/InputMessage";
<osm:fulfillmentMode name="{fn:normalize-space(.//im:SalesOrder/im:DataArea/im:FulfillmentModeCode/
text())}"
```

This is the XML in the incoming customer order:

```
<im:FulfillmentModeCode>Deliver</im:FulfillmentModeCode>
```

In this case, the XQuery returns **Deliver**.

About Order Item Specification XQuery Expressions

The Order Item Specification editor provides the following areas to define XQuery expressions related to order decomposition:

- About Order Item Specification Order Item Property XQuery Expressions
- About XQuery Expressions for Mapping Products and Fulfillment Patterns
- About Order Item Specification Order Item Hierarchy XQuery Expressions
- About Order Item Specification Condition XQuery Expressions

About Order Item Specification Order Item Property XQuery Expressions

This topic describes how to use the Order Item Specification editor, **Order Item Properties** tab, Property Expression area, **XQuery** tab to write an expression that specifies order item properties based on the input context. These expressions have the following characteristics:

- Context: The Property Expression area XQuery input document is a node from the node-set returned by the order item selector. See "About Order Sequence Order Item Selector XQuery Expressions". OSM runs every order item Property Expression area XQuery against each node (starting with the first and ending with the last node) in the node-set returned by the order item selector.
- Prolog: You can declare the following variables within the prolog to access additional context information:

- The **$inputDoc** variable can be declared in the prolog of an OSM XQuery to provide access to the original input customer order. This external function can be useful if you need to generate order item properties based on elements outside of the order item node-set defined in the order item selector. The format for declaring this variable in the XQuery prolog is:

  ```
  declare variable $inputDoc as document-node() external;
  ```

  You can then access this variable within the XQuery body. For example, the following XQuery body uses **$inputDoc** to define the **ItemReferenceName** value:

  ```
  let $inputOrderData:= $inputDoc/GetOrder.Response/_root
  fn:normalize-space(cso:ItemReferenceName/text())
  ```

  For more information about typical customer order structures, see *OSM Modeling Guide*.

- The **$salesOrderLines** variable can be used in an OSM XQuery to provide access to original order item node-set before it is selected by the orchestration sequence's order item selector. This can be useful if the order item selector XQuery changes the selected order item node-set (for example, by rearranging the order of the elements). The format for declaring this variable in the XQuery prolog is:

  ```
  declare variable $salesOrderLines as document-node() external;
  ```

  You can access this variable within the XQuery body. For more information about typical customer order structures, see *OSM Modeling Guide*.

- Body: The XQuery body must specify the order item element that provides the values for each order item property you define.

After these XQuery expressions have run against an order item, the order item and the order item properties become internally accessible as an XQuery context for other OSM entities. For example,

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model" id="1288881040699">
   <osm:name>Commercial Fixed Service [Add]</osm:name>
   <osm:orderItemSpec xmlns="http://xmlns.oracle.com/communications/ordermanagement/model">
      <osm:name>CustomerOrderItemSpecification</osm:name>
      <osm:namespace>
         http://oracle.communications.ordermanagement.unsupported.centralom
      </osm:namespace>
   </osm:orderItemSpec>
   <osm:productSpec xmlns="http://xmlns.oracle.com/communications/ordermanagement/model">
      <osm:name>Service.Fixed</osm:name>
      <osm:namespace>
         http://oracle.communications.ordermanagement.unsupported.centralom
      </osm:namespace>
   </osm:productSpec>
   <osm:properties xmlns:im="http://oracle.communications.ordermanagement.unsupported. centralom">
      <im:typeCode>PRODUCT</im:typeCode>
      <im:parentLineId>3</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
      <im:lineId>4</im:lineId>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productSpec>Fixed Service Plan Class</im:productSpec>
      <im:serviceId>552131313131</im:serviceId>
      <im:fulfillPatt>Service.Fixed</im:fulfillPatt>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
```

```
   <osm:properties>
</osm:orderItem>
```

The following examples show some ways to map data in an incoming customer order to an order item property. The current context is a single node from salesOrderLines, which is one of the nodes returned by running the orchestration sequence order item selector against the input message. See "About Order Sequence Order Item Selector XQuery Expressions".

- Order management personnel need to know what the requested delivery date is for order items. Adding the date to the order item allows the order management personnel to see it in the OSM web clients. In addition, OSM needs the requested delivery date to calculate the order start date.

  To retrieve the requested delivery data for an order item, OSM looks in the incoming customer order for the **<requestedDeliveryDate>** element:

  ```
  <im:requestedDeliveryDate>2001-12-31T12:00:00</im:requestedDeliveryDate>
  ```

  The definition of the **requestedDeliveryDate** order item property includes the following XQuery, which returns the text of the **<requestedDeliveryDate>** element:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  fn:normalize-space(im:requestedDeliveryDate/text())
  ```

- Order management personnel need to identify order items in the OSM web clients. The **lineItemName** order item property includes the following XQuery:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  fn:normalize-space(fn:concat(im:itemReference/im:name/text(),'
  [',im:serviceActionCode/text(),']'))
  ```

  This XQuery looks for two elements, **<name>** and **<serviceActionCode>**:

  ```
  <im:name>Fixed Caller ID</im:name>
  <im:serviceActionCode>Add</im:serviceActionCode>
  ```

  It then concatenates the text retrieved from the two elements to form the order item name, in this case Fixed Caller ID [Add].

- Order management personnel need to identify the products or orchestration product specifications from the customer order so that order items can be mapped to orchestration fulfillment patterns. See "About XQuery Expressions for Mapping Products and Fulfillment Patterns". The following example shows the product specification data in the message, contained in the **<primaryClassificationCode>** element:

  ```
  <im:primaryClassificationCode>Mobile Service Feature Class
  </im:primaryClassificationCode>
  ```

  The **productClass** order item property uses the following XQuery expression to get the data:

  ```
  declare namespace im="http://xmlns.oracle.com/InputMessage";
  fn:normalize-space(im:itemReference/im:primaryClassificationCode/text())
  ```

About XQuery Expressions for Mapping Products and Fulfillment Patterns

OSM relies on the fulfillment pattern mapping property of the order item specification to determine the fulfillment pattern for a line item.

Historically, the fulfillment pattern name needed to be derived by cartridge developers at design time using a mapping file generated by Design Studio. Cartridges that have been developed

using these patterns can continue to be built and deployed with the latest version of Design Studio.

This mechanism has been simplified, however, when using OSM 7.5.0 or later (target server version of the cartridges must reflect this) along with Design Studio 8.0.0.0.0 or later.

Cartridge developers can now simply populate the **Fulfillment Pattern Mapping Property** field with the **product specification name** and OSM will determine the correct fulfillment pattern based on new files generated by Design Studio.

This can be done by providing an XQuery that navigates the incoming order to the data structure where this data is held. The location of the product specification name will be specific to the incoming order payload.

For example, for a TMF 641 order, the following XQuery will return the service name:

```
(: Declare incoming order namespace:)
declare namespace tmf="http://oracle.communications.orchestration.com/tmf-api/
serviceOrdering/%{TMF641_VERSION}/serviceOrder/inputMessage";

declare variable $line :=.;

fn:normalize-space($line/tmf:service/tmf:serviceSpecification/tmf:name/text())
```

For example, for a TMF 622 order, the following XQuery snippet will return the product name:

```
(: Declare incoming order namespace:)
declare namespacetmf="http://oracle.communications.orchestration.com/tmf-api/
productOrderingManagement/%{TMF622_VERSION}/productOrder/inputMessage";

declare variable $line :=.;

let $ps := $line/tmf:product/tmf:productSpecification/tmf:name/text()
return $ps
```

Developers still have the option of returning the fulfillment pattern name if they wish. This is particularly useful when you have a default fulfillment pattern for line items that do not represent a product or service.

As an example, you could enhance the above XQuery examples with a check for an empty product or service name, and if necessary return a fulfillment pattern instead (where **Non.Service.Offer** is a fulfillment pattern).

```
   else

     'Non.Service.Offer'
```

About Order Item Specification Order Item Hierarchy XQuery Expressions

This topic describes how to use the Order Item Specification editor **Order Item Hierarchies** tab, Key Expression and Parent Key Expression areas, **XQuery** subtabs to write expressions that specify the relative hierarchy of order items, in the same order or between different orders, based on an order item value, such as **lineId** and **parentLineId** and has the following characteristics:

- Context: The Key Expression and Parent Key Expression area XQuery input document is the order item. Specifically order item properties that indicate the relative hierarchy, such as order item **lineId** and **parentLineID** properties. For example:

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
```

```
 id="1288881040699">
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>PRODUCT</im:typeCode>
      <im:parentLineId>3</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
      <im:lineId>4</im:lineId>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Fixed Service Plan Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Fixed</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

- Prolog: You can declare the order item specification namespace and the OSM namespace in the XQuery prolog. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="http://
  oracle.communications.ordermanagement.unsupported.centralom";
  ```

  You can declare the **OrchestrationXQueryFunctions** class in the prolog to use the ancestors method that returns the current node and all ancestors of the current node based on the specified hierarchy definition. This method can be useful when creating dependencies between order items based on hierarchy. For example:

  ```
  declare namespace osmfn =
  "java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQu
  eryFunctions";
  ```

  See "OSM XQuery Functions" for more information about the **OrchestrationXQueryFunctions** class. See *OSM Modeling Guide* for an example of how the **ancestors** method is used.

- Body: The XQuery body must specify an order item property defined in the order item specification.

  For example, for the Key Expression, you can identify a unique key for each order item, typically the order item line ID:

  ```
  fn:normalize-space(osm:properties/im:LineId/text())
  ```

  For example, for the Parent Key Expression, you can identify a parent order line item, typically the line ID for the parent order line item:

  ```
  fn:normalize-space(osm:properties/im:parentLineId/text())
  ```

In the following example, the key expression uses the parent order line item's **<lineId>** element from the order item property customer order:

```
declare namespace im="http://oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
fn:normalize-space(osm:properties/im:lineId/text())
```

The parent key expression uses the child order line item's **<parentLineId>** element from the incoming customer order:

```
declare namespace im="http://oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
fn:normalize-space(osm:properties/im:parentLineId/text())
```

About Order Item Specification Condition XQuery Expressions

This topic describes how to use the Order Item Specification editor **Orchestration Conditions** tab, Condition Expression area, **XQuery** subtab to write expressions that specifies an order item property value as a condition that you can then use in an order decomposition rule or in a fulfillment pattern to determine whether an order item gets included in an order component. The XQuery for the condition has the following characteristics:

- Context: The Condition Expression area XQuery input document is the order item properties you want to use as conditions. For example, the following order item contains the **region** and the **ServiceActionCode** order item properties, that could be associated to conditions:

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
 id="1288881040699">
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>PRODUCT</im:typeCode>
      <im:parentLineId>3</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
      <im:lineId>4</im:lineId>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Fixed Service Plan Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Fixed</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

  See "About Orchestration Fulfillment Pattern Order Component Condition XQuery Expressions" for a description of the XQuery condition based on the **ServiceActionCode**. See "About Decomposition Rule Condition XQuery Expressions" for a description of the XQuery condition based on the **region**.

- Prolog: You can declare the order item specification namespace and the OSM namespace in the XQuery prolog. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="http://
  oracle.communications.ordermanagement.unsupported.centralom";
  ```

- Body: The XQuery body must evaluate an order item property defined in the order item specification. These order item properties are available from the OSM namespace using the properties element. For example, the following expression evaluates to true if the value of **region** is anything other than **Sao Paulo** and the order item gets included in the order component. If the **region** were set to **Sao Paulo**, then the order item would not be included in the order component.

  ```
  fn:not(fn:normalize-space(osm:properties/im:region/text())='Sao Paulo')
  ```

Another condition could be created that would only evaluate to true if the value of **region** was set to **Sao Paulo**. In this case, the order item would only be included in the order component if the **region** were set to **Sao Paulo**.

About Orchestration Fulfillment Pattern Order Component XQuery Expressions

The Orchestration Fulfillment Pattern editor provides the following areas to define XQuery expressions related to order decomposition:

- About Orchestration Fulfillment Pattern Order Component Condition XQuery Expressions
- About Associating Order Items Using Property Correlations XQuery Expressions

> **Note:**
>
> The XQuery expressions discussed in this chapter also apply to the Orchestration Dependency editor.

About Orchestration Fulfillment Pattern Order Component Condition XQuery Expressions

This topic describes how to use the Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Order Components** subtab, **Conditions** subtab **XQuery** subtab to write an expression that specifies whether to include or exclude an order item from an order component. You can create a new orchestration fulfillment pattern from the Orchestration Fulfillment Pattern editor or select from conditions created in the Order Item Specification. See "About Order Item Specification Condition XQuery Expressions" for more information about the context, prolog, and body of condition XQuery expressions.

The following example XQuery expression only evaluates to true if the value of **ServiceActionCode** is not NONE or UPDATE. For example, if the value of **ServiceActionCode** were ADD, then the order item would be included in the order component.

```
fn:boolean
(
(osm:properties/im:ServiceActionCode/text()!="NONE" and
osm:properties/im:ServiceActionCode/text()!="UPDATE") or
(
```

About Associating Order Items Using Property Correlations XQuery Expressions

This topic describes how to use the Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Order Components** subtab, **Order Item Association** subtab, Property Correlation selection, **XQuery** subtab to write an expression that associates order items to order components that are not assigned by their orchestration fulfillment pattern. These order item associations are typically required when external systems need a specific context for an order item and includes the following characteristics:

- Context: The **Order Item Association** subtab XQuery input documents are multiple order items in the order after decomposition contained in the **fromOrderComponent** element and the entire set of order items included in the order contained in the

toOrderComponent element. You can make an XQuery association based on the contents of these order items that create an association between the unique order item IDs. For example:

```
<fromOrderComponent xmlns="">
   <osm:orderItem
    xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
    id="1234">
      <osm:name>Speed By Demand [Add]</osm:name>
.....
      <osm:properties
       xmlns:im="http://oracle.communications.ordermanagement.unsupported.
       centralom">
         <im:typeCode>PRODUCT</im:typeCode>
         <im:parentLineId>3</im:parentLineId>
         <im:requestedDeliveryDate>2013-06-31T12:00:00
         </im:requestedDeliveryDate>
         <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
         <im:lineId>4</im:lineId>
         <im:SiteID>10</im:SiteID>
         <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
         <im:productClass>Speed by Demand class</im:productClass>
         <im:serviceId>552131313131</im:serviceId>
         <im:productSpec>Service.Fixed</im:productSpec>
         <im:lineItemPayload> [34 lines]
         <mi:region>Sao Paulo</im:region>
      <osm:properties>
   </osm:orderItem>
   <osm:orderItem [37 lines]
   <osm:orderItem [42 lines]
   <osm:orderItem [57 lines]
   <osm:orderItem [57 lines]
</fromOrderComponent>
<toOrderComponent xmlns="">
   <osm:orderItem [35 lines]
   <osm:orderItem [37 lines]
   <osm:orderItem [42 lines]
   <osm:orderItem
    xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
    id="5678">
      <osm:name>Broadband Bundle [Add]</osm:name>
.....
      <osm:properties
       xmlns:im="http://oracle.communications.ordermanagement.unsupported.
       centralom">
         <im:typeCode>PRODUCT</im:typeCode>
         <im:parentLineId>3</im:parentLineId>
         <im:requestedDeliveryDate>2013-06-31T12:00:00
         </im:requestedDeliveryDate>
         <im:lineItemName>Broadband Bundle [Add]</im:lineItemName>
         <im:lineId>4</im:lineId>
         <im:SiteID>10</im:SiteID>
         <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
         <im:productClass>Broadband Bundle Class</im:productClass>
         <im:serviceId>1112223333</im:serviceId>
         <im:productSpec>Broadband.Bundle</im:productSpec>
         <im:lineItemPayload> [34 lines]
         <im:region>Sao Paulo</im:region>
      <osm:properties>
   </osm:orderItem>
   <osm:orderItem [57 lines]
   <osm:orderItem [57 lines]
```

```
    <osm:orderItem [42 lines]
    <osm:orderItem [37 lines]
    <osm:orderItem [37 lines]
    <osm:orderItem [57 lines]
</toOrderComponent>
```

- Prolog: You can declare the order item namespace and the OSM namespace in the XQuery prolog. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="http://
  oracle.communications.ordermanagement.unsupported.centralom";
  ```

- Body: The XQuery body must specify a dependency between the order item and the associated order item using something similar to the following syntax:

  ```
  let $fromItem := osm:fromOrderComponent/osm:orderItem[osm:name/text()="Speed By
  Demand [Add]"]
  let $toItem := osm:toOrderComponent/osm:orderItem[osm:name/text()="Broadband Bundle
  [Add]" and osm:properties/im:SiteID/text() = $fromItem/osm:properties/im:SiteID/
  text()]
  return
  <osm:dependency fromOrderItemId='{$fromItem/@id}' toOrderItemId='{$toItem/@id}'/>
  ```

  where

  - **osm:fromOrderComponent**: Returns the set of order items included in the order component after the decomposition phase.
  - **osm:toOrderComponent**: Returns the entire set of order items included in the order.
  - **osm:orderItem**: These are the order items in the **fromOrderComponent** or **toOrderComponent** categories.
  - **osm:dependency fromOrderItemId='{$fromItem/@id}**: The output of the XQuery specifying the source order item ID for the association.
  - **toItem='{$childOrderItem/@id}'/>**: The output of the XQuery specifying the target order item ID for the association.

  Given the sample provided in the context bullet, this XQuery would return the following association:

  ```
  <osm:dependency fromOrderItemId='1234' toOrderItemId='5678'/>
  ```

The following example shows an XQuery that associates all child order items with their parent items. See *OSM Modeling Guide* for more information. The output of the XQuery expression returns a node-set of <osm:dependency fromOrderItemId='{$fromOrderItem/@id}' toOrderItemId=' {$toOrderItem/@id}'/> where item IDs are the @id attribute of the order item.

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace prop="http://oracle.communications.ordermanagement.unsupported.centralom";
(:   $fromOrderItemList contains all order items in the selected order component:   :)
for $fromOrderItem in $fromOrderItemList
let $fromOrderItemList := osm:fromOrderComponent/osm:orderItem
(:    $childOrderItems contains all children for the current $fromOrderItem:    :)
let $childOrderItems := osm:toOrderComponent/osm:orderItem/osm:properties
      [prop:ParentLineID/text() = $fromOrderItem/osm:properties/prop:LineID/text()]
(:    Returns the association between all parents and their children:    :)
for $childOrderItem in $childOrderItems
return
  <osm:dependency fromOrderItemId='{$fromOrderItem/@id}' toOrderItemId='{$childOrderItem/@id}'/>
```

About Decomposition Rule Condition XQuery Expressions

This topic describes how to use the Decomposition Rule editor, **Conditions** tab, **Conditions Details** subtab, **XQuery** subtab to write an expression that associates a condition with a decomposition rule. You can create the condition in the order item specification and select them in the decomposition rule, or you can create them directly in the decomposition rule. See "About Order Item Specification Condition XQuery Expressions" for more information about the context, prolog, and body of condition XQuery expressions.

The following is an example of two decomposition rules, each having a condition set that determines whether an order item is included in the target order component or not. In this example:

- The decomposition rule that targets the target system order component for region 1 has the following decomposition condition:

  ```
  isRegion1
  ```

- The decomposition rule that targets the target system order component for region 2 has the following decomposition condition:

  ```
  isOtherRegion
  ```

The XQuery for the **isRegion1** decomposition rule condition is:

```
declare namespace im="http://oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
fn:normalize-space(osm:properties/im:region/text())='Toronto')
```

This condition specifies the value of the **region** order item property. If the value is **Toronto**, the decomposition rule condition is true, and the order item is included in the region 1 target system order component.

The XQuery for the **isOtherRegion** decomposition rule condition is:

```
declare namespace im="http://
oracle.communications.ordermanagement.unsupported.centralom";declare namespace
osm="http://xmlns.oracle.com/communications/ordermanagement/model";fn:not(fn:normalize-
space(osm:properties/im:region/text())='Toronto')
```

This condition also specifies the value of the **region** order item property, but evaluates to true only if the value is not **Toronto**. All order items that have any other value are included in the region 2 target system order component.

The following example includes a variation on the **isRegion1** decomposition rule that specifies that all the order items from the source order component to the target order component that have at least one order item with a **region** property of **Toronto** are included in the order component. Otherwise, if the condition evaluates to false then none of the order items in **fromOrderComponent** are included in the resulting order component.

```
declare namespace im="http://oracle.communications.centralom";
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
fn:exists(osm:fromOrderComponent/osm:orderItem[fn:normalize-space(osm:properties/
im:Region/text())='Toronto'])
```

For some functions, there is only one target system in the topology. For example, if you have only one collections system in the topology, you will have one dependency rule that uses a simple mapping from the source collections function order component to the collections target system order component, and no decomposition condition is necessary.

About Component Specification Custom Component ID XQuery Expressions

This topic describes how to use the Order Component Specification editor, **Component ID** tab, Component ID area, **XQuery** subtab to write an expression that specifies a custom component ID for an order component. These custom component IDs are typically required when the default component IDs are not sufficiently specific. See *OSM Modeling Guide* for more information about the default component ID. The Component ID XQuery includes the following characteristics:

- Context: The **Component ID** tab XQuery input document is the order item and the order item properties you want to use to create a custom component ID with. For example, the following order item contains the **SiteID** and **requestedDeliveryDate** order item properties. In a simple scenario, you can use this element to group all order items that share the same **SiteID** value and further delineate groups based on **requestedDeliveryDate** date range.

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
 id="1288881040699">
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>Bundle</im:typeCode>
      <im:parentLineId>3</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
      <im:lineId>4</im:lineId>
      <im:SiteID>10</im:SiteID>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Fixed Service Plan Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Fixed</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

- Prolog: You can declare the order item namespace and the OSM namespace in the XQuery prolog. In more complicated XQuery expressions, you can also use the **OrchestrationXQueryFunctions** OSM Java package to specify component IDs based on order item hierarchies, order item requested delivery date, order component duration, order component minimum duration separation, or a combination of some or all of them. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="http://
  oracle.communications.ordermanagement.unsupported.centralom";
  declare namespace osmfn =
  "java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQu
  eryFunctions";
  ```

  See "OSM XQuery Functions" for more information about the **OrchestrationXQueryFunctions** class.

- Body: The body must return a string. Every order item that ends with the same string gets included in the order component. For example, if you wanted to group all order items based on the **SiteID** value, you could specify the following XQuery:

```
return osm:properties/im:SiteID/text()
```

The following topics describe OSM Java package methods.

For more information about how the OrchestrationXQueryFunctions are used in custom Component ID XQuery expressions and for more complicated custom group ID generation scenarios that use Orchestration XQueryFunction, see the following topics:

- Custom Order Component IDs Based on Hierarchy
- Custom Component IDs Based on Requested Delivery Date and Duration
- Custom Component IDs by Duration and Minimum Separation Duration
- Combining Order Item Hierarchy with Duration-Based Groupings

Custom Order Component IDs Based on Hierarchy

A more common scenario where custom order component IDs can be used is when you need additional groupings of order components at the granularity level. For example, three levels of decomposition from Function, System, to Bundle, results in the following component IDs:

- BillingFunction
- BillingFunction.BillingSystem
- BillingFunction.BillingSystem.Bundle

If you had order items in the Bundle order components that were part of different bundles that go to different the billing system, you would need to separate each order item bundle into different bundle order component. A component ID for such a scenario could look like this:

- For billing system 1: BillingFunction.BillingSystem.Bundle.2/BundleGranularity
- For billing system 2: BillingFunction.BillingSystem.Bundle.6/BundleGranularity

To create custom component IDs for this scenario, you could use the following order item properties:

- **typeCode**: This property specifies if the order line item is an offer, bundle, or product. This element also defines the product hierarchy of the order line items. For example:

  ```
  OFFER
     BUNDLE
        PRODUCT
  ```

- **lineId** and **parentLineId**: These properties specify the hierarchical relationship between the bundle and product order line items. You can create separate component IDs for every order item bundle and associate all product order items with their corresponding bundle component ID. To identify all ancestor order items that may be a bundle, you can use the XQuery **ancestors** function, as explained later.

For example, the following four order items include two bundles and two associated products. These order items have the following characteristics:

- **Order Item 1** includes:
  - **typeCode**: BUNDLE
  - **lineId**: 2
  - **parentLineId**: 1 (for example, an order item with an OFFER **typeCode**. This order item is not specified in this example).

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
```

```
 id="1234">
   <osm:name>FIXED BUNDLE - BUNDLE</osm:name>
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>BUNDLE</im:typeCode>
      <im:parentLineId>1</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Fixed Bundle</im:lineItemName>
      <im:lineId>2</im:lineId>
      <im:SiteID>5</im:SiteID>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Fixed Bundle Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Fixed</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

- Order Item 2 includes:
  - **typeCode**: PRODUCT
  - **lineId**: 3
  - **parentLineId**: 2 (This matches the **lineID** of order item 1 indicating that order item 1 is the parent of order item 2).

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
 id="56789">
   <osm:name>FIXED CALLER ID - PRODUCT</osm:name>
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>PRODUCT</im:typeCode>
      <im:parentLineId>2</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Commercial Fixed Service [Add]</im:lineItemName>
      <im:lineId>5</im:lineId>
      <im:SiteID>7</im:SiteID>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Fixed Bundle Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Fixed</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

- Order Item 3 includes:
  - **typeCode**: BUNDLE
  - **lineId**: 6
  - **parentLineId**: 1 (This indicates that both order item 1 and order item 3 share the same parent).

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
 id="10111213">
```

```
   <osm:name>BroadBand BUNDLE - BUNDLE</osm:name>
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>BUNDLE</im:typeCode>
      <im:parentLineId>1</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Broadband Bundle</im:lineItemName>
      <im:lineId>6</im:lineId>
      <im:SiteID>5</im:SiteID>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Broadband Bundle Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Broadband</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

- Order Item 4 includes:
  - **typeCode**: PRODUCT
  - **lineId**: 7
  - **parentLineId**: 6 (This matches the **lineID** of order item 3 indicating that order item 3 is the parent of order item 4).

```
<osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
 id="14151617">
   <osm:name>BroadBand Service - PRODUCT</osm:name>
.....
   <osm:properties
    xmlns:im="http://oracle.communications.ordermanagement.unsupported.
    centralom">
      <im:typeCode>PRODUCT</im:typeCode>
      <im:parentLineId>6</im:parentLineId>
      <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
      <im:lineItemName>Fixed Bundle</im:lineItemName>
      <im:lineId>7</im:lineId>
      <im:SiteID>5</im:SiteID>
      <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
      <im:productClass>Broadband Bundle Class</im:productClass>
      <im:serviceId>552131313131</im:serviceId>
      <im:productSpec>Service.Broadband</im:productSpec>
      <im:lineItemPayload> [34 lines]
      <im:region>Sao Paulo</im:region>
   <osm:properties>
</osm:orderItem>
```

The customer order includes two bundles with two products. The hierarchy is:

```
Fixed Bundle - order item 2
     Fixed Caller ID - order item 5
Broadband Bundle - order item 6
     BroadBand Service - order item 7
```

To create the separate customized component IDs for the bundle order items 1 and 3, and include all their corresponding children order items you need to:

- Return a separate component ID for each BUNDLE **typeCode**. This causes BUNDLE order components to be generated.

- Ensure that the PRODUCT **typeCode** for that bundle are included in its parent order item.

To do so, the XQuery uses the **ancestors** function to find whether the order item has a BUNDLE **typeCode** or has a BUNDLE **typeCode** in one of its parent order items. If the order item is a bundle, then a OSM creates a component ID for the bundle. If the order item has a bundle in one of its parent order items, then OSM includes the order item in its parent order item component ID. The following example shows an XQuery that does this.

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace prop="http://
oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osmfn =
"java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQueryF
unctions";
(:    The following part of the XQuery identifies the order line hierarchy definition
and retrieve all of the predecessor order line items in the bundle:    :)
let $ancestors := osmfn:ancestors("CustomerOrderItemSpecification","default","http://
oracle.communications.ordermanagement.unsupported.centralom")

(:   The following part of the XQuery finds the BUNDLE order item and generates an ID
based on the bundle order item lineID:      :)
   return
   if (fn:exists($ancestors[osm:properties/prop:typeCode='BUNDLE']))
      then (
               concat($ancestors[osm:properties/prop:typeCode=('BUNDLE')]
               [1]/osm:properties/prop:lineId/text(),'/BundleGranularity')
          )
      else (
              'ALL_OFFERS_AND_NON_SERVICE_BILLING/BundleGranularity'
)
```

This XQuery finds the child order line items, finds their parent order line items, and creates a bundle order component for each of the bundle lines. The component IDs are:

- BillingFunction.BillingSystem.Bundle.2/BundleGranularity
- BillingFunction.BillingSystem.Bundle.6/BundleGranularity

In another example, there is one offer with two bundles and two products in each bundle. The following table shows the hierarchy of bundles and products. The component IDs use the line IDs of the two bundle items.

Line Number	Line Name	Line typeCode	Parent Line ID	Value to Use in Component ID
1	Triple Play	OFFER	-	-
2	Fixed Bundle	BUNDLE	1	2
2.1	Fixed Service	PRODUCT	2	2
2.2	Call Forwarding	PRODUCT	2	2
5	Broadband Bundle	BUNDLE	1	5
5.1	Broadband Service	PRODUCT	5	5
5.2	High-Speed Internet	PRODUCT	5	5

Custom Component IDs Based on Requested Delivery Date and Duration

In some scenarios, you may want to create custom Order Component IDs based on order item requested delivery date and duration. For example, the following custom component ID XQuery creates order component grouping based on the order item requested delivery dates:

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace prop="http://
oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osmfn =
"java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQueryF
unctions";
let $groupDuration := "P2D"
return
osmfn: getGroupIdByDateTime ($groupDuration)
```

The XQuery creates a new order component for an order item based on the order item's requested delivery date and includes all order items within this group that fall within two days of the first order item's requested delivery date in the group. The XQuery does the same thing for all other order items within the order.

The following table shows how five order items would be grouped given a custom Order Component ID XQuery that creates a new component IDs.

> **Note:**
>
> The group ID names are static with the first order component always called Group1 and the next Group2, and so on.

Order Item	Requested Delivery Date	Group ID
A	June 9, 2014	Group1
B	June 10, 2014	Group1
C	June 11, 2014	Group2
D	June 12, 2014	Group2
E	June 12, 2014	Group3

See "About Component Specification Custom Component ID XQuery Expressions" for more information about the context, prolog, and body of this XQuery. See "OSM XQuery Functions" for more information about the **OrchestrationXQueryFunctions** class.

Custom Component IDs by Duration and Minimum Separation Duration

You can specify a minimum duration separation value for order items that fall very close to a custom Order ID grouping based on order item requested delivery date and duration. For example, the following XQuery adds a minimum separation value of one day:

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace prop="http://
oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osmfn =
"java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQueryF
unctions";
let $groupDuration := "P2D"
let $minSeparationDuration := "P1D"
return
osmfn: getGroupIdByDateTime ($groupDuration, $minSeparationDuration)
```

All order item requested delivery dates that fall within one day of a two day grouping, would be included in the two day grouping.

The following table shows how the five order items would be grouped given a one day minimum separation duration.

Order Item	Requested Delivery Date	Group ID
A	June 9, 2014	Group1
B	June 10, 2014	Group1
C	June 11, 2014	Group1
D	June 12, 2014	Group2
E	June 12, 2014	Group2

See "About Component Specification Custom Component ID XQuery Expressions" for more information about the context, prolog, and body of this XQuery. See "OSM XQuery Functions" for more information about the **OrchestrationXQueryFunctions** class.

Combining Order Item Hierarchy with Duration-Based Groupings

You can combine the function to create custom Component IDs based on order item requested delivery date, duration, and minimum duration separation, or a combination of these functions with order component ID generation based on order item hierarchy. The following example creates separate component IDs for order items that, although they have the same requested delivery date, are part of different order item hierarchical groupings:

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace prop="http://
oracle.communications.ordermanagement.unsupported.centralom";
declare namespace osmfn =
"java:oracle.communications.ordermanagement.orchestration.generation.OrchestrationXQueryF
unctions";
let $groupDuration := "P2D"
let $minSeparationDuration := "P1D"
return
osmfn: getGroupIdByDateTime ($groupDuration, $minSeparationDuration)
let $rootAncestorID := osmfn:ancestors("eboLineItem", "default", "http://
xmlns.oracle.com/communications/ordermanagement")[fn:last()]/osm:properties/
prop:BaseLineId/text()
return fn:concat($rootAncestorId, '/', $groupId)
```

The following table shows how five hierarchically divided order items would be grouped given a one day minimum separation duration.

Order Item	Requested Delivery Date	Group ID	Component ID
A.1	June 9, 2014	Group1	A/Group1
A.1.1	June 11, 2014	Group1	A/Group1
A1.2	June 19, 2014	Group2	A/Group2
A.1.3	June 20, 2014	Group2	A/Group2
B.1	June 9, 2014	Group1	B/Group1
B.1.1	June 11, 2014	Group1	B/Group1
B.1.2	June 12, 2014	Group1	B/Group2

See "About Component Specification Custom Component ID XQuery Expressions" for more information about the context, prolog, and body of this XQuery. See "OSM XQuery Functions" for more information about the **OrchestrationXQueryFunctions** class.

About Component Specification Duration XQuery Expressions

This topic applies to the Order Component editor, **Duration** tab, Duration Expression area, **XQuery** subtab.

- Context: There is no input document for this expression.
- Prolog: There is no prolog required for this expression.
- Body: The XQuery body returns a duration value based on the XQuery you enter:

  `PyYmMdDThHmMsS`

  where

  - **P** begins the expression.
  - *y***Y** specifies the year.
  - *m***M** specifies the month.
  - *d***D** specifies the day.
  - **T** separates the parts of the expression indicating the date from the parts of the expression indicating the time.
  - *h***H** specifies the hour.
  - *m***M** specifies the minutes.
  - *s***S** specifies the seconds.

The following example is a hard-coded duration expression for seven hours:

```
PT7H0M0S
```

For more information about how OSM uses these fields to calculate order component durations, see *OSM Modeling Guide*.

About Orchestration Fulfillment Pattern Duration XQuery Expressions

This topic applies to the Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Duration** subtab, Duration Expression area, **XQuery** subtab. The functionality for this tab has been deprecated and is displayed to provide backward compatibility with older cartridges.

For the recommended functionality for configuring order component durations, see "About Orchestration Fulfillment Pattern Component Duration XQuery Expressions" and "About Component Specification Duration XQuery Expressions".

About Orchestration Fulfillment Pattern Component Duration XQuery Expressions

This topic applies to the Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Order Components** subtab, **Duration** subtab, Duration Expression area, **XQuery** subtab.

- Context: There is no input document for this expression.
- Prolog: There is no prolog required for this expression.
- Body: The XQuery body returns a duration value based on the XQuery you enter:

  `PyYmMdDThHmMsS`

where

- **P** begins the expression.
- *y***Y** specifies the year.
- *m***M** specifies the month.
- *d***D** specifies the day.
- **T** separates the parts of the expression indicating the date from the parts of the expression indicating the time.
- *h***H** specifies the hour.
- *m***M** specifies the minutes.
- *s***S** specifies the seconds.

The following example is a hard-coded duration expression for three hours:

```
PT3H0M0S
```

For more information about how OSM uses these fields to calculate order component durations, see *OSM Modeling Guide*.

Dependency XQuery Expressions

This topic includes information about Orchestration XQuery expressions related to orchestration dependencies:

- About Order Item Dependency Property Correlation XQuery Expressions
- About Wait Delay Duration XQuery Expressions
- About Wait Delay Date and Time XQuery Expressions
- About Order Data Change Wait Condition XQuery Expressions
- About Order Item Inter-Order Dependency XQuery Expressions

About Order Item Dependency Property Correlation XQuery Expressions

This topic describes how to use one of the following fields:

- Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Dependencies** tab, **Order Item Dependency** subtab, **XQuery** subtab for the **Property Correlation** selection
- Orchestration Dependency editor, **Order Item Dependencies** tab, **XQuery** subtab for the **Property Correlation** selection

to write an expression that specifies dependencies between different order items using order item properties. The Property Correlation XQuery has the same context, prolog, and body structure as the Orchestration Fulfillment Pattern editor, **Order Components** tab, **Order Item Association** subtab, **XQuery** subtab. See "About Associating Order Items Using Property Correlations XQuery Expressions" for more information.

The following example shows a dependency that requires provisioning of an Internet service before shipping a modem. This involves two order items: provision Internet service and ship modem. The correlating property is the order item ID.

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace im="http://sample.broadband";
let $bbProvision := osm:fromOrderComponent/osm:orderItem[osm:name="Internet Service"]
let $bbModem := osm:toOrderComponent/osm:orderItem[osm:name/text()='Broadband Modem'
```

```
   and osm:properties/im:SiteID/text() = $bbProvision/osm:properties/im:SiteID/text()]
return
   <osm:dependency fromOrderItemId='{$bbProvision/@id}' toOrderItemId='{$bbModem/@id}'/>
```

In this example:

- **$bbProvision** contains the broadband service order item in the blocking Provision order component.
- **$bbModem** is the broadband modem in the waiting Ship order component.
- The XQuery returns a dependency from the Internet Service order item to its associated Broadband Modem order item, identified by **$bbProvision/@id** and **$bbModem/@id**.

If the order item IDs are:

- $bbProvision/@id = 1301589468772
- $bbModem/@id = 1301589468785

Then the XQuery returns the following:

```
<osm:dependency fromOrderItemId='1301589468772' toOrderItemId='1301589468785'/>
```

About Wait Delay Duration XQuery Expressions

This topic describes how to use one of the following fields:

- Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Dependencies** subtab, **Wait Condition** subtab, Wait Delay area, Duration Expression area **XQuery** subtab for the **Duration** selection
- Orchestration Dependency editor, **Wait for Condition** tab, Wait Delay area, Duration Expression area **XQuery** subtab for the **Duration** selection

to write an expression that specifies the duration of delay, based on an order item property, before starting a waiting order component after all dependencies have been resolved.

- Context: The **Duration** XQuery input document is the entire set of order items included in the order contained in the **toOrderComponent** element. You can return the value of requestedDeliveryDate to help determine the wait delay duration. For example:

```
<toOrderComponent xmlns="">
 <osm:orderItem [35 lines]
 <osm:orderItem [37 lines]
 <osm:orderItem [42 lines]
 <osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model" id="5678">
 <osm:name>Broadband Bundle [Add]</osm:name>
.....
  <osm:properties xmlns:im="http://oracle.communications.ordermanagement.unsupported. centralom">
   <im:typeCode>PRODUCT</im:typeCode>
   <im:parentLineId>3</im:parentLineId>
   <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
   <im:lineItemName>Broadband Bundle [Add]</im:lineItemName>
   <im:lineId>4</im:lineId>
   <im:SiteID>10</im:SiteID>
   <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
   <im:productClass>Broadband Bundle Class</im:productClass>
   <im:serviceId>1112223333</im:serviceId>
   <im:productSpec>Broadband.Bundle</im:productSpec>
   <im:lineItemPayload> [34 lines]
   <im:region>Sao Paulo</im:region>
  <osm:properties>
```

```
    <osm:orderItem [57 lines]
    <osm:orderItem [57 lines]
    <osm:orderItem [42 lines]
    <osm:orderItem [37 lines]
    <osm:orderItem [37 lines]
    <osm:orderItem [57 lines]
   </toOrderComponent>
```

- Prolog: You can declare the order item namespace and the OSM namespace in the XQuery prolog. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="http://
  oracle.communications.ordermanagement.unsupported.centralom";
  ```

- Body: The XQuery body returns a duration value based on the requestedDeliveryDate order item property:

  ```
  let $mydate := osm:toOrderComponent[1]/osm:orderItem[1]/osm:properties[1]/
  *[namespace-uri()='http://
  oracle.communications.ordermanagement.unsupported.centralom' and local-
  name()='requestedDeliveryDate'][1]/text()
  return
  if (fn:current-dateTime()- xs:dateTime($mydate) < xs:dayTimeDuration('PT10H'))  then
      'PT10H'
  else
      'PT10M'
  return
  ```

  where

  - **osm:toOrderComponent**: Provides the entire set of order items included in the order.
  - **osm:orderItem**: These are the order items in the **toOrderComponent** category. The remainder of this expression identifies the namespace of the order item specification and returns the value of the requestedDeliveryDate element.
  - The **if** statement checks to see if the value of the requestedDeliveryDate is less than the hard-coded dayTimeDuration value. These values conform to the XSD duration data type.
  - The **then** statement returns 10 hours if the **if** statement evaluates to true.
  - The **else** statement return 10 months if the **if** statement evaluates to false.

The following example shows the sample XQuery to return a duration value.

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace im="http://oracle.communications.ordermanagement.unsupported.centralom";

let $mydate := osm:toOrderComponent[1]/osm:orderItem[1]/osm:properties[1]/*[namespace-uri()='http://
oracle.communications.ordermanagement.unsupported.centralom' and local-name()='requestedDeliveryDate']
[1]/text()
return
if (fn:current-dateTime()- xs:dateTime($mydate) < xs:dayTimeDuration('PT10H'))  then
    'PT10H'
else
    'PT10M'
```

About Wait Delay Date and Time XQuery Expressions

This topic describes how to use one of the following fields:

- Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Dependencies** subtab, **Wait Condition** subtab, Wait Delay area, Duration Expression area **XQuery** subtab for the **Date Time Expression** selection
- Orchestration Dependency editor, **Wait for Condition** tab, Wait Delay area, Duration Expression area **XQuery** subtab for the **Date Time Expression** selection

to write an expression that specifies the date and time, based on an order item property, for starting a waiting order component after all dependencies have been resolved.

- Context: The **Date Time Expression** XQuery input document is the entire set of order items included in the order contained in the **toOrderComponent** element. You can use the requestedDeliveryDate order item property to determine the date and time that the XQuery should start after all blocking items have resolved. For example:

```
<toOrderComponent xmlns="">
 <osm:orderItem [35 lines]
 <osm:orderItem [37 lines]
 <osm:orderItem [42 lines]
 <osm:orderItem
 xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model" id="5678">
 <osm:name>Broadband Bundle [Add]</osm:name>
.....
  <osm:properties xmlns:im="http://oracle.communications.ordermanagement.unsupported. centralom">
   <im:typeCode>PRODUCT</im:typeCode>
   <im:parentLineId>3</im:parentLineId>
   <im:requestedDeliveryDate>2013-06-31T12:00:00</im:requestedDeliveryDate>
   <im:lineItemName>Broadband Bundle [Add]</im:lineItemName>
   <im:lineId>4</im:lineId>
   <im:SiteID>10</im:SiteID>
   <im:ServiceActionCode>UPDATE</im:ServiceActionCode>
   <im:productClass>Broadband Bundle Class</im:productClass>
   <im:serviceId>1112223333</im:serviceId>
   <im:productSpec>Broadband.Bundle</im:productSpec>
   <im:lineItemPayload> [34 lines]
   <im:region>Sao Paulo</im:region>
  <osm:properties>
 <osm:orderItem [57 lines]
 <osm:orderItem [57 lines]
 <osm:orderItem [42 lines]
 <osm:orderItem [37 lines]
 <osm:orderItem [37 lines]
 <osm:orderItem [57 lines]
</toOrderComponent>
```

- Prolog: You can declare the order item namespace and the OSM namespace in the XQuery prolog. For example:

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace im="http://
oracle.communications.ordermanagement.unsupported.centralom";
```

- Body: The XQuery body returns a date and time value based on the requestedDeliveryDate order item property:

```
osm:toOrderComponent[1]/osm:orderItem[1]/osm:properties[1]/*[namespace-uri()='http://
oracle.communications.ordermanagement.unsupported.centralom' and local-
name()='requestedDeliveryDate'][1]/text()
```

osm:toOrderComponent: returns the entire set of order items included in the order and returns the requested delivery date of all order items for the wait delay date and time.

The following example shows the sample XQuery to return a date time value.

```
declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
declare namespace im="http://
oracle.communications.ordermanagement.unsupported.centralom";

osm:toOrderComponent[1]/osm:orderItem[1]/osm:properties[1]/*[namespace-uri()='http://
oracle.communications.ordermanagement.unsupported.centralom' and local-
name()='requestedDeliveryDate'][1]/text()
```

About Order Data Change Wait Condition XQuery Expressions

This topic describes how to use the Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Dependencies** subtab, **Wait Condition** subtab, Wait for Condition area, **XQuery** subtab for the **Data Change Notification** selection.

This topic describes how to use one of the following fields:

- Orchestration Fulfillment Pattern editor, **Orchestration Plan** tab, **Dependencies** subtab, **Wait Condition** subtab, Wait for Condition area, **XQuery** subtab for the **Data Change Notification** selection
- Orchestration Dependency editor, **Wait for Condition** tab, Wait for Condition area, **XQuery** subtab for the **Data Change Notification** selection

to write an expression that specifies a value that must exist in order item property (typically a blocking order item property) before a waiting order item starts.

- Context: The **Data Change Notification** XQuery input document is the task view task data that was changed using an update order transaction.
- Prolog: You can declare the $blockingIndexes variable in the XQuery prolog that contains an index of data element for all blocking order items. For example:

  ```
  declare variable $blockingIndexes as xs:integer* external;
  ```

- Body: The XQuery body returns a specific value and will wait until all blocking order items have the corresponding value and the XQuery returns true.

The following example shows the XQuery that evaluates the data change. The dependency is met when all blocking order items have reached a state of PROVISION STARTED.

```
(:   The $blockingIndexes variable contains data element indexes for all blocking order items:   :)
declare variable $blockingIndexes as xs:integer* external;
(:   Specify "PROVISION STARTED" as the data value that must be met:   :)
let $expectedMilestoneCode := "PROVISION STARTED"
(:   $milestoneValues contains a set of milestones for all blocking order items:   :)
let $milestoneValues :=
     /GetOrder.Response/_root/ControlData/Functions/ProvisioningFunction/orderItem/orderItemRef[
     fn:index-of($blockingIndexes, xs:integer(@referencedIndex)) !=
     0]/milestone[text() eq $expectedMilestoneCode]
(:   Return true only if all the milestones in ProvisioningFunction/orderItem/orderItemRef are PROVISION
STARTED:   :)
return fn:count($milestoneValues) eq fn:count($blockingIndexes)
```

The following example returns true when at least one blocking item is completed.

```
declare namespace oms="urn:com:metasolv:oms:xmlapi:1";
declare variable $blockingIndexes as xs:integer* external;
let $component := //ControlData/Functions/NetworkProvisionFunction
let $lineItem := $component/orderItem/orderItemRef[fn:index-of($blockingIndexes,
xs:integer(@referencedIndex)) != 0]
return
     if (fn:exists($lineItem))
     then
```

```
    let $statusValue := $lineItem/OrderItemStatus/text() = "completed"
    return
    if (fn:count($statusValue)>0)
    then
        fn:true()
    else
        fn:false()
else
    fn:false()
```

About Order Item Inter-Order Dependency XQuery Expressions

This topic describes how to use the Order Item Specification editor, **Order Item Dependency** tab, Order Item Selector area, **XQuery** tab to write an expression that creates dependencies between the order items on the follow-on order and the order items on the base order. It is the follow-on order that generates this dependency on the base order.

- Context: The **Order Item Selector** XQuery input document is typically an order item on a follow-on order (the waiting order).
- Prolog: You can declare the OSM namespace, the cartridge namespace for the target order (the base order), and the namespace of the query task that contains the order data you want to view. For example:

  ```
  declare namespace osm="http://xmlns.oracle.com/communications/ordermanagement/model";
  declare namespace im="CommunicationsSalesOrderFulfillmentPIP";
  declare namespace osmc="urn:oracle:names:ordermanagement:cartridge:
  CommunicationsSalesOrderFulfillmentPIP:1.0.0:view:CommunicationsSalesOrderQueryTask";
  ```

- Body: The CRM that sends the follow-on order must specify the reference number that uniquely identifies the base order and also the order item line ID of the blocking order item. You can define a variable (such as **$dependingLineId**) that extracts the dependent line ID from the order item context. For example:

  ```
  let $dependingLineId := fn:normalize-space(osm:properties/
  im:DependingSalesOrderBaseLineId)
  ```

  You can configure a web service data instance provider that runs a FindOrder web service that searches for orders based on the reference value (**osm:properties/prop:Ref/text()**) in the follow-on order that generates a FindOrder response that includes the order ID of the base order. See *OSM Developer's Guide* for more information about configuring web service data instance providers. For example:

```
<ord:FindOrder xmlns:ord="http://xmlns.oracle.com/communications/ordermanagement"
   xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model"
   xmlns:prop="http://oracle.communications.ordermanagement.unsup.centralom">
   <ord:ViewBy>
      <ord:AmendmentFilter>
         <ord:LevelOfDetail>AmendmentsSummary</ord:LevelOfDetail>
      </ord:AmendmentFilter>
      <ord:LifecycleEventFilter>
         <ord:RetrieveLifecycleEvents>false</ord:RetrieveLifecycleEvents>
      </ord:LifecycleEventFilter>
   </ord:ViewBy>
   <ord:SelectBy>
      <ord:Reference>{fn:normalize-space(osm:properties/prop:Ref/text())} </ord:Reference>
   </ord:SelectBy>
</ord:FindOrder>
```

This data instance provider returns the order ID of the base order which you can capture in an XQuery variable (such as **$parentOrderID**). You can use this variable in a data

instance provider that runs a GetOrder web service to obtain the order item details from the base order. For example, the following XQuery populates the GetOrder request message using the results from the "findOrder" data instance provider to provide the value for the order ID of the base order in the Order ID field:

```
<ord:GetOrder xmlns:ord="http://xmlns.oracle.com/communications/ordermanagement"
    xmlns:osm="http://xmlns.oracle.com/communications/ordermanagement/model">
    <ord:OrderId>{vf:instance("findOrder")/ord:Order[last()]/ord:Amendments/
ord:AmendedOrderSummary/ord:Id/text()}</ord:OrderId>
<ord:View>CommunicationsSalesOrderQueryTask</ord:View>
</ord:GetOrder>
```

This data instance provider returns all order item instances in the base order that you can then search through to find the blocking order item using the **$dependingLineId** variable. You can capture the results in an XQuery variable (such as **$parentOrderItemId**). For example:

```
let $parentOrderItemId :=fn:normalize-space(vf:instance("getOrder") /ord:Data/
osmc:_root/osmc:ControlData/osmc:OrderItem [osmc:BaseLineId=$dependingLineId]/@index)
```

The XQuery body returns the order ID of the base order and the order item property that specifies the blocking order item on the base order:

```
 <osm:dependency fromOrderId="{$parentOrderId}"
fromOrderItemId="{$parentOrderItemId}"/>
```

where

- **<osm:dependency fromOrderId**: Returns the base order ID.
- **fromOrderItemId**: Returns the blocking order item property value that controls the dependency. OSM internally monitors the blocking order item until it is no longer being processed by any order component on the base order.

The following example shows an XQuery for an inter-order dependency.

```
declare namespace ord="http://xmlns.oracle.com/communications/ordermanagement";
declare namespace im="CommunicationsSalesOrderFulfillmentPIP";
declare namespace osmc="urn:oracle:names:ordermanagement:cartridge:
CommunicationsSalesOrderFulfillmentPIP:1.0.0:view:CommunicationsSalesOrderQueryTask";
let $dependingLineId := fn:normalize-space(osm:properties /
im:DependingSalesOrderBaseLineId)
return
   if(fn:not($dependingLineId = ''))
   then
(: Use the data instance behavior "findOrder" to find the base order:   :)
      let $parentOrderId := fn:normalize-space(vf:instance("findOrder")/
ord:Order[last()]/ord:Amendments/ord:AmendedOrderSummary/ord:Id/text())
(: Use the data instance behavior "getOrder" to find the associated order item ID in the
base order: :)
      let $parentOrderItemId :=
         fn:normalize-space(vf:instance("getOrder")/ord:Data/
         osmc:_root/osmc:ControlData/osmc:OrderItem[osmc:BaseLineId=$dependingLineId]/
@index)
      return
         if(fn:not($parentOrderId = '') and fn:not($parentOrderItemId = ''))
      then
(: Return the dependency: :)
      <osm:dependency fromOrderId="{$parentOrderId}"
fromOrderItemId="{$parentOrderItemId}"/>
      else()
   else()
```

Order Transformation Manager XQuery Expressions

The following topics provide reference information about order transformation manager XQuery expressions:

- About Transformation Sequence XQuery Expressions
- About Mapping Rule XQuery Expressions
- Order Item Parameter Binding XQuery Expressions
- About Transformed Order Item Fulfillment State XQuery Expressions

About Transformation Sequence XQuery Expressions

When working with Transformation Sequence editor, see the following topics for information about defining XQuery expressions related to transformation sequences:

- About Order Item Context XQuery Expressions
- About Related Order Item Selector XQuery Expressions
- About Stage Condition XQuery Expressions

About Order Item Context XQuery Expressions

This topic describes how to use the Transformation Sequence editor, **Dependencies** tab, **Order Item Context** subtab, Expression area, **XQuery** subtab to write an expression that defines the context order items for the order transformation. To see the **Order Item Context** subtab, you must select a transformation stage in the tree on the Dependencies tab.

- Context: The input document is the complete set of source order items.
- Prolog: You can declare the order item namespace in the XQuery prolog. For example:

```
declare namespace prop='http://oracle.communications.centralom';
```

- Body: The XQuery body returns the source order items that should be considered the context for the transformation stage.

The following example shows an XQuery expression for selecting an order item context.

```
declare namespace prop='http://oracle.communications.centralom';
osm:orderItem[osm:properties/prop:serviceIntance = 'Y']
```

About Related Order Item Selector XQuery Expressions

This topic describes how to use the Transformation Sequence editor, **Dependencies** tab, **Related Order Item Selector** subtab, Expression area, **XQuery** subtab to write an expression that defines the related order items for a particular context order item. To see the **Related Order Item Selector** subtab, you must select a transformation stage in the tree on the Dependencies tab.

- Context: The input document is a context order item.
- Prolog: You can declare the order item namespace and the namespace for the order transformation manager functions in the XQuery prolog. For example:

```
declare namespace prop='http://oracle.communications.broadband;
declare namespace
```

```
    otmfn="java:oracle.communications.ordermanagement.orchestration.transformation.XQuery
    Functions.";
```

- Body: The XQuery body returns the source order items related to the context order items.

The following example shows an XQuery expression that returns sibling order items as related order items to the order transformation.

```
    declare namespace prop='http://oracle.communications.broadband;
declare namespace
otmfn="java:oracle.communications.ordermanagement.orchestration.transformation.XQueryFunc
tions.";
    let $siblings := otmfn:siblings (., '{http://
oracle.communications.broadband}default')
    return $siblings[! fn:exists(osm:properties[prop:serviceInstance = 'Y'])]
```

For more information about the **transformation.XQueryFunctions** class, install the OSM SDK and extract the OSM Javadocs from the *OSM_home*/**SDK/osm7.***w.x.y.z*-**javadocs.zip** file (where *OSM_home* is the directory in which the OSM software is installed and *w.x.y.z* represents the specific version numbers for OSM). See *OSM Installation Guide* for more information about installing the OSM SDK.

About Stage Condition XQuery Expressions

This topic describes how to use the Transformation Sequence editor, **Dependencies** tab, **Stage Condition** subtab, Expression area, **XQuery** subtab to write an expression that determines whether a particular transformation stage should be run. To see the **Stage Condition** subtab, you must select a transformation stage in the tree on the Dependencies tab.

- Context: The input document is the complete set of target order items.
- Prolog: You can declare the order item property and parameter namespaces in the XQuery prolog. For example:

```
    declare namespace prop='http://oracle.communications.broadband';

declare namespace parm='http://oracle.communications.broadband';
```

- Body: The XQuery body returns a Boolean, with true meaning that the transformation stage should be run and false meaning that the transformation stage should not be run.

The following example shows an XQuery expression that returns true if certain parameters have not been defined, and false if the parameters are already defined.

```
declare namespace prop='http://oracle.communications.cso';
declare namespace parm='http://oracle.communications.broadband';
not(fn:exists(osm:properties/prop:Parameters[fn:exists(parm:uploadSpeed) and fn:exists
(parm:downloadSpeed)]))
```

About Mapping Rule XQuery Expressions

When working with Mapping Rule editor, see the following topics for information about defining XQuery expressions related to order decomposition:

- About Mapping Condition XQuery Expressions
- About Action Mapping XQuery Expressions
- About Entity-to-Entity Advanced Mapping XQuery Expressions
- About Entity-to-Data-Element Advanced Mapping XQuery Expressions

- About Data-Element-to-Data-Element Advanced Mapping XQuery Expressions
- About Reverse Mapping XQuery Expressions
- About Multi-Instance XQuery Expressions

About Mapping Condition XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Condition** subtab, Expressions area, **XQuery** subtab to write an expression that defines a condition that must be satisfied to apply this mapping.

- Context: The input document is a target order item.
- Prolog: You can declare the order item namespace in the XQuery prolog. For example:

```
declare namespace prop='http://oracle.communications.broadband';
```

- Body: The XQuery body returns a Boolean, with true meaning that the mapping rule should be run and false meaning that the mapping rule should not be run.

The following example shows an XQuery expression that will process the rule only if the target action is **None**.

```
declare namespace prop='http://oracle.communications.cso';
osm:properties/prop:Action/text() = 'None'
```

About Action Mapping XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Actions** subtab Action Mappings area, **XQuery** subtab to write an expression that defines the mapping for an action code for a particular mapping rule. To access this field, you must deselect **Use Relationship Action Map** and select the **Advanced** option.

- Context: The input document is a source order item.
- Prolog: You can declare the following variables within the prolog to determine the action code.
  - You can declare **$sourceValue** to access the action code of the source order item. This is the **Order Item Action** property value for the source order item.
  - You can declare **$currentTargetValue** to access the action code of the target order item. This is the **Order Item Action** property value for the target order item.
- Body: The XQuery body returns an action code, or returns () to leave the current value unchanged.

The following example shows an XQuery expression that returns the source action code if the target action code is not already set and otherwise leaves the target action code unchanged.

```
    declare $sourceValue external;
declare $currentTargetValue external;
if (! fn:empty($currentTargetValue))
    $sourceValue
else
   ()
```

About Entity-to-Entity Advanced Mapping XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Mapping** subtab, Mapping Rule Item area, **XQuery** subtab to write an expression that defines an advanced

mapping between two entities. This field is displayed when you select the target of an entity-to-entity mapping. This is the only type of mapping available for entity-to-entity mapping.

- Context: The input document is a source order item.
- Prolog: You can declare any namespaces needed to construct the target property (or properties) in the XQuery prolog. For example:

```
declare namespace prop='http://oracle.communications.cso;
```

- Body: The XQuery body returns a list of order item properties to be set on the target order item. If the property already exists on the target order item, it will be overwritten by the value returned from this XQuery expression.

The following example shows an XQuery expression that returns the structured Parameters property for the target order item.

```
declare namespace prop='http://oracle.communications.cso;
<prop:Parameters xmlns:param="http://oracle.communications.broadband">
   <parm:AAAAccount>Account1</parm:AAAAccount>
   <parm:DownloadSpeed>6</parm:DownloadSpeed>
   <parm:UploadSpeed>0.6</parm:UploadSpeed>
   <parm:MAC/>
   <parm:Brand>Siemens</param:Brand>
   <parm:Model>4200</parm:Model>
   <parm:Firewall>Y</parm:Firewall>
</prop:Parameters>
```

About Entity-to-Data-Element Advanced Mapping XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Mapping** subtab Mapping Rule Item area, **XQuery** subtab to write an expression that defines an advanced mapping between an entity and a data element. This field is displayed when you select the target of an entity-to-data-element mapping and select the **Advanced** option in the **Mapping Rule Item** topic.

- Context: The input document is a source order item.
- Prolog: There is no prolog for this XQuery.
- Body: The XQuery body returns a data element value or returns () to leave the current value unchanged.

The following example shows an XQuery expression that returns "Y" if a particular parameter exists, and () if it does not exist.

```
if fn:exists(vf:instance("checkMe")/somevalue)
   "Y"
else
   ()
```

About Data-Element-to-Data-Element Advanced Mapping XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Mapping** subtab **Configuration** subtab, **XQuery** subtab to write an expression that defines an advanced mapping between two data elements. This field is displayed when you select the target of a data-element-to-data-element mapping and select the **Advanced** option in the **Mapping Rule Item** topic.

- Context: The input document is a source order item during normal transformation. If invoked during forward data propagation, the input document is empty.
- Prolog: You can declare the order item namespace in the XQuery prolog. For example:

```
declare namespace prop='http://oracle.communications.centralom';
```

You can also declare the following variable within the prolog to determine the action code.

  - You can declare **$value** to contain the values of the target data elements.

- Body: The XQuery body returns one or more data element values or returns () to leave the current value unchanged.

The following example shows an XQuery expression that returns the target value of a data element based on the value of the source data element.

```
declare variable $value external;
if (fn:empty($value)) then ('unknown') else (fn:concat('Loc: ', $value))
```

The following example shows an XQuery expression that returns the target value of a data element based on characteristics of the source order item.

```
declare namespace prop='http://oracle.communications.centralom';
if (fn:exists(osm:properties/prop:ServicePoint/text()))
then (fn:concat('Loc: ', fn:normalize-space(osm:properties/prop:ServicePoint/string())))
else ('unknown')
```

About Reverse Mapping XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Mapping** subtab, **Bi-Directional Mapping** subtab, **XQuery** subtab to write an expression that defines an advanced mapping between two data elements. This field is displayed when you select the target of a data element-to-data element mapping and select the **Advanced** option in the **Mapping Rule Item** topic, if **Supports Bi-Directional Mapping** is selected in the **Details** subtab of the **Mapping** tab for the selected mapping.

- Context: The input document is empty.
- Prolog: You can declare the following variables within the prolog to determine the action code.
  - You can declare **$value** to access the updated target value.
- Body: The XQuery body returns the updated source value.

The following example shows an XQuery expression that returns () if the return value is unknown and otherwise returns the updated value.

```
declare variable $value external;
if ('unknown' = $value) then() else (fn:substring($value, 5))
```

About Multi-Instance XQuery Expressions

This topic describes how to use the Mapping Rule editor, **Mapping** tab, **Mapping** subtab, **Multi-Instance Expression** subtab, **XQuery** subtab to write an expression that defines key mapping for a multi-instance structure. This field is displayed when you select the target of a data element-to-data element mapping and select the **Advanced** option in the **Mapping Rule Item** topic, if the target data element is a member of a multi-instance structure.

- Context: The input document is a source order item.
- Prolog: You can declare the order item namespace in the XQuery prolog. For example:

  ```
  declare namespace prop='http://oracle.communications.broadband';
  ```

- Body: The XQuery body returns a key value that identifies a source order item instance.

The following example shows an XQuery expression that returns the concatenation of two source order item properties for the key value.

```
fn:concat(prop:areaCode, '-', prop:localNumber)
```

About Transformed Order Item Fulfillment State XQuery Expressions

This topic describes how to use the Transformed Order Item Fulfillment State Composition Rule Set editor, **Composition Rules** tab, **Source Order Item** subtab, **XQuery** field to write an expression that defines the conceptual model entities that should be present if the condition is to be evaluated. This field is only available when you have a condition selected in the tree in the tab, and you have selected the **Advanced** option on the subtab.

- Context: The input document is the order.
- Prolog: You can declare **$orderItemIndex** to access the index of the order item being considered.
- Body: The body of the XQuery will return a Boolean value indicating whether the current rule should be used to calculate the fulfillment state.

The following example shows an XQuery expression that returns true if a particular order item property has a specific value.

```
declare variable $orderItemIndex external;

let $orderData := fn:root(.)/GetOrder.Response
let $orderItem := $orderData/_root/ControlData/OrderItem[@index=$orderItemIndex]
return
    if (fn:exists($orderItem) and fn:data($orderItem/AnyProperties) = 'ABC')
    then fn:true()
    else fn:false()
```

Order Item Parameter Binding XQuery Expressions

This topic discusses best practices for writing an order item parameter binding XQuery. It also provides sample XQuery code that can be used in your cartridges.

> **Note:**
>
> Order Item Parameter Binding applies to product, service, and resource specifications equally.

- XQuery Context:

  The order item is passed into the XQuery as the context. As an example from the XQuery below:

  ```
  let $lineItem := .
  ```

- Body:

  The body of the XQuery should return a node set of elements that correspond to the conceptual model entity data elements.

OSM best practice suggests the use of two XQuery files to do the order item parameter binding.

A main XQuery file which is specified in the binding expression on the **Order Item Parameter Binding Editor** (**ParameterBindings Tab**). This XQuery is generic in nature and can be used across many solutions.

The second is an XQuery library module that is tied to the incoming order type. This library module is imported by the main XQuery and handles order specific processing.

The main XQuery uses an OSM model variable to know what module file to use. You will need to define an OSM model variable ***BINDING_MODULE*** to point to the location of the module that understands the input order. You can see an example of the main XQuery below:

```
(: This binding module is aware of the incoming order structure so that it can parse the
incoming payload as needed :)
 import module namespace bindingModule = "http://oracle.communications.orchestration.com/
parameterBindingModule" at "%{BINDING_MODULE}";


(: This variable defines the input document of the incoming request. To maintain
backward compatibility :)
declare variable $inputDoc as document-node() external;

(:Product Type will be populated by server runtime with setting from parameter binding
entity. To maintain backward compatibility :)
declare variable $parameterKeyName as xs:string external;

(: bindingMap will be populated by server runtime with the correct mapping content for
the product specification :)
declare variable $bindingMap as document-node() external;


declare function local:mapParameter(
    $parameterMap as node(),
    $incomingName as xs:string,
    $incomingValue as xs:string) as element()*
{

    (: Find the conceptual model entry in the map that has a key matching the incoming
name :)
    let $matchedAttribute := $parameterMap/attributes/attribute[key = $incomingName]

    let $conceptualModelUri := $parameterMap/conceptualModelURI
    let $conceptualModelName := $matchedAttribute/value/text()



    let $result :=
        if (fn:exists($conceptualModelName))
        then
        (
            element {QName($conceptualModelUri, $conceptualModelName)}{
                $incomingValue
            }
        )
        else ()
    return
    (
        $result
    )
 };


let $lineItem := .
```

```
let $parameterList := bindingModule:getIncomingParameters($lineItem)

return
(
    for $parameter in $parameterList
    return
    (
        let $incomingName := bindingModule:getIncomingName($parameter)
        let $incomingValue := bindingModule:getIncomingValue($parameter)

        return
            local:mapParameter($bindingMap, $incomingName, $incomingValue)
    )
)
```

In the main XQuery, when the OSM target server version is 7.5.0, then the OSM server will populate the variable ***$bindingMap*** with the content of the correct XML binding file that is generated by Design Studio.

The following example is a TMF622 BindingModule XQuery. It provides functions to retrieve the parameters and parameter values from a TMF 622 order item. Both the namespace and the location of the parameters are specific to a TMF 622 order:

```
(: OSM namespaces :)
declare namespace oms="urn:com:metasolv:oms:xmlapi:1";

(: Incoming order namespace as supplied by OSM Gateway :)
declare namespace tmf="http://oracle.communications.orchestration.com/tmf-api/
productOrderingManagement/%{TMF622_VERSION}/productOrder/inputMessage";

(:~
 : Get the product characteristics for a line item in the 622 product order structure
 :
 : @param $orderItem order line item context.
 : @return XML elements of the product characteristics for the given product order line
item.
 :)
declare function bindingModule:getIncomingParameters(
    $orderItem as element()) as element()*
{
    let $product := $orderItem/tmf:product
    return (
        $product/tmf:productCharacteristic
    )
};

(: Namespace specific method to retrieve the parameter(attribute) name from the
payload :)
declare function bindingModule:getIncomingName(
    $incomingAttribute as element()) as xs:string
{
    $incomingAttribute/tmf:name/text()
};


(: Namespace specific method to retrieve the parameter(attribute) value  from the
payload :)
declare function bindingModule:getIncomingValue(
    $incomingAttribute as element()) as xs:string
{
    let $value := $incomingAttribute/tmf:value/text()
```

```
    return(
        if(fn:exists($value))
        then
        (
            $value
        )
        else ""
    )
};
```

The following example is a TMF641 BindingModule XQuery. It provides functions to retrieve the parameters and parameter values from a TMF 641 order item. Both the namespace and the location of the parameters are specific to a TMF 641 order:

```
(: OSM namespaces :)
declare namespace oms="urn:com:metasolv:oms:xmlapi:1";

(: Incoming order namespace as supplied by OSM Gateway :)
declare namespace tmf="http://oracle.communications.orchestration.com/tmf-api/
serviceOrdering/%{TMF641_VERSION}/serviceOrder/inputMessage";

(:~
 : Get the service characteristics for a line item in the 641 service order structure
 :
 : @param $orderItem order line item context.
 : @return XML elements of the service characteristics for the given service order line
item.
 :)
declare function bindingModule:getIncomingParameters(
    $orderItem as element()) as element()*
{
    let $service := $orderItem/tmf:service
    return (
        $service/tmf:serviceCharacteristic
    )
};

(: Namespace specific method to retrieve the serviceCharacteristic name from the
payload :)
declare function bindingModule:getIncomingName(
    $incomingAttribute as element()) as xs:string
{
    $incomingAttribute/tmf:name/text()
};


(: Namespace specific method to retrieve the serviceCharacteristic value  from the
payload :)
declare function bindingModule:getIncomingValue(
    $incomingAttribute as element()) as xs:string
{
    let $value := $incomingAttribute/tmf:value/text()

    return(
        if(fn:exists($value))
        then
        (
            $value
        )
        else ""
    )
};
```

See the "How to Access to the Parameter Binding Document Using the new XQuery Context (Doc ID 3001948.1)" KM article on My Oracle Support for more information about accessing the parameter binding document using the new XQuery context.