Oracle Fusion Cloud SCM

Implementing Order Management

24B

Oracle Fusion Cloud SCM
Implementing Order Management

24B

F92830-01

Copyright © 2011, 2024, Oracle and/or its affiliates.

Author: carl casey

Contents

Get Help i

1 Hello 1

Overview 1
Details 29

2 Implement Business Process Flows 39

Order-to-Cash 39
Business-to-Business Messaging 61
Various Setups 71

3 Integrate 187

Introduction 187
Oracle Applications 207
Web Services 548
Upstream Source Systems 618
Downstream Fulfillment Systems 633
Cross-References 727
Drop Ship 738

4 Set Up Features 793

Approvals 793
Credit 844
Projects 895
Dual Units of Measure 913
Agreements 951
Trade Compliance 968
Coverages and Subscriptions 996
Customer Items 1034
Accounting 1043

Tax 1047
More 1097

5 Import and Transform 1105

Import 1105
Transform 1222

6 Orchestrate 1249

Overview 1249
Create 1267
Deploy 1291
Assign 1298
Pause 1304
Hold 1385
Fulfillment Task 1414

7 Use Business Rules 1443

Overview 1443
Guidelines 1452
Oracle Business Rules 1489
Visual Information Builder 1515

8 Process Sales Orders 1525

Control Behavior 1525
Manage Order Status 1551
Manage Order Values 1563
Constrain Actions 1605
Ship 1623
Manage Delay 1666
Change Orders 1671
Return Orders 1718

9 Configure-to-Order 1747

Overview 1747
Main Setup 1763

Web Services 1820
Other Setups 1852
More 1889

10 Extend 1905
Extensions 1905
Flexfields 2097

11 Email, Reports, and Attachments 2205
Email 2205
Reports 2208
Attachments 2219

12 Maintain and Troubleshoot 2227
Tools and Techniques 2227
Details 2250

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Use help icons to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 Hello

Overview

What's New in Implementing Order Management

Get a summary about content that's new or significantly revised in each update of Implementing Order Management.

To get details about:

- New features in each update, see *Order Management in Oracle Cloud Release Readiness*.
- Known issues for each update, see *Oracle Supply Chain Management Cloud Functional Known Issues and Maintenance Packs (Doc ID 1563075.1)*.

Update 24B

Topic	Description
Cancel Backorders and Enforce Single Shipments	New. Set up Order Management so it automatically cancels backorders and enforces single shipments.
Overview of Setting Up Projects in Order Management	Revised. The disable behavior for the DOO_PRJ_FULFILLMENTLINE_UPDATE constraint applies only on sales orders that you submit after you disable the constraint.
Guidelines for Using Extensions to Get Data from Oracle Applications	Revised. We recommend you use REST API to access data through business objects. If you must use a public view object, then you must use only the public view objects that Oracle supports.
Call Web Services from Order Management Extensions	Revised. You must not register a connector that uses an Oracle internal web service or an Oracle host.
Overview of Setting Up Sales Agreements in Order Management	Revised. Order Management doesn't track the quantity or the amount that it consumes for a sales agreement line.
Manage Channel Programs on Sales Orders	Revised. Order Management doesn't apply incentives to option classes, or to included items that are part of an assemble-to-order or pick-to-order configured item, or part of a kit.
Manage Order Management Parameters	Revised. See the new Cancel Supply When Canceling Backorders parameter.

Update 24A

Topic	Description
Manage Processing Constraints for Drop Ship	New. Get the latest about how Order Management uses processing constraints to help keep your drop ship flows running smoothly.
Create Integrated Coverage Lines in Order Management	New. Create an order line that contains a coverage in Order Management, then integrate the line with Subscription Management.

Topic	Description
Create Coverages for Assets	New. Create an order line that contains a coverage that covers an asset in Order Management, then integrate the line with Subscription Management.
Renew Coverages and Subscriptions in Order Management	New. Use Order Management to renew your coverages and subscriptions.
Manage Order Management Parameters	Revised. See the Days to Retry Recovery for Business Events parameter.
Set Up Drop Ship in Order Management	Revised. See the revised Manage Suppliers and Supplier Sites subtopic.
Guidelines for Using Extensions to Get Data from Oracle Applications	Revised. See the Filter Your PVOs and the Specify the Path subtopics.
Entities and Attributes That You Can Use When You Integrate Order Management with Other Oracle Applications	Revised. See the new Attributes You Must Not Use subtopic.
Attributes That You Can Use With Order Management Extensions	Revised. Reflects the latest update.
Import and Fulfill Large Volumes of Sales Orders	Revised. • If it takes more than 5 minutes to import a large sales order through REST API, then you might encounter a timeout error, but the import will continue to run in the background. • Removed the Enable the Features subtopic because the features now come predefined as enabled.
Guidelines for Setting Up Configuration Models	Revised. Your hierarchy can include parents and children, but there are limits about what you can include in ATO items, PTO items, and kits.
Guidelines for Pausing Orchestration Processes	Revised. See the new Assign the Result subtopic.
Display Ship-to Addresses On Sales Orders	Revised. See the revised Set Default Value for Ship-to Address subtopic.
Overview of Using Extensible Flexfields in Order Management	Revised. See the revised Entities That You Can Use With Extensible Flexfields subtopic.
Keep Your Order Number When You Import	Revised. Has some new examples to help you decide whether to keep your the number from your source system.
Track Items as Assets in Order Management	Revised. If you use a serial, and if you set the Enable Asset Tracking attribute to Customer Asset, then Oracle Asset Management automatically creates an asset.
Use Order Profiles to Control Order Management Behavior	Revised. Get new details about the Truncate Attribute Values for Accounts Receivable profile.

Update 23D

Topic	Description
Use REST API to Update and Split Fulfillment Lines	New. Use the Sales Orders for Order Hub REST API to update or split fulfillment lines across sales orders without having to revise them.
Troubleshoot Problems with Configure-to-Order	Revised. If the same option item occurs more than one time in your model's hierarchy, then you must explicitly define each node.

Topic	Description
Guidelines for Using Web Services to Integrate Order Management	Revised. If you use Oracle Application Development Framework (ADF), then you must include the entire order line in your payload even if you don't modify any part of the line. However, if you use REST API or FBDI with REST API, then you can exclude that entire line from the payload.
Attributes That You Can Use with Web Services	Revised. Has new details about how to use the SubmitFlag attribute.
Use Order Management Extensions to Apply Holds	Revised. Before update 23D, you could apply a hold only on the order line. You can now apply a hold on an order header too.
Aggregate Requests That Order Management Sends to Your Fulfillment System	Revised. Use this topic only if you implemented Order Management on Update 23C or earlier.
Guidelines for Integrating Order Management with Transportation Management *Integrate Order Management with Transportation Management*	Revised. Changed DOO_TP_REQ_ACCEPT to DOO_TP_REQ_ACCEPTED. Also, use DOO_TP_REQ_ACCEPTED to mark the task as done.
Set Up Features, Manage Change, and More for Drop Ship	Revised. See the revised Aggregate Fulfillment Lines subtopic.
Use Order Profiles to Control Order Management Behavior	Revised. See the Aggregate According to Number of Order Lines That Changed parameter.
Guidelines for Setting Up Units of Measure	Revised. See the new Create an Interclass Conversion subtopic.
Use REST API to Apply and Release Holds	Revised. Has new details about setting up the profile option.
Set Up Sales Agreements in Order Management	Revised. If you create more than one contract line for the same item, then make sure you use a different UOM on each line.
Manage Order Management Parameters	Revised. • We increased the default value of the Number of Processes for Order Import parameter from 1 to 4. This will help to make your import more efficient. • If you set the Use Price on Imported Orders That Oracle Fusion Already Priced parameter to Yes, then make sure your import payload includes the charge details.
Overview of Integrating Order Management with Subscription Management *Integrate Order Management with Subscription Management*	Revised. You no longer have to import a subscription. You can now also add one directly in the Order Management work area.
Use Web Services to Import Orders	Revised. You can't use the Order Import web service to update the quantity on a split fulfillment line, but you can use REST API to do this.
Privileges That You Need to Implement Order Management	Revised. See the new Identify the Privileges That You Need subtopic.
Troubleshoot Problems in Downstream Fulfillment	Revised. See the new Deactivating Customer Data After You Submit a Sales Order subtopic.
Freeze Price on Sales Orders	Revised. See the new Attributes That You Must Include When You Freeze Price subtopic.

Update 23C

Topic	Description
Import and Fulfill Large Volumes of Sales Orders	New. Improve the performance of your order-to-cash flow when you need to fulfill a large number of sales orders.
Integrate Recurring Billing Between Order Management and Accounts Receivable	New. Integrate recurring billing between Oracle Order Management and Oracle Accounts Receivable.
Use Compensation Patterns to Manage Purchase Orders	New. Add a compensation pattern to manage purchase orders when you don't use Oracle Global Order Promising.
Use REST API to Apply and Release Holds	New. Use the applyHold action and the releaseHold action on the Sales Orders for Order Hub REST API to apply and release holds on sales orders and fulfillment lines.
Troubleshoot Extensible Flexfields in Order Management	New. Use this topic to help you troubleshoot problems with your extensible flexfields.
Use Order Profiles to Control Order Management Behavior	Revised. See the new Number of Orchestration Processes to Start Concurrently for Large Sales Orders profile option.
Extend Order Lines	Revised. Specify the product ID instead of the product number when you create an order line.
Import Orders Into Order Management	Revised. See the new Include the Batch Name subtopic.
Use Proven Coding Techniques	Revised. Don't use an order management extension and a product transformation rule to add order lines to the same sales order.
Manage Lookups in Order Management	Revised. You can specify lookup values in your REST API payload.
Manage Order Attributes That Identify Change	Revised. You no longer need to restart the server when you modify an order attribute that identifies change.
Guidelines for Setting Up Holds on Sales Orders	Revised. Get new details about how to use holds with REST API, drop shipments, purchase orders, and other behavior.
Overview of Sending Notifications from Order Management to Other Systems	Revised. See the What Happens If I Need More Than One Event? subtopic.
Use Rate Plans with Your Subscriptions	Revised. Get important new details about implementing rate plans.

Update 23B

Topic	Description
Overview of Integrating Order Management with Subscription Management	New. Integrate Order Management with Subscription Management so you can fulfill coverages and subscriptions.
Temporarily Reserve Supply with Order Management	New. Temporarily reserve supply while shopping or quoting.
Import Rollup Charges for Configured Items	New. Specify whether you want Oracle Pricing to calculate the rolled up price when you import an assemble-to-order or pick-to-order item
Recognize Revenue on Different Dates	Revised. Make sure the custom fulfillment task that you use as the fulfillment completion step isn't a pause task because the pause task doesn't update the fulfillment date on the fulfillment line.
Use FBDI and REST API to Import Large Volumes of Sales Orders	Revised. See the new Guidelines subtopic.
Use the Import Sales Order Template	Revised. See the new Cancel Order Lines subtopic.

Topic	Description
Fix Errors in All Sales Orders	Revised. Get some updated information about how often to run the scheduled process, and how to set the scope, task type, and order orchestration function.
Use Order Profiles to Control Order Management Behavior	Revised. Get new details about how to set Total Number of Order Lines in Each Hour and Percent of Order Lines in Error in Each Hour.
Manage Sales Orders and Purchase Orders	Revised. See the new Create Purchase Orders subtopic.
Reduce Inventory Without Picking or Shipping	Revised. See the new Process Inventory Transaction Lines as Group subtopic.
Use Extensible Flexfields In Approval Rules	Revised. Revised some of the steps that involve using the Start Synchronization button.
Guidelines for Setting Up Orchestration Processes	Revised. See the new Use the DOO_ScheduleShipInvoice Orchestration Process subtopic.
Manage Processing Constraints	Revised. Get details about how to set the Invert Validation Rule Set attribute and the Group Number attribute.
Convert Shipment Costs to Freight Charges	Revised. Has new details about how to map attributes in your payload.
Guidelines for Processing Return Orders	Revised. Added another example to the Specify the originalOrderReference Entity subtopic.
Overview of Transformation Rules	Revised. Order Management doesn't run transformation rules or posttransformation rules with returns.
Manage Order Management Parameters	Revised. See the revised content for the Compare Change Order to Fulfillment Values parameter.
Create Your Own Task Type	Revised. Include your charges.
Set Up Approval Rules for Sales Orders	Revised. See the new Attributes That You Can Use with Approval Rules subtopic.
Various	Revised. Don't use the Earliest Acceptable Arrival Date attribute. Order Management doesn't support it.
Guidelines for Importing Pricing Data	Topic deleted. This content is now in *Use REST API to Manage Pricing Details*.

Update 23A

Topic	Description
Recognize Revenue on Different Dates	New. Order Management comes predefined to send the shipment date to Revenue Management. You can send a different date.
Constrain Updates According to Inventory Management's Status	New. Use a predefined validation rule set to constrain the updates that you can make on a fulfillment line according to the shipment line's status in Inventory Management.
Update Attributes on Split Order Lines for Partial Shipments	New. Update attributes on a split order line for a partial shipment.
Manage Your Watchlists	New. To improve performance, you can control the options that your users can select for the watchlist.
Manage Connector Details Between Order Management and Your Fulfillment System Send Notifications from Order Management to Other Systems	Revised. Clarified the WSDLs that you use.
Import Different Kinds of Data	Revised. See the new Display Line Numbers section.
Orchestration Process Status	Revised. You can't modify the status for a predefined shipment task or invoice task in some conditions.

Topic	Description
Manage Connector Details Between Order Management and Your Fulfillment System	Revised. See the revised Create More Than One Connector for Each Target section.
Attributes That You Can Use with Web Services	Revised. There are some new attributes that you can use: OverrideScheduleDateFlag, CreditChkAuthNumber, and CreditChkAuthExpDate.
Overview of Integrating Order Management with Accounts Receivable	Revised. New description and shipping example about how you can integrate Order Management with Accounts Receivable to calculate various charges, such as tax.
Use the Import Sales Order Template	Revised. If you import more than one sales order, then Order Management creates them in a sequence according to the Source Transaction Identifier attribute and the Source Transaction System attribute.
Guidelines for Setting Up Credit Check	Revised. You can't set up an orchestration process that skips credit check according to the type of sales order.
Update or Close Fulfillment Lines That Remain Open	Revised. • Make sure all instances of the Update or Close Sales Orders scheduled process are currently in the Succeeded status or the Error status. If you find a process that isn't in one of these statuses, such as Retrying, then you must wait for that instance to finish before you start a new instance. • If you encounter a problem, try running the process without a value in the Interval Hours parameter.
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. Don't enable the Translatable option.
More Set Up Details for Extensible Flexfields	Revised. New details about how to mask data.
Overview of Setting Up Sales Agreements in Order Management	Revised. You can't use a parent customer's agreement with a child customer.
Modify How Order Management Displays Attributes	Revised. Modify an attribute's width.
Integrate Your Own Order Line Attributes between Order Management and Oracle Receivables	Revised. We added another example.
Troubleshoot Order Management	Revised. If you don't want to require the primary salesperson, then you can disable an option.
Troubleshoot Problems With Order Import	Revised. • See how to get a specific version of a sales order • Get details about how to import the OriginalSourceOrderNumber attribute.
Specify the Orchestration Process When You Import	Revised. You must use only an assignment rule to assign the orchestration process for all order lines, or assign each line in your import payload when you import.
Overview of Importing Orders Into Order Management	Revised. Has a new Compare the Import Technologies section.
Use Order Management Extensions to Apply Holds	Revised. All new content.
Entities That You Can Use With Order Management Extensions	Revised. Removed the ShippableFlag attribute.
Guidelines for Setting Up Orchestration Processes	Revised. Don't modify a predefined orchestration process. Instead, create a copy of it, then modify the copy to meet your needs.

Topic	Description
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. Get details about the number of attributes, segments, and contexts that you can add.
Use Business Rules When You Can't Use Extensions	Revised. If you don't want to use a business rule or an extension to set the scheduled ship date, you can use a patch action on the salesOrdersForOrderHub REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.
Guidelines for Processing Return Orders	Revised. Learn about when you need to include the originalOrderReference entity.
Processing Constraints	Revised. See the new How Order Management Displays Your Constraint's Message subtopic.
Extend Credit Check	Revised. See the new Manage the Authorization Expiration Date subtopic.
Parallel Processing in Orchestration Processes Orchestration Subprocesses	Removed. You can no longer run orchestration processes in parallel or use subprocesses.
Manage Assignment Rules for Orchestration Processes	Removed. Use *Assign Orchestration Processes* instead.

Update 22D

Topic	Description
Extend Order Lines	New. Click the link to see how.
Guidelines for Setting Up Holds on Sales Orders	Revised. If you apply a hold on the order header, then Order Management will also apply the hold on all order lines in that order, including lines that are open, closed, or cancelled.
Convert Your Currency	Revised. If you don't set a value, then Order Management uses US Dollar, by default.
Manage Order Management Parameters	Revised. Use the Specify Default Values for Tax Determinants parameter to specify when to apply default values on attributes that affect tax.
Add Promotional Items That Reward Your Customers	Revised. Get new details about how to use business rules and extensions.
Guidelines for Processing Return Orders	Revised. See the new Assign Your Orchestration Process subtopic.
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. If you begin the name of your context or segment with a number, then Order Management automatically prefaces the name with an underscore (_) at run time.
Attributes That You Can Use When You Integrate Order Management with Accounts Receivable	Revised. New use cases and revised attributes.
Guidelines for Pausing Orchestration Processes	Revised. Get new troubleshooting tips.
Manage Order Management Parameters	Revised. See the Specify Default Values for Tax Determinants section.
Use the Import Sales Order Template	Revised. If the value that you import for any attribute starts with a space character and you want to keep the space, then you must enclose the value with double quotation marks (" ").
More Guidelines for Setting Up Approval	Revised. Use the Information Only action to keep someone who needs to know about the approval informed but doesn't actually approve.

Topic	Description
Create Your Own Task Type	Revised. You must use a web service to send a response to your fulfillment system. The payload that you use is different depending on the service that you use.
Guidelines for Managing Shipment Sets	Revised. Learn how to assign an orchestration process for a shipment set.

Update 22C

Topic	Description
Use Order Management Extensions to Apply Holds	New. Use the new applyHold method with the On Save or the On Start of Submission event to create a hold.
Use Extensible Flexfields In Approval Rules	New. Use an extensible flexfield as part of the condition in an approval rule.
Specify the Orchestration Process When You Import	New. Specify the orchestration process when you import through file-based data import (FBDI) or REST API instead of having to use a rule to assign the process.
Fix Errors in All Sales Orders	Revised. There are some new parameters on the scheduled process. Check it out!
Troubleshoot Order Management	Revised. Troubleshoot a problem that takes a long time to open or query on the Overview page in the Order Management work area.
Manage Order Management Parameters	Revised. In some cases, If you don't use jeopardy, then set Enable Orchestration Process Planning and Calculate Jeopardy to No.
Update or Close Fulfillment Lines That Remain Open	Revised. Make sure you have the correct credentials.
Manage Order Management Parameters	Revised. Get new details about how to set the Number of Processes for Order Import parameter.
Use Order Profiles to Control Order Management Behavior	Revised. Get details about the new Fulfillment Task Retries profile option.
Actions That You Can Set When Routing Requests to Fulfillment Systems	Revised. Get the latest details about these actions.
Freeze Price on Sales Orders	Revised. Don't modify freeze attributes after you import.
Import Different Kinds of Data	Revised. Make sure you prefix the line numbers in your source orders.
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. Learn how to search for extensible flexfields in the Order Management work area.
Use Order Profiles to Control Order Management Behavior	Revised. Use the new Populate Split Lines with Values from Original Line profile.

Update 22B

Topic	Description
Keep the Dates on Your Sales Orders and Purchase Orders Synchronized	New. Keep the dates on your sales orders in Order Management synchronized with the dates on your purchase orders in Procurement.
Use Order Management Extensions to Get Values from Extensible Flexfields	New. Use the getOrCreateContextRow method to add new data or update data through a flexfield. Use getContextRow to get data that you already added.

Topic	Description
Use FBDI and REST API to Import Large Volumes of Sales Orders *Use REST API to Import Large Volumes of Sales Orders* *Use Order Management Extensions to Import Large Volumes of Sales Orders*	New. Read these topics to learn how you can import a large volume of sales orders.
Use Different Line Types	New. Order Management comes predefined to send the ORA_BUY line type to Pricing Administration. If you use any other line type in your price list, then you must specify it on your service mapping.
Don't Refund Lines That You Return to Your Customer	New. Create a rule that doesn't refund your customer when you return the item to your customer.
Guidelines for Setting Up Orchestration Process Steps	Revised. Get details about the manual task.
Customer Items in Order Management	Revised. Get details about the validation that Order Management does.
Troubleshoot Problems in Downstream Fulfillment	Revised. Use an order management extension to update sales credits.
Entities That You Can Use With Order Management Extensions	Revised. Get details about the attributes that you can use with dual units of measure. Go to this topic, then search for the word **Secondary** to find them.
Use Reports and Analytics with Order Management	Revised. Learn how to add the primary salesperson to a report.
Set Up Business Units for Selling Profit Centers	Revised. Get details about how to import an assessable value.
Troubleshoot Problems With Order Import	Revised. Use REST API to import a cancel on an order line.
Actions That You Can Set When Routing Requests to Fulfillment Systems	Revised. Get details about changes to the Override Compensation Pattern, Override Operation, Set Acknowledgement Timeout, and Set Maximum Time to Wait Before Allowing Cancel actions.
Set Up Customer Items for Order Management	Revised. You must set up search for your customer item.

Update 22A

Topic	Description
Calculate Credit for Referenced Returns	New. Specify whether to calculate credit according to the secondary quantity that you actually receive or to the quantity that your customer requests when they place a return order.
Integrate Order Management with Accounts Receivable When You Use Financial Orchestration	New. If you use Financial Orchestration, then you can use this topic to set up your integration.
Display Ship-to Addresses On Sales Orders	Revised. Get new details about how to use a web service to set ship-to addresses.
Set Up Customer Items for Order Management	Revised. If you update the Trading Partner Item attribute in Product Information Management, then Order Management will update the Customer Item attribute on the sales order.

Topic	Description
Manage Order Management Parameters	Revised. Use the new Response Time in Seconds for Copy Action Improve parameter to improve your user experience. Specify the number of seconds to wait before allowing the user to navigate away from the order while Order Management creates a copy of it.
Update or Close Fulfillment Lines That Remain Open	Revised. Get details about parameter names that have changed and combinations of parameters that you can use.
Use Order Profiles to Control Order Management Behavior	Revised. Use the new Map Sales Credit from Parent to Child profile to map the sales credit on the parent configured item to the sales credit on all child configure options.
Collect Source System Data	Revised. You must set the Planning Method attribute to MPR Planning or MPS Planning.
Set Default Values for Shipping Methods	Revised. Consider how Promising uses the request type.
Set Up Item Substitution in Order Management	Revised. See the revised guidelines section.
Overview of Importing Orders Into Order Management	Revised. Get access to a new example template that you can use to import a return order.
Import Addresses into Order Management	Revised. There are two new values that you can use for the Address Use Type attribute.
Pause for Shipping	Revised. See the new Send Nonshippable Lines to Invoicing Without Waiting for Shipping to Ship Shippable Lines subtopic.
Use a Service Mapping to Integrate Order Management with Procurement	Revised. You must include the _Custom suffix to any entity that you add. Adding the Price, NegotiatedByPreparerFlag, and NoteToBuyer attributes is optional, they aren't required.
Import and Update Source Orders That Include Coverages and Subscriptions	Revised. If Order Management already split a line, then you can't use Oracle Application Development Framework (ADF) to add a coverage or subscription to the line.
Modify How Order Management Displays Attributes	Revised. Use line numbers to filter large sales orders.
Select Fulfillment Lines for Orchestration Process Steps	Revised. Don't attempt to ship a purchasable item.
Use Decision Tables and Bucket Sets in Business Rules	Revised. You must leave the View Object Name and Where Clause parameters empty, and you must set Refresh Collected Data to Yes when you run the Generate Bucket Sets scheduled process.
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. You can update an extensible flexfield on a closed sales order only in the Order Management work area. You can't through import or integration.
Guidelines for Setting Up Trade Compliance	Revised. The ORA_DOO_TRADE_COMPLIANCE_TYPE lookup and its codes come predefined. You can't modify it, and you can't create your own codes.
Troubleshoot Problems with Configure-to-Order	Revised. Fix problems that happen when you import a manual price adjustment for a configured item.
Set Up Item Substitution in Order Management	Revised. See the new guidelines.
Use Proven Coding Techniques	Revised. Make sure you finish processing the reply from the first call before you make the second call.
Guidelines for Setting Up Orchestration Process Steps	Revised. If you delete a step, and if the step references a status value from the orchestration process or the fulfillment line, then you must delete the status value before you delete the step.
Keep Your Order Number When You Import	Revised. You can't apply the Retain Sales Order Number profile option at the user level.

Update 21D

Topic	Description
Dual Units of Measure	New section. See how you can use the dual units of measure feature in your Order Management implementation to order, price, manufacture, receive, pick, pack, and ship an item in the primary UOM but price it in the secondary UOM.
Inventory	New section. Reduce the amount of on-hand quantity in your warehouse to fulfill sales orders that you submit in Order Management.
Migrate Orchestration Processes in Order Management	New topic, Migrate your orchestration processes from one environment to another, such as from your test environment to your production environment.
See How Long it Takes for Your Extension to Finish	New topic. Your order management extension might result in taking a long time to save or submit a sales order. Use this code to measure how much time its taking to finish running.
Manage Order Management Parameters	Revised topic. Use the Return Control to Users After Requests to Save Sales Orders parameter and improve your user experience.
Import Different Kinds of Data	Revised topic. Get details about how to import a configuration model.
Guidelines for Importing Pricing Data	Revised topic. Get details about how to import rate plans.
Create One Invoice for Sales Orders with Items That Can and Can't Ship	Revised topic. Wait until Order Management finishes shipping the shippable item, then run the Import AutoInvoice scheduled process.
Job Roles and Duty Roles That You Use to Implement Order Management	Revised topic. Limit access in Oracle Transactional Business Intelligence to the Order Management, Fulfillment Lines Real Time subject area.

Update 21C

Topic	Description
Customer Items in Order Management Set Up Customer Items for Order Management	New. Get the latest details about how to set up customer items.
Recover an Advance Shipment Notice	Revised. Get the latest details.
Use Decision Tables and Bucket Sets in Business Rules	Revised. Get new details about how to run the Generate Bucket Sets scheduled process.
Guidelines for Using Extensions to Get Data from Oracle Applications	Revised. Get new details about how to find the public view objects that you need.
Overview of Importing Orders into Order Management	Revised. The import technologies use slightly different names and values for some attributes.

Topic	Description
Import Different Kinds of Data	Revised. Get details about importing inventory data.

Update 21B

This help content is new or revised for the release.

Topic	Description
Use Credit Card Tokens to Improve Security	New. Use tokens to improve security for the credit cards that you use with Order Management.
Use Integration Algorithms to Implement Complex Logic	New. Use an integration algorithm when you must implement logic that you can't do through only a service mapping.
Import and Export Processing Constraints	New. Import and export processing constraints between environments, such as between your test environment and your production environment.
Integrate a Predefined Attribute Between Order Management and Accounts Receivable	New. Use a service mapping to map the Contract Start Date from the fulfillment line of a sales order, to the Start Date on the invoice line of an invoice in Accounts Receivable.
Integrate Your Own Header Attribute Between Order Management and Oracle Receivables	New. Use an extensible flexfield on the order header in Order Management to capture a value, then send that value to Oracle Receivables so Receivables can display it on an invoice.
Integrate Your Own Order Line Attributes between Order Management and Oracle Receivables	New. Use extensible flexfields and descriptive flexfields to integrate attributes on an order line in Order Management and send their values to Oracle Receivables.
Guidelines for Importing Pricing Data	New. Use guidelines to help you import pricing data.
Item Validation Organization	Revised. Get details about how to set the Item Validation Organization parameter.
Manage Order Management Parameters	Revised. Specify the number of seconds to wait before giving control back to the user for sales orders that have a large number of order lines.
Use Order Profiles to Control Order Management Behavior	Revised. Use the Hours to Wait Before Allowing Date Changes on Fulfillment Lines profile to specify the number of hours to wait.
Overview of Importing Orders into Order Management	Revised. We recommend that you use file-based data import (FBDI) instead of web services or REST API to import a large number of source orders.
Use SQL to Query Order Management Data	Revised. Find out which reservations have finished.

Topic	Description
Use Order Profiles to Control Order Management Behavior	Revised. Get details about the Send Fulfillment Details for Drop Shipments to Accounts Receivable order profile.
Set Up Features, Manage Change, and Do Other Setups	Revised. • Get details about the Cancel Fulfillment Line That Drop Ships processing constraint. • Consider how Global Order Promising uses time zones when it schedules a drop shipment.
Apply Logic in Business Rules	Revised. The priority determines the sequence that Order Management uses when it applies the rule.
Overview of Assigning Orchestration Processes	Revised. • If you create your own rule, then consider the predefined rules when you set priority. • Don't create an assignment rule that assigns a fulfillment line to an orchestration process when you revise a sales order.
Indicate an Ownership Change During Drop Ship	Revised. Learn how to recover an advance shipment notice.
Troubleshoot Problems With Order Import	Revised. • Order Management automatically creates the line number to make sure each line number is unique in the sales order. You can't modify this value. • Order import reports only one error at a time.
Track Items as Assets in Order Management	Revised. If you sell covered items and coverage items, and if you have a step that calls the DOO_ Subscription task, then you must call the DOO_AssetManagement task before you call the DOO_ Subscription task.
Import Different Kinds of Data	Revised. Import a source order and immediately close it.
Overview of Using Flexfields to Integrate Order Management with Other Oracle Applications	Revised. If your flexfield is on a fulfillment line, then use only one context for each entity.
Resume Paused Orchestration Processes	Revised. Examine some example payloads that release a pause task.
Troubleshoot Problems in Downstream Fulfillment	Revised. Fix a problem when the price on the order line is different than the price on the invoice.
Create a Return Reason	Revised. You can't disable or change the End Date on predefined lookup codes that have an ORA_ prefix,
Manage Connector Details Between Order Management and Your Fulfillment System	Revised. If you create more than one connector for the same target system, then you must use the same user name and password on each of those connectors.
Integrate Order Management with Global Trade Management	Revised. Use the DOO_MANAGE_ORCHESTRATION_ORDER_TRADE_COMPLIANCE_INTERFACE_WEB_SERVICE privilege when you set up the integration.

Topic	Description
Fix an Order Status That Doesn't Update to Shipped	Revised. What does Partially Closed mean, and how do I fix that?
Troubleshoot Problems with Web Services	Revised. Fix a performance problem when you use OrderFulfillmentResponseService with Integration Cloud Service.
Don't Use Order Promising to Schedule Fulfillment	Revised. Includes new details about the attributes you must set on the order line.

Update 21A

Topic	Description
Credit Cards	New section. Integrate Order Management with Oracle Payments so your users can select an existing credit card or add a new one to a sales order.
Example of Integrating Order Management with Other Oracle Applications	New. Use a demonstration to learn how to integrate Order Management with another Oracle Application.
Remove Infolets in the Order Management Work Area	New. Remove an infolet that displays on the Overview page of the Order Management work area.
Get and Display Approval Details for Sales Orders	New. Get approval details and use them for your specific needs, such as in an audit report.
Resolve Problems with the Warehouse Attribute on Sales Orders	New. Resolve a problem where the Warehouse attribute on the order line is empty or doesn't contain the warehouse that you need to pick.
Get Things Moving Again in Accounts Receivable	New. Manually run a scheduled process to resolve a sales order that's stuck in a particular status.
Transfer Inventory Between Business Units to Fulfill Sales Orders	New. Transfer inventory between business units when there isn't enough inventory to fulfill a sales order for one of them.
Set Up Currency Conversion	New. Set up Order Management so it converts currency when you sell into markets that use different currencies.
Guidelines for Processing Return Orders	Revised. Don't assign outbound lines to an orchestration process that processes returns.
More Code Examples for Order Management Extensions	Revised. If your billing application already authorized payment, then run an extension to prevent the user from editing payment attributes.
Use Order Profiles to Control Order Management Behavior	Revised.

Topic	Description
	• Use the Fulfillment Flows for Recalculating Dates profile to specify how to recalculate dates that change during fulfillment. • Use the aggregator profiles together. If you don't enable Aggregate According to Number of Order Lines That Changed, then Order Management ignores Aggregator Hold Timeout Period in Minutes. • Use the Respond Immediately on Start of Submission Request profile to specify when the Sales Order for Order Hub REST API sends a response to each request that it receives. • Use the Global Order Promising to Recalculate Dates in Order Management profile to manage shipment dates.
Import and Update Source Orders That Include Coverages or Subscriptions	Revised. If you import a billing plan, then make sure the PeriodicityCode attribute doesn't contain ONE TIME under the BillingPlans entity for the order line in your import payload.
Manage Order Management Parameters	Revised. Specify values for new parameters: Use Price on Imported Orders That Oracle Already Priced, and Use Pricing Algorithms to Calculate Totals for Sales Order.
Import Orders into Order Management	Revised. Use the Order Interface Status parameter to specify whether to process source orders that failed during a previous import.
Troubleshoot Problems With Order Import	Revised. Fix a sales order that's stuck in shipped status.
Import Different Kinds of Data	Revised. If you import a draft sales order, then do another import that changes the draft, then Order Management assigns a new order number to the draft. You must use the new order number in any subsequent updates or processing.
Transformation Rules	Revised. Order Management runs transformation rules differently depending on where you manage your sales orders.
Fulfillment Tasks	Revised. Don't use the DOO_Subscription task type.
Retain Sales Order Number During Import	Revised. Get details about how to make sure the order number in Order Management and the order number in Accounts Receivable are the same.
Indicate an Ownership Change During Drop Ship	Revised. Use the Manage Receiving Transactions scheduled process to receive and deliver a drop shipment in Oracle Receiving.
Troubleshoot Order Management	Revised. If you adjust price on a referenced return line for an outbound line that has recurring billing, and if the unit price on the return line is different from the unit price on the original line, then you must set the CancelReasonCode attribute to ORA_PRICE_CHANGE
Troubleshoot Problems with Web Services	Revised. Fix a problem when you have gaps in your order line numbers.
Troubleshoot Tax Set Up	Revised. Use the Assessable Value attribute to calculate tax for an order line that has a value of zero in the Amount attribute.
Example Web Service Payloads That Integrate Order Management	Revised. You can't include the Attachment entity in a stage order payload.

Topic	Description
Specify Transaction Types When Integrating Order Management with Oracle Receivables	Revised. Get details about how to set the transaction type before you transform.
Guidelines for Integrating Order Management with Transportation Management	Revised. Make sure the user has the privilege you need.
Freeze Price on Sales Orders	Revised. Learn now to freeze price on return orders.
Display Contacts on Sales Orders	Revised. Learn how to remove duplicate contacts that display in the list of values when you click the down arrow in the Contact attribute on the order header.
Create Different Types of Business Rules	Revised. If the order header and the order line contain the same attribute, such as the Freight Terms attribute, and if you need to transform the value on the line, then use a posttransformation rule.
Set Default Value for Contact on Order Header	Revised. Learn how to remove duplicate contacts from the Contact attribute on the order header.
Administer Email Messaging in Order Management	Revised. Contains a new guidelines subtopic.
Import Customer Items into Order Management	Revised. Clarified what a customer item is and how to import it.
Guidelines for Setting Up Configuration Models	Revised. If you import a revision that includes the entire component list for a configuration model, then you must make sure the sequence that you use in the revision is identical to the sequence you used in the original import.
Troubleshoot Problems in Downstream Fulfillment	Revised. • Troubleshoot an order line that's stuck in waiting status. • Troubleshoot a problem where the warehouse hasn't reserved inventory for your item.
Guidelines for Setting Up Orchestration Processes	Revised. If you copy or duplicate an orchestration process, then make sure you validate it and remove all validation errors before you create the copy or duplicate.
Guidelines for Setting Up Orchestration Process Steps	Revised. Use Advanced Mode when you add a business rule to the first step of a custom orchestration process.
Filter Lines In Your Extensions, Rules, and Constraints	Revised. If you set the nonKitModelFlag attribute to Y on a line-selection criteria, and if the order line contains a configuration model, and if the model contains a kit, then select the line and process it.
Import Tax on Sales Orders	Revised. Get details about how How Order Management integrates tax with Accounts Receivable.
Resume Paused Orchestration Processes	Revised. Make sure you have the privileges you need to release pause tasks.

Topic	Description
Manage Connector Details Between Order Management and Your Fulfillment System	Revised. If you attach a large number of documents to the sales order, then we recommend that you set Invocation Mode to Asynchronous Service.
Create Different Types of Business Rules	Revised. Don't create a routing rule that depends on a pause task and that sets the connector.
Guidelines for Setting Up Holds on Sales Orders	Revised. You can't use the web service to release a hold on a fulfillment line.

Update 20D

Topic	Description
Tax	New section. Learn how to set up and apply tax to sales orders.
Create Direct Links to Order Management Pages	New. Create a direct link from your application or an Oracle Application to a page in the Order Management work area.
Use Order Management to Fulfill Different Types of Sales Orders	New. Use Order Management to fulfill a range of order types.
Automatically Import Source Orders into Order Management	New. You can use the Order Import Template to manually import orders from your source system into Order Management. This topic describes how to do it automatically.
Troubleshoot Global Order Promising in Order Management	New. Get details about how to troubleshoot problems that including promising in your implementation.
Don't Use Order Promising to Schedule Fulfillment	New. Your implementation might not need to use Order Promising to schedule fulfillment. You can set up Order Management so it doesn't.
Create a Return Reason	New. Create a return reason so the Order Entry Specialist can select it when returning an order line.
Another Example of Importing Flexfield Data Through Web Services	New. Add an extensible flexfield to store a status on the order line, and import it through the Receive Order Request Service.
Example of Using an Extension to Get Data from Oracle Applications	New. Identify the public view object you need, then reference it in an order management extension.
Set Default Values for Attributes on Sales Orders	New. Create a business rule that sets the default value for an attribute on a sales order.
Use Different Attributes in the Condition and Do Simple Math	New. Create a pretransformation rule on a variety of attributes in the condition or action, and include a simple mathematical equation.
Set Values for Attributes That Depend On Each Other	New. Some attributes depend on each other. For example, if you need to use a shipment set, you must first enable the **`Shipset`** attribute.

Topic	Description
Set Default Values for Shipping Methods	New. You can specify the default value that Order Management uses to set the Shipping Method attribute when the Order Entry Specialist creates a sales order.
Set Warehouse According to the Value of Some Other Attribute	New. Set up Order Management so it sets the default value for the Warehouse attribute on the fulfillment line according to the value of the Business Unit attribute on the order header.
Use Business Rules When You Can't Use Extensions	New. Create a business rule when an Order Management extension doesn't meet your requirements.
Add Promotional Items That Reward Your Customers	New. Add an item to a sales order to reward customers who purchase larger quantities.
Set Attribute Values After You Transform Source Orders	New. Set the default value for an attribute after you transform the source order.
Set Promised Ship and Arrival Dates on Fulfillment Lines	New. Learn how to set the Promised Ship Date and the Promised Arrival Date.
Use Flexfields to Allow Users to Set Attribute Values	New. Allow your Order Entry Specialists to enter a value in a flexfield to indicate they need to change the value of an attribute.
Remove Records That Contain Data About Actions to Improve Performance	New. Remove records that contain data about actions that Order Management users have done while managing sales orders. Removing these records can improve performance.
Import Price Details	Revised. If you copy a sales order that has a configured item, and if the order also includes an order line that has a coverage for the configured item, then Order Management uses the pricing that you specify in the Pricing Administration work area for the coverage item even if you freeze pricing during import.
Set Up Coverage for Sales Orders	Revised. • You must map time units for your coverages and subscriptions. • You can't use an order management extension or a transformation rule to set shipment set the values on an order line that contains a coverage. • If you import a sales order, and if your import includes a shipment set on an order line that contains a coverage item, then the import ignores the value for the shipment set on the line.
Overview of Importing Orders into Order Management	Revised. If you must import binary data, then you must convert your binary data to text before you do the import, and you must use the base64 encoding scheme when do the conversion.
Set Up Orchestration Processes for Coverage Items	Revised. Use a pause task to make sure the coverage item and covered item go to billing at the same time.
Troubleshoot Problems With Order Import	Revised. If you import a referenced return order but Receivables doesn't create a credit memo for the return, then make sure the business unit on the order line that you import is the same as the business unit on the original order line.
Guidelines for Setting Up Approval	Revised. Use the Make Notifications Secure option to secure your notifications.

Topic	Description
Import Source Orders That Include Coverages or Subscriptions	Revised. Learn how to import coverages or subscriptions through FulfillmentResponseService.
Create One Invoice for Sales Orders with Items That Can and Can't Ship	Revised. Make sure you use square brackets to enclose the list when you set the GL Date.
Code Snippets for Order Management Extensions	Revised. Set the Requested Ship Date on the order line to the same value that the Requested Ship Date attribute contains on the order header.
Set Up Shipping Tolerances in Order Management	Revised. Set the Over Shipment Tolerance attribute to a positive number that's greater than zero. Set the Under Shipment Tolerance to a positive number that's greater than zero but less than 100.
Guidelines for Setting Up and Using Drop Ship	Revised. Get new details about predefined processing constraints. They're in the Manage Change subtopic.
Guidelines for Setting Up an Integration Algorithm	Revised. Make sure you can use Purchase Request Service to update the attributes that you map.
Guidelines for Using Extensions to Get Data from Oracle Applications	Revised. Get details about tables, views, and public view objects.
Troubleshoot Order Management	Revised. Troubleshoot an extension that doesn't run when you cancel a sales order.

Update 20C

Topic	Description
Identify Flexfield Contexts and Category Codes for Your Business Rules	New. Identify the flexfield contexts and category codes that you reference in an Oracle Business Rule.
Examples of Pausing Orchestration Processes	New. Peruse these examples to learn how you can use business rules to meet your business requirements.
Example of Integrating Order Management with an External Fulfillment System	New. Assume you need to integrate Order Management with your shipment system. Use this topic to learn how.
Get Help From My Oracle Support	New. Use the My Oracle Support website at support.oracle.com to troubleshoot a range of problems.
Include Price, Discounts, and Shipping Charges in Your Payloads	New. Make sure that the values you send in the payloads you use to integrate with various systems include accurate values for price, discounts, and shipping charges.
Cancel Backordered Quantity During Order Import	New. You can use the order import service to cancel a backordered line.

Topic	Description
Set Up Item Substitution in Order Management	New. Set up substitution in Order Management so you can use a substitute item to fulfill an order line when the preferred item is out of stock.
Overview of Integrating Order Management with Revenue Management	New. Automate how Revenue Management gets sales order data from Order Management so Revenue Management can recognize revenue for the sales order.
Manage Order Attributes That Identify Change	Revised. Restart the Oracle server when you do the Manage Order Attributes That Identify Change task.
Guidelines for Setting Up Order-to-Cash	Revised. Avoid a runtime error. Consider how processing constraints affect your setup.
How Configure-to-Order Works	Revised. You can't cancel a configure option in a pick-to-order item or kit when the shipment is out of proportion.
Use Integration Cloud Service with Order Management	Revised. Get details about articles on Oracle Help Center that describe other ways you can use Integration Cloud Service with Order Management.
Manage Processing Constraints	Revised. If you create your own processing constraint or modify a predefined one, and if you move to a new update, then you must generate packages immediately after you do the update.
Convert Shipment Costs to Freight Charges	Revised. Includes revised payloads and new diagrams to help you set up the connector.
Use Order Profiles to Control Order Management Behavior	Revised. Specify the number of minutes for the hold operation to wait before timing out while aggregating order lines.
Set Up Coverage for Sales Orders	Revised. Get details about how Coverage Start Date works.
Quick Start for Setting Up Order-to-Cash	Revised. Includes some revised details about setting up your URL and security.
Use Extensible Flexfields to Integrate Order Management with Other Applications	Revised. The attribute View Object Attribute is required for each mapping you create in this topic.
Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications	Revised. If you add a value in the Alias attribute, then make sure the value begins with an upper case character.
Troubleshoot Order Management	Revised. • If the Category area of the Edit Extensible Flexfield page doesn't have any rows, then run the Publish Extensible Flexfield Attributes scheduled process. • If a return line doesn't have a price, then promote your pricing algorithms. • If your return line is stuck in the Awaiting Receiving status, then run the Send Receipt Confirmation scheduled process. • If you encounter a currency conversion error, then you might need to set up a default value for the conversion rate type.

Topic	Description
	• If you can't set a value for the Profile Value attribute on the Manage Administrator Profile Values page, then you might need to collect data.
Allow Users to Return Items Without Original Sales Order	Revised. Get new details about how Pricing prices a return.
Example Web Service Payloads That Integrate Order Management	Revised. Get new example payloads for manual price adjustment.
Guidelines for Setting Up and Using Drop Ship	Revised. • The new Modify Extensible Flexfield when Line is in Requisition constraint prevents the Order Management user from modifying an extensible flexfield on the fulfillment line when the line is at the requisition stage in the procurement application. • If Order Management applies a hold on a sales order, then Procurement also applies a hold on the corresponding purchase order. However, a buyer can use the Purchase Orders work area to remove the hold.
Guidelines for Setting Up Holds on Sales Orders	Revised. You can now use a web service to apply or release a hold on a sales order that's in Draft status.
Overview of Using Extensible Flexfields in Order Management	Revised. If the order line that the fulfillment line references is closed, then the Order Entry Specialist can't update the value in the extensible flexfield on the fulfillment line.
Guidelines for Setting Up Extensible Flexfields in Order Management	Revised. Make sure the name for each segment or context you create uses the correct nomenclature.
Code Snippets for Order Management Extensions	Revised. Includes new snippets that you can use with your order management extensions.
Display Ship-to Addresses On Sales Orders	Revised. Set the default value for the Contact Method attribute on each sales order.
Guidelines for Importing Orders into Order Management	Revised. If you don't want to submit your imported order to fulfillment, but instead want to import it as a draft sales order, then set the Submit Flag attribute to N on the DOO_ORDER_HEADERS_ALL_INT worksheet.
How Drop Ship Works in Order Management	Revised. Procurement doesn't use the Product Category or Product Fiscal Classification to create a purchase order.

Update 20B

Topic	Description
Collect Planning Data for Order Management	New. Get details about how to collect planning data, including a complete list of data that you can collect.
Modify Report Templates	New. The Reports and Analytics work area comes predefined with several reports that you can use. You can modify a template to meet your needs.

Topic	Description
Import Shipping Method	New. If you import a shipping method through REST API or FBDI, you must make sure the method is valid.
Guidelines for Setting Up Units of Measure	New. Use guidelines to help you set up the units of measure that you use in Order Management.
Create One Invoice for Sales Orders with Items That Can and Can't Ship	New. Set up Order Management so it creates a single invoice when your sales order includes order lines that can ship and lines that can't.
Guidelines for Creating Business Rules	Revised. If your rule references an attribute that doesn't contain a value, and if your rule does a calculation that requires a value, then it might create an error or result in a null pointer exception at run time.
Guidelines for Managing Shipment Sets	Revised. Includes some new guidelines. Includes new details about importing a shipment set.
Use Extensions to Get Values for Return Orders	Revised. Get purchase order details from the order header and order line of the original order, then copy them to the return order and return order line.
Use Diagnostics to Troubleshoot Sales Orders	Revised. Use the Order Management Health Check test to scan through your database and identify problems across sales orders.
Drop Ship section	New and revised. Read the new topics and revised topics that clarify how to set up drop ship.
Guidelines for Importing Orders into Order Management	Revised. If you import an order, revise it, submit it, then import it again in a subsequent import, then you must import the most recent attribute values in the subsequent import.
Quick Start for Setting Up Order-to-Cash	Revised. You must deploy the predefined Update Shipping Request Validation processing constraint.
Set Up Messages in Order Management	Revised. Read up on some new details about how to set up messages.
Use SQL to Query Order Management Data	Revised. Find the reservations that have finished for an item in your sales order.
Overview of Setting Up Sales Agreements in Order Management	Revised. Learn about new date attributes you can use when you import or integrate an agreement.
Guidelines for Setting Up Approval	Revised. You can use the order type attribute on the order header in an approval rule.
Troubleshoot Order Management Setup	Revised. Write an order management extension that won't error out when your sales order doesn't use the English language. Examine predefined processing constraints to see if that's the cause of an error you're encountering.
Guidelines for Setting Up and Using Drop Ship	Revised. If the quantity on the advance shipment notice (ASN) exceeds the ordered quantity, then Oracle Receiving rejects the fulfillment request regardless of how you set the Over-Receipt Action attribute.

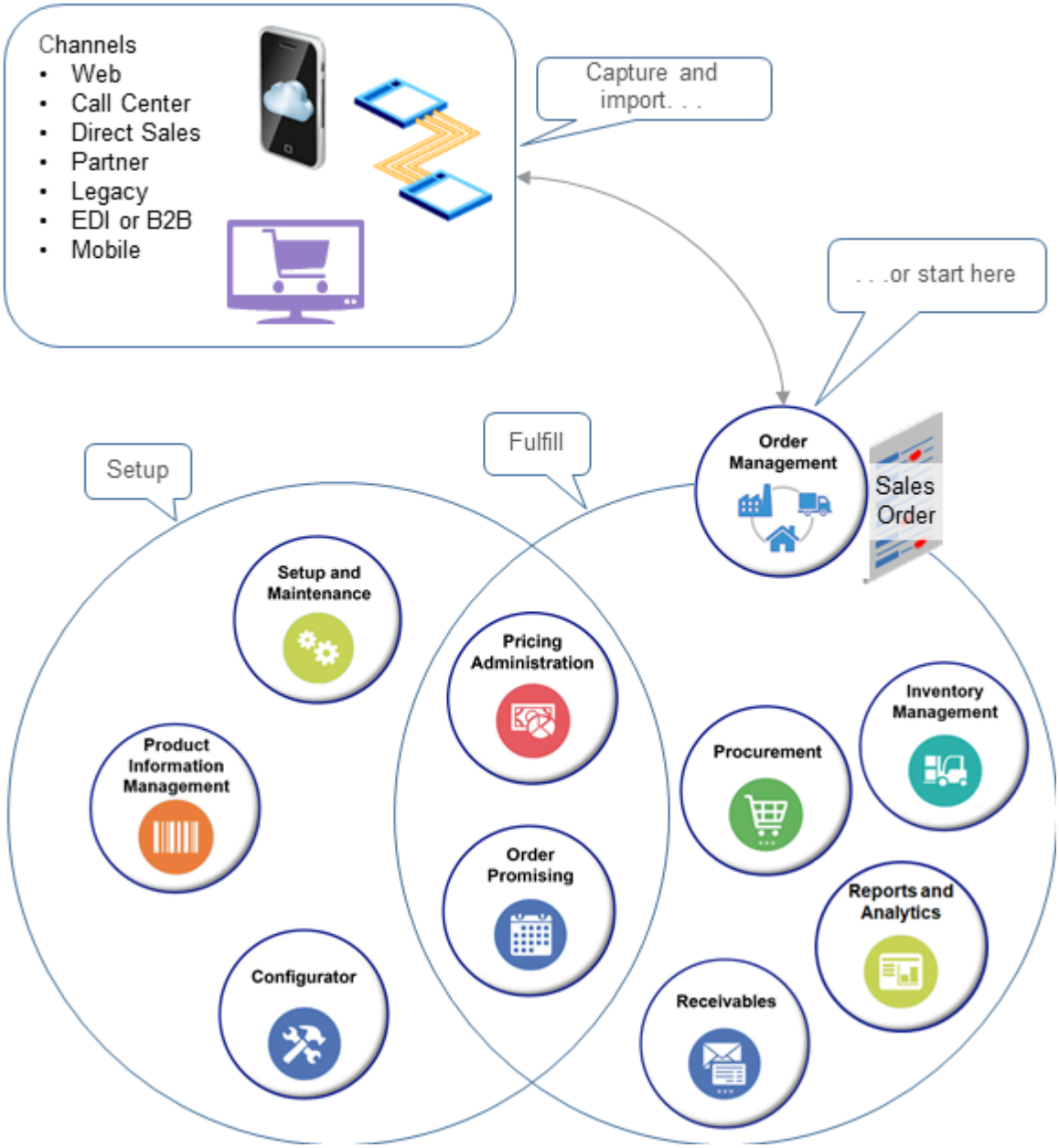
Topic	Description
Use Order Profiles to Control Order Management Behavior	Revised. If you use OPS as your source system in a channel, then you must set Retain Sales Order Number to N.
Allow Users to Return Items Without Original Sales Order	Revised. Requirements for what you can return depend on whether the item is pick-to-order or assemble-to-order.
Guidelines for Setting Up Orchestration Process Steps	Revised. Make sure you specify the value*What's New in Implementing Order Management* Canceled as an exit criteria status to exit the wait task in your orchestration process.
Copy Setups Between Instances of Order Management	Revised. If you migrate from a production environment to a test environment, and if your migration includes an orchestration process that's currently in progress, then the sales order that references the process will become stuck. You can't recover the stuck order.
Manage Order Management Parameters	Revised. Order Management doesn't use the value for any other attribute as the preparer. For example, it doesn't use the Created By attribute. It uses only the value you specify in Preparer for Procurement.
Example Web Service Payloads That Integrate Order Management	Revised. Use the ProjectDetail tag to include project details in your payload.

Overview of Order Management

Order Management is a supply chain management application that improves order fulfillment for your business processes. It includes predefined integrations, centrally managed orchestration policies, global availability, and fulfillment monitoring that can help increase customer satisfaction and profitability.

Capture Orders and Fulfill Them

Use Oracle Order Management to capture customer demand and fulfill sales orders.



Capture demand from channels in your order capture system and import them into Oracle Order Management:

- Web sales
- Call centers
- Direct sales
- Partners
- Legacy order capture systems and other systems
- Electronic data interchange or business-to-business flows
- Mobile platforms

You can also capture orders directly in Oracle's Order Management work area.

Use Oracle Applications and their work areas to set up your supply chain environment.

- Set up integrations, specify default values, specify application behavior, and do other setups in Setup and Maintenance.
- Set up your item in Product Information Management.
- Price the item in Pricing Administration.
- Specify how to determine availability for your item in Oracle Global Order Promising.
- Configure your configured items in Configurator.

Use Oracle applications and their work areas to orchestrate order fulfillment across sources, such as suppliers, finance, and distribution centers.

- Track and revise sales orders in Oracle Order Management.
- Procure the item that you will ship in Oracle Procurement.
- Reserve and track the item in Oracle Inventory Management.
- Get details about the item in Reports and Analytics.
- Receive the item into accounting in Oracle Receivables.

Manage Orders Across Channels

Order Management can support your global, order-to-cash process that uses more than one channel. Centralize and manage order capture across channels, do order promising, orchestrate fulfillment policies, monitor order status, and manage exceptions. Create and price each sales order directly in Order Management.

- Create sales orders directly in Order Management. Enter details for a new sales order, revise a sales order, modify order lines, view change history, place a sales order on hold, or cancel a sales order. Create a return order for an existing sales order.
- Use a common definition of sales orders across channels so you can view and search them in a consistent way.

  For example, search for Item A across sales orders that other users created in Order Management, and across sales orders that an administrator imported from an e-commerce site, call center, or through electronic data interchange (EDI).
- Use an orchestration process to route and manage each sales order across more than one fulfillment system. For example, use a schedule, ship, and bill process to route order lines to two different enterprise resource planning (ERP) systems.
- Send status updates from more than one fulfillment system to a capture system that resides outside of Order Management.

  For example, return a SHIPPED order status to an e-commerce system even if one warehouse system returns a Shipped status for an order line, but another warehouse returns an SHP status for the same line.
- Get a summary of statuses and exception orders. For example, view a graph that displays sales orders that are at risk of missing a promise date.
- Use Global Order Promising to collect supply data from more than one source and to set up business rules that automatically select the best fulfillment location to meet demand from any channel.

  Select according to future availability, expected delivery date, and preferred delivery method. Allocate scarce supply according to customer, channel, or to resolve order exceptions.

Integrate

- Use a mixture of cloud and on-premise environments for your capture and fulfillment systems. For example, import source orders from a cloud capture system, then fulfill them in an on-premise ERP system.
- Use a predefined integration with other Oracle cloud services to centrally manage orchestration policies, get global availability, monitor status, and manage exceptions.
- Import source orders from your order capture system, such as an e-commerce system, edit them in Order Management, and then fulfill them in your fulfillment system.

For details about how Order Management integrates channels with other systems, see *Overview of Integrating Order Management*.

Monitor Progress and Manage Exceptions

Use the Order Management work area to monitor progress and manage exceptions.

- Get a summary of statuses and exceptions according to customer, item, or supplier, and drill into data to get more detail.
- Filter sales orders according to customer, item, fulfillment location, supplier, status, or age.
- View order status.
- Use a Gantt chart to monitor processes.
- View and fix exceptions on one or more order lines.
- If a sales order is at risk of not meeting fulfillment dates, then use a jeopardy feature to identify issues and take corrective action.
- Use embedded intelligence to resolve exceptions. Use analytic details to make informed choices.

Simplify Change Orders

Set up compensation patterns and roll back steps so you can consistently control change orders. Use change order logic to make sure Order Management processes and revises sales orders consistently across all sales orders.

For example, if Order Management receives a quantity change from a source order, and if it hasn't shipped the item, then use change logic that allows the change, and roll back the fulfillment process so it can reschedule and send a new shipment request to your fulfillment system.

- Specify attributes that affect the change order and that automatically recognize a change order.
- Create rules that automatically handle changes so the user isn't required to intervene in every change that happens.
- Coordinate change order tasks with your fulfillment system.
- Set up tasks in the orchestration process that the change affects.
- Adjust fulfillment steps that the change affects.
- Cancel orders, add lines to orders, and change quantity.

Set Up Processing

Modify the predefined orchestration process that Order Management uses to meet most of your business requirements. You can also create a new one to meet your specific requirements.

- Use common, conditional, or interrelated flows for order lines.

- Create change order rules that automatically modify order fulfillment.
- Create rules that calculate the order completion date according to the requested date.
- Calculate lead times, modify order compensation steps, and so on.
- Create common statuses to use across your business process.
- Create draft orders in Order Management that you can submit later for processing.

You can also write your own Groovy script in an order management extension to modify your Order Management deployment. Create the extension point that determines when to run this script. For details, see *Overview of Creating Order Management Extensions*.

Enrich Source Orders For Fulfillment

- Create a business rule that transforms each source order into a sales order, modifies order attributes, and creates order lines.
- Add details to improve fulfillment. For example, add or modify attributes or add more items to an order so Order Management can efficiently fulfill the sales order.

Manage Problems and Recover from Errors

- Use a jeopardy feature to help you monitor each sales order against a fulfillment date, predict whether the sales order is on schedule to meet the fulfillment date and, if not, take corrective action.
- Use a central work area to recover sales orders that are in an error state.
- Locate the problem, identify the root cause, then adjust order fulfillment parameters to fix the problem.
- Modify dates and attributes that affect process planning.
- Query and examine a subset of sales orders according to a criteria that you specify.
- Take action on a single sales order, or on more than one sales order at the same time.
- Schedule a background process that automatically recovers errors according to parameters and filters that you set up.

Get Details

See:

- *Implementing Order Management*
- *Using Order Management*
- *Administering Pricing*
- *Tables and Views for Oracle SCM Cloud*
- For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.

To get technical details not covered in these books, see *Technical Reference for Order Management (Doc ID 2051639.1)*.

Related Topics

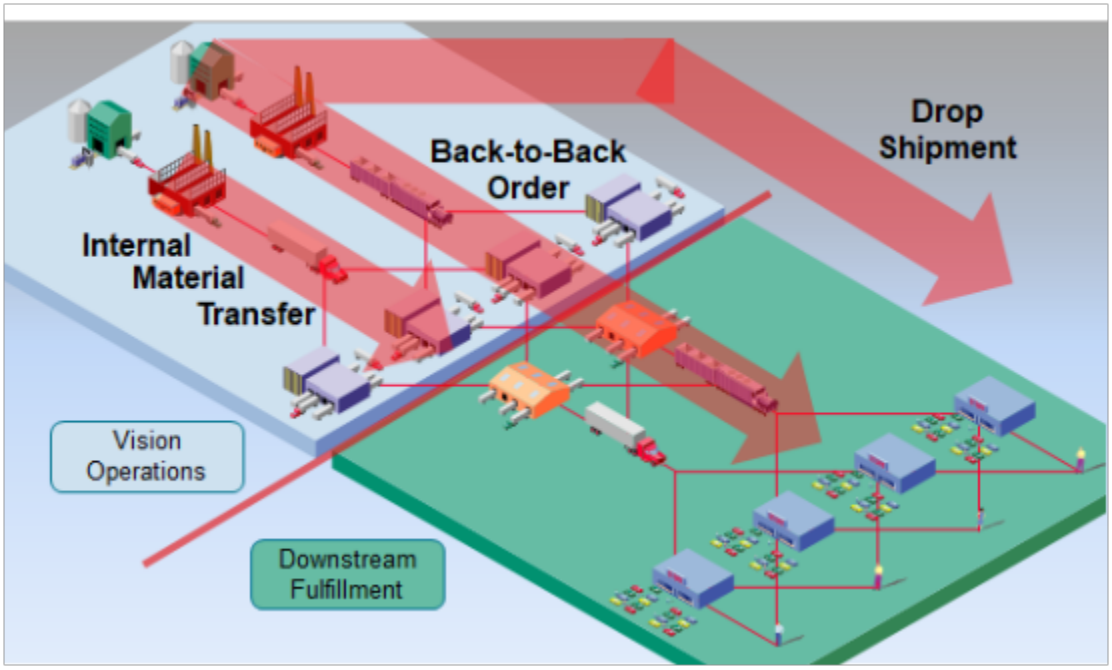
- How Order-to-Cash Works in Order Management

Use Order Management to Fulfill Different Types of Sales Orders

Use Order Management to fulfill a range of order types.

- Order a physical item that you ship, such as desktop computer.
- Order a nonphysical item that you don't ship, such as a subscription or warranty.
- Change an existing order, such as adding an item to an order or changing the quantity.
- Transfer an item between different parts of the same organization. You create a transfer order in Supply Chain Orchestration and import into Order Management.
- Return an item.

Order Management can work with your fulfillment system in different fulfillment flows.



Note

Flow	Value
Back-to-back	Global Order Promising reserves supply against each sales order until you ship it.
Drop shipment	Your supplier ships the sales order directly to your customer, bypassing your factory, warehouse, and distribution system. The shipper for a drop shipment that happens outside of your organization is usually a 3rd party supplier. The shipper for a drop shipment that happens inside of your organization is another business unit.
Internal material transfer	Transfer an item between two different parts of the same organization. For example, transfer the AS54888 desktop computer from the Vision Incorporated factory to the Vision Incorporated warehouse. Global Order Promising can promise an internal material transfer and prioritize supply across sales orders.

Flow	Value

Related Topics

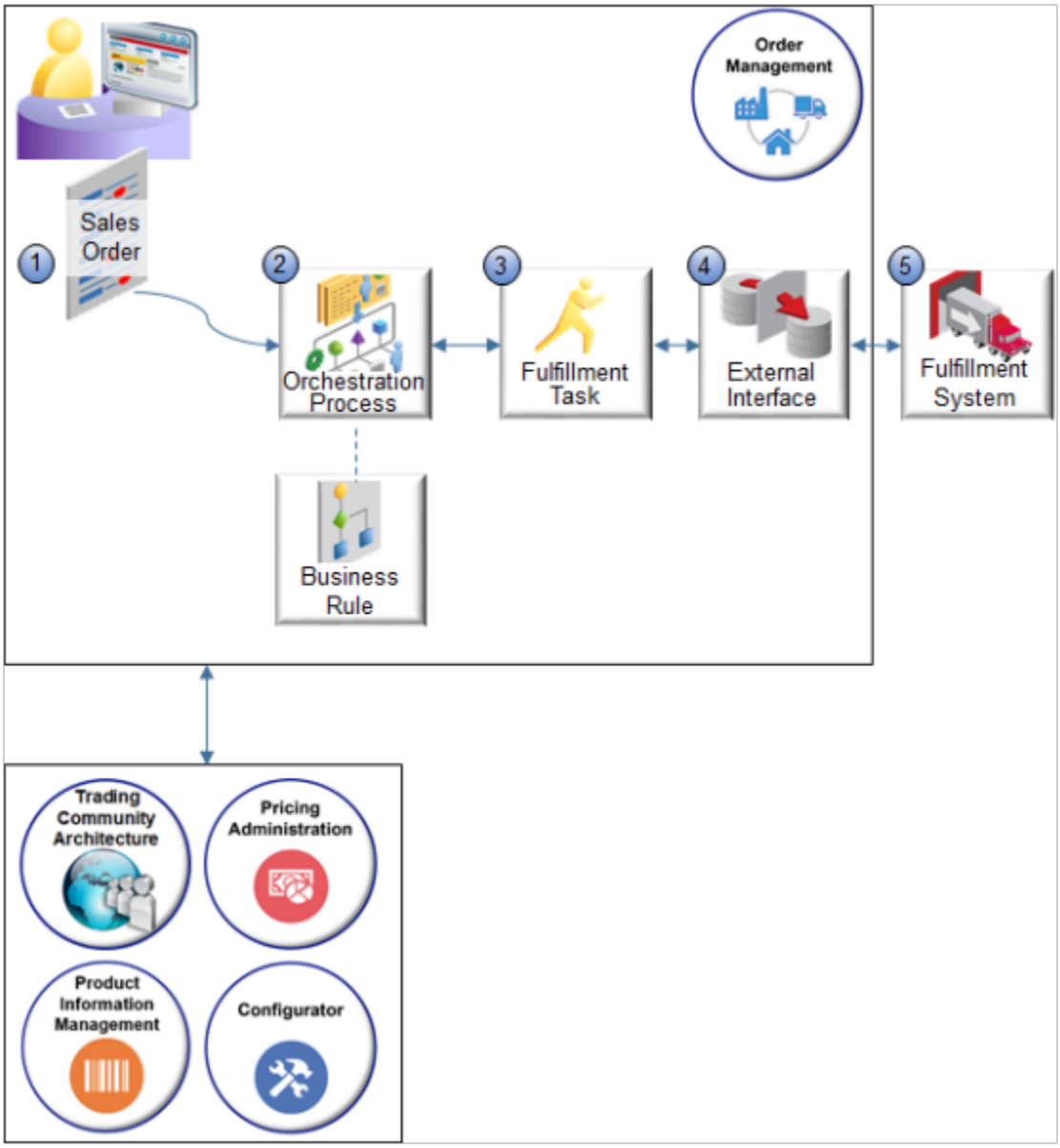
- Overview of Drop Ship in Order Management
- Use Supply Chain Orchestration in Your Back-to-Back Flows
- Orchestrate Internal Material Transfers

Details

How Order-to-Cash Works in Order Management

Use Order Management to order and fulfill an item, such as a printer, a configured item, such as a desktop computer, or a service, such as monthly maintenance for a network of desktop computers.

Here's how Order Management fulfills a sales order when the user creates it in the Order Management work area.


Order Management
Sales Order
1
2
3
4
5
Orchestration Process
Fulfillment Task
External Interface
Fulfillment System
Business Rule
Trading Community Architecture
Pricing Administration
Product Information Management
Configurator

Note

1. A user, such as the Order Entry Specialist, clicks Create Order in the Order Management work area to create a new sales order.
    - The user searches the Customer attribute on the order header. Order Management gets details about the customer from the party object in Trading Community Architecture. For details, see *Overview of Displaying Customer Details on Sales Orders*.
    - Order Management communicates with Pricing Administration to get pricing details that apply to the entire order, such as the pricing strategy and pricing segment. The strategy depends on the value that you set in the Customer attribute. For details, see *How Profiles, Segments, and Strategies Work Together*.
    - The user searches for an item on the catalog line of the sales order. Order Management gets details about the item from the Product Information Management work area. You use the Product Information Management work area to set up and manage details about the item.

      Order Management also communicates with the Pricing Administration work area to price the line. For details, see *How Pricing Prices Sales Orders*.
    - The user clicks Add to move the line to the order line area, then clicks Submit.
    - Order Management validates the order, converts order lines to fulfillment lines, then an assignment rule identifies the orchestration process that Order Management runs to fulfill each fulfillment line. You can set up this rule. For details, see *Assign Orchestration Processes*.
2. An orchestration process contains steps that fulfill the order. Examples of steps include schedule, reserve, ship, invoice, and so on.

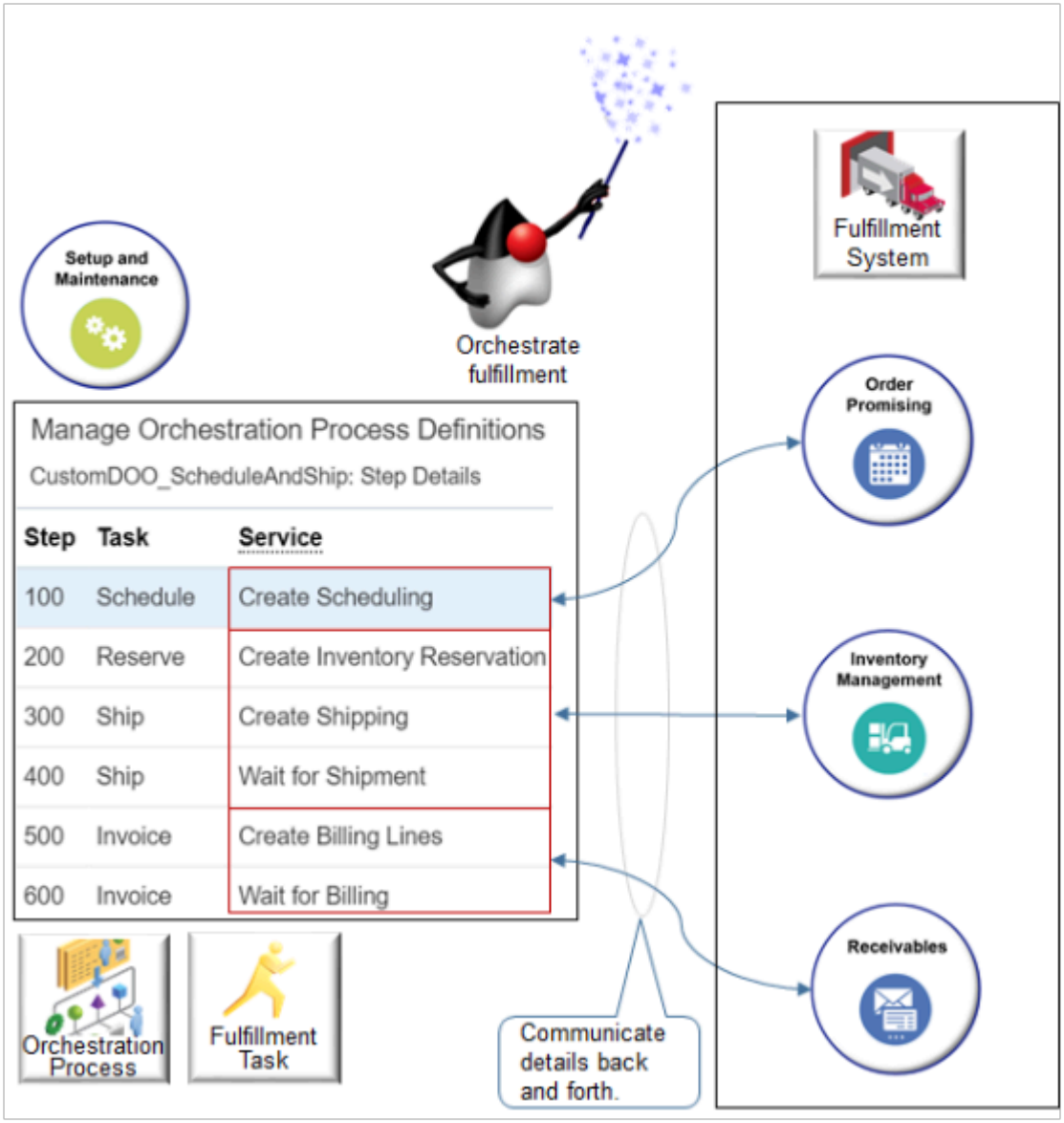
   The steps vary so they optimize order fulfillment according to the needs of each order. The orchestration process determines process logic, such as statuses to use for the order, how to do forward planning and backward planning, how to compensate for change to the order, and so on.

   Here's more detail about what the orchestration process does.
    - References data in Product Information Management to determine how to orchestrate the order according to the item. You can specify this data. For details, see *Get Data from Product Information Management*.
    - References Supply Chain Planning to cross-reference and validate data.

      You can set up data collection parameters, enable data for collections, and so on. For details, see *Quick Start for Setting Up Order-to-Cash*.
    - Uses business rules to orchestrate each order. You can set up a variety of rules.

      For example, set up a rule that uses a different lead time depending on whether the inventory organization resides in Denver or San Francisco, or a rule that makes sure Order Management doesn't attempt to ship a nonshippable item, such as a video that your customers can only stream from the cloud. For details, see *Overview of Using Business Rules With Order Management*.

3. To orchestrate fulfillment, each orchestration process step calls a service that does a fulfillment task.



For example:

- The Create Scheduling service communicates with Order Promising. Order Promising determines availability and promises the order.
  - Cross-references items before it sends the fulfillment request to your fulfillment system.
  - Gets supply and demand data from Oracle Supply Chain Planning to determine availability in the supply chain, such as inventory levels in warehouses.
  - Schedules the best fulfillment options for each fulfillment line. Your users can also manually use the Order Management work area to select some options.

  Most predefined orchestration processes include a scheduling step that calls Order Promising. You can also set up an orchestration process that doesn't include a scheduling step and that doesn't use

Order Promising. For example, if your organization sells downloadable literature that doesn't require a warehouse or shipping.

- Task services interpret replies and updates from fulfillment systems, such as Inventory Management. For example, the orchestration process calls:
  - The Create Inventory Reservation service to reserve inventory for your item, then waits until Inventory Management sends confirmation that it reserved inventory.
  - The Create Shipping service to create a shipment, and the Wait for Shipment service to wait for confirmation from Inventory Management.
  - The Create Billing Lines service and the Wait for Billing service to call Receivables and finish the order-to-cash process.

For details about how to use task services, see *Fulfillment Tasks*.

You use the Setup and Maintenance work area to set up an orchestration process.

4. The interface communicates order details between Order Management and the fulfillment system. It primarily routes the fulfillment request and converts data so the fulfillment system can correctly use the data.

   You can also use the interface to send a request to your fulfillment system that resides outside of Order Management to fulfill the order. The system that resides outside of Order Management processes the request and sends completion updates through the interface so the orchestration process can move to the next step. For details, see *Overview of Integrating Order Management with Other Oracle Applications*.
5. Here's what happens during fulfillment.

   - Inventory Management manages logistics and inventory for the order, including schedule, reserve, receive, and ship each item.
   - Receivables processes billing details, including one-time charges and recurring charges, then sends them to Oracle Financials.
   - Oracle Financials does financial transactions for the order. It creates an invoice, manages accounts receivable, processes payments, and manages revenue.

   The fulfillment system communicates updates through the interface to task services throughout fulfillment.

Note

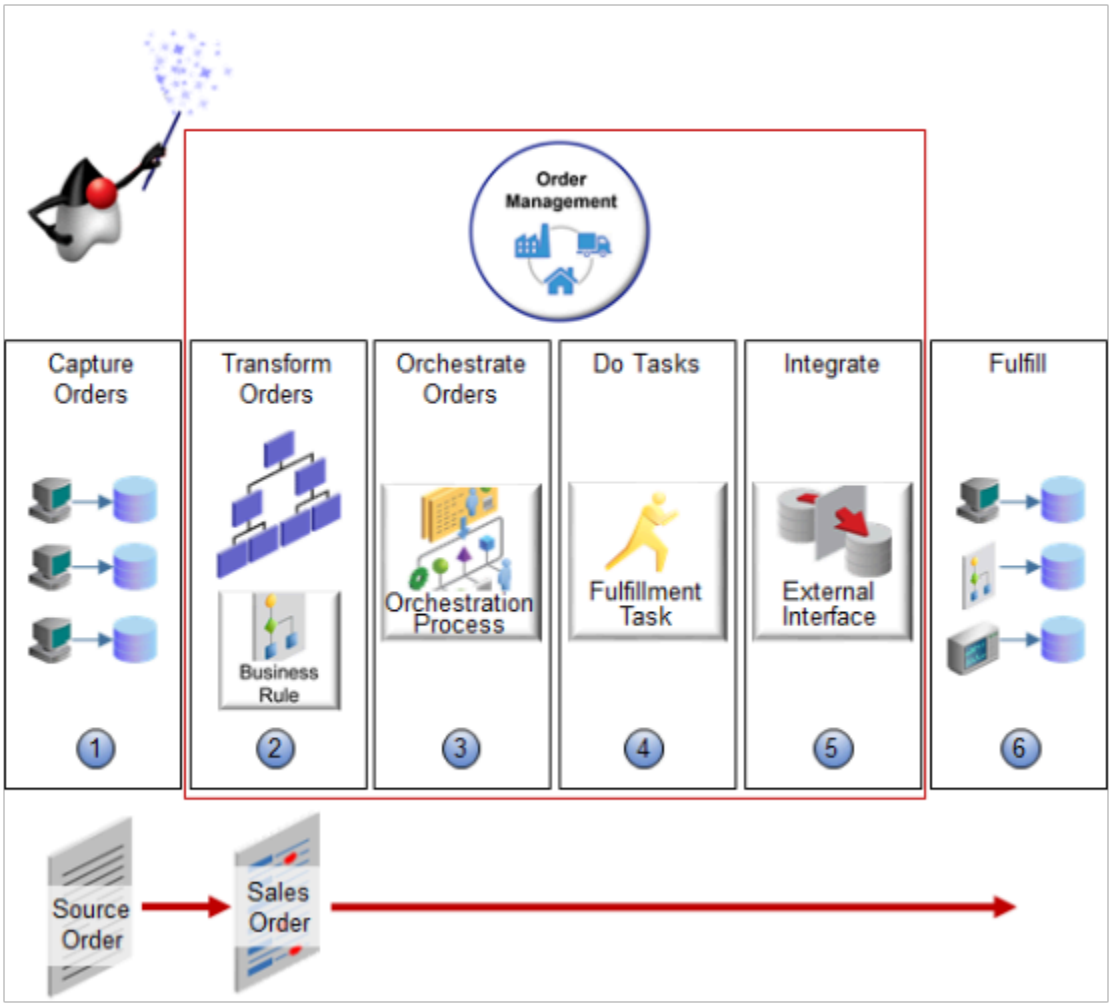
- This topic describes one way to fulfill orders. You can set up Order Management to do fulfillment differently, according to your unique fulfillment environment.
- You can import orders into Order Management from an upstream order capture system. For details, see *Overview of Importing Orders Into Order Management*.
- For details about other fulfillment flows, such as drop ship, back-to-back shipping, or internal transfer, see *Implementing Manufacturing and Supply Chain Materials Management*. For illustrations of these flows, see:
  - *Order to Cash Standard Pick from Stock Order Flow (Doc ID 2239712.1)*
  - *Order to Cash Back to Back (B2B), Buy Flow (Doc ID 2276547.1)*
  - *Order to Cash Back to Back (B2B), Make Flow (Doc ID 2275403.1)*
  - *Order to Cash Back to Back (B2B), Transfer Flow (Doc ID 2273369.1)*

Related Topics

- How Order-to-Cash Works with Order Capture Systems
- Overview of Order Management
- How Data Flows Through Order Management
- Overview of Drop Ship in Order Management
- Overview of Orchestration Processes

How Data Flows Through Order Management

Order Management uses processes and tasks to fulfill each source order that it receives from a source system.



Note

1. **Capture orders**. A source system that resides outside of Order Management sends a source order to Order Management.
   The terminals and databases in step 1 in the diagram represent different channels that your deployment can use as source systems. For example, a legacy system, or some other channel.
   The flow for a sales order that the Order Entry Specialist creates in the Order Management work area is similar, except it typically starts with orchestration at step 3.
2. **Transform orders**. Transformation rules separate the source order into a hierarchy that Order Management can use and process.
   Transformation finishes, then assignment rules assign an orchestration process to the sales order.
3. **Orchestrate orders**. One or more orchestration processes contain steps that do fulfillment tasks. Each fulfillment request starts here.
4. **Do tasks**. Task services send a request to the fulfillment system to run the fulfillment task, such as ship the item or invoice the item. A system that resides outside of Order Management can also send a request to a task service.
5. **Integrate**. Here's what the interface does.
   - Cross-references data that the request contains.
   - Uses routing rules to identify the fulfillment system that fulfills each fulfillment line. For details, see *Route Requests from Order Management to Fulfillment Systems Without Cross-References*.
   - A connector sends the request to the fulfillment system.
6. **Fulfill**. The fulfillment system processes the request.
   - The fulfillment system accepts the request then sends a reply.
   - The interface converts the reply from the fulfillment system.
   - The task service processes the reply from the fulfillment system.
   - The orchestration process runs the next orchestration process step.

Interface

The interface manages communication between Order Management and your fulfillment system.

- The fulfillment system doesn't communicate directly with the orchestration process. Instead, the interface provides an intermediary that Order Management uses to route requests.
- Order Management uses a web service to route each request from the task service to the fulfillment system, and then from the fulfillment system to the task service.
- The connector modifies the structure and content of the outbound message so it matches the inbound interface that the fulfillment system uses.
- You can set up each web service and routing rule, then register the connector web service. This set up integrates the fulfillment system with the interface.

The interface provides benefits.

- Keeps the source system or fulfillment system that resides outside of Order Management separate from the orchestration process so it isn't necessary to modify the orchestration process every time you integrate another source system or fulfillment system.

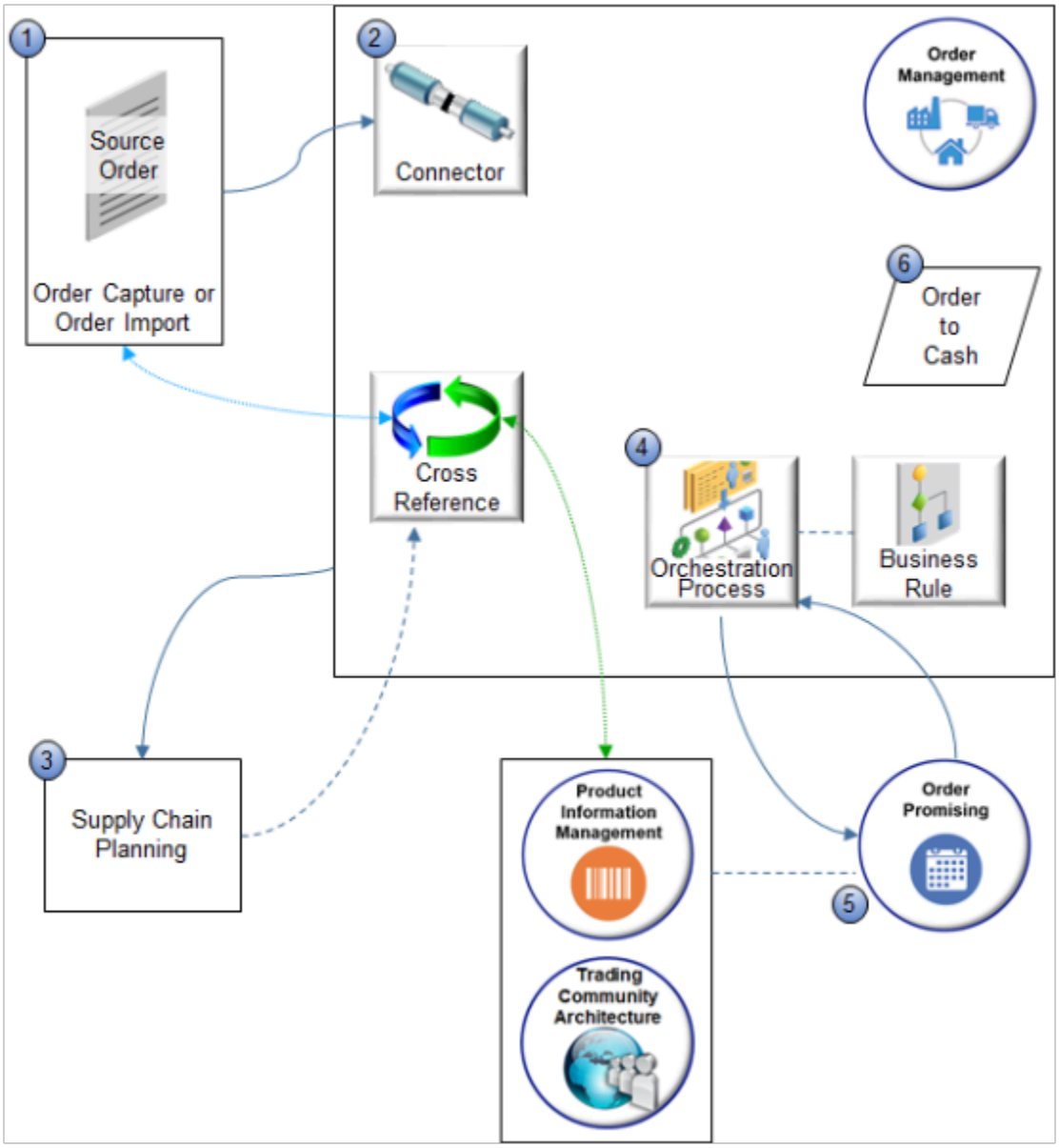
- Provides a flexible integration that you can use in a Service Oriented Architecture (SOA) with a system or application that resides outside of Order Management.
- Provides a complete, open, and integrated solution that lowers cost of ownership.

Related Topics

- Task Services
- Manage Connector Details Between Order Management and Your Fulfillment System
- Manage Routing Rules
- Integrate Order Management with Source Systems

How Order-to-Cash Works with Order Capture Systems

Use an order capture system that resides outside of Order Management, such as Oracle Configure, Price and Quote Cloud, to send a source order to Order Management.



For the most part, Order Management uses the order-to-cash flow. For details, see *How Order-to-Cash Works in Order Management*. Note these important differences.

1. An order capture system that resides outside of Order Management captures a source order. For example, a user might use your legacy application to enter a source order.
2. You set up a that the capture system can use to send the source order to Order Management. If the source system and Order Management use different domain values, then the connector transforms the structure and values from the domain that the source system uses to the domain that Order Management uses.
   For details, see *Integrate Order Management with Source Systems*.
3. Supply Chain Planning collects data about the source order, then uses it to create cross-references and planning data.
   Use a cross-reference help to manage the representation of data across systems. A cross-reference relates business data between your order capture system, order fulfillment system, and Order Management. For

example, an item cross-reference can create a relationship between an item, such as Widget x, that resides in your source system, and a representation of Widget x in Order Management.

You can cross-reference across warehouses, units of measure, carriers, currencies, shipping methods, payment terms, accounting rules, invoicing rules, service levels, tax classification codes, and so on.

4. An orchestration process references business rules to transform each source order into a sales order that Order Management can understand and fulfill.
5. Order Promising gets item details from Product Information Management, customer details from Trading Community Architecture, and planning data from Supply Chain Planning to promise the sales order.
6. Order Management uses the order-to-cash flow to continue processing.

Note

- Order Management updates the order status, fulfillment line status, and invoice status during the flow, and communicates each update to your order capture system.
- Order Management can use an order capture service to communicate updates to the source system. To receive these updates, the source system must subscribe to the events. If Order Management receives an update from the source system, then it replans the orchestration process. It replans every time it receives an update.
- Here are the attributes that Order Management uses to calculate the planned dates for each step and task, starting with the first step it does in chronological order.
    - Default Lead Time
    - Lead Time UOM
    - Lead-Time Expression
- Order Management doesn't do the same amount of transformation for a sales order that a user creates in the Order Management work area because these orders already include the data and use the structure that Order Management requires.
- A single source order might contain order lines that include one-time charges, and other order lines that include recurring charges. If Order Management receives a source order that includes a recurring charge, then Oracle Receivables creates a recurring billing invoice for these lines.
- Order import typically uses the same flow that this topic describes, except you import source orders from a spreadsheet or through a web service.

Related Topics

- Import Orders Into Order Management
- How Order-to-Cash Works in Order Management
- Integrate Order Management with Source Systems
- How the Order Orchestration and Order Promising Processes Use the Collected Planning Data

2 Implement Business Process Flows

Order-to-Cash

Roadmap for Setting Up Order-to-Cash

Use this road map as the high-level procedure you use to implement order-to-cash.

- Do the tasks that this topic references in the same sequence that the topic displays them.
- Steps after step 3 primarily reference tasks in the Orders functional area. Do each of these tasks depending on your business requirements.
- To do a task, click each one in the Orders area. If the Orders area doesn't display a task, then search for it in the Search Tasks field.
- The set up you do depends on your business requirements. You typically do most or all of the tasks that integrate Order Management. Other tasks depend on your requirements, such as tasks that control order status, constrain changes, or modify an orchestration process.
- Use the Details column to get help with the task.
- The Prepare column describes what you do to prepare for implementation. Do this set up before you administer order-to-cash. It will help to avoid interruption and downtime during administration. Use the Prepare column to check off preparation work as you finish it.
- To monitor progress, add a check mark to the Done column when you finish each functional area or task.
- Learn how implement drop ship. For details, see *Overview of Drop Ship in Order Management*.

Implement order-to-cash.

1. For details, see *Guidelines for Setting Up Order-to-Cash*. It contains details that help to make the implementation go smoothly.
2. Go to the Setup and Maintenance work area, then select the Order Management offering.
3. This topic assumes you will modify the predefined implementation that's available through Setup. For details about setting up a new implementation, see Points to Consider When Implementing Order-to-Cash.
4. Set up these functional areas.

Functional Area	Done
Initial Users	_____
Enterprise Profile	_____
Organization Structures	_____
Users and Security	_____

Functional Area	Done
Items	_____
Catalogs	_____
Customers	_____

These areas reference common tasks. A typical implementation requires that you complete most or all common tasks. Learn how to set them up. For details, see *Implementing Common Features for Oracle SCM*.

5. In the Functional Areas list, click **Orders**.
6. In the Orders area, click **Required Tasks > All Tasks**
7. Integrate Order Management.

Task	Details	Prepare
Manage Web Service Details	*Integrate Order Management with Source Systems*	_____Gained access to the administrator privilege and administrator role so we can use Oracle Wallet Manager.
Manage Trading Community Source Systems Manage Upstream and Fulfillment Source Systems Manage External Interface Web Service Details	*Integrate Order Management with Source Systems*	_____Located the time zone where the server that the source system uses is located. _____Identified the URL that locates the connector that resides on the source system. _____Identified the User Name and Password that the Status Update service requires.
Manage Trading Community Source Systems Manage Item Relationships	*Guidelines for Using Web Services to Integrate Order Management*	_____Identified source system details _____Identified role and privilege requirements _____Identified cross-reference requirements
Manage External Interface Web Service Details	*Example of Integrating Order Management with Your Fulfillment System*	_____Identified an integrated development environment we can use to create a transformation style sheet.

Task	Details	Prepare
		_____Acquired the user credentials that the service provider needs when calling their web service. _____Acquired contact details for the IT administrator who works for the service provider. _____Identified the URL that locates the web service that resides on the fulfillment system. _____Acquired the security certificate from the service provider.
Manage External Interface Routing Rules	*Route Requests from Order Management to Fulfillment Systems*	_____Created a list that includes the names of the items we must route to fulfillment systems, the unique identifier for each item, and the name of each fulfillment system where Order Management must route the request.

8. Control application behavior. For details, see the section that starts with *Use Order Profiles to Control Order Management Behavior*.

Task	Details	Prepare
Manage Order Management Parameters	*Manage Order Management Parameters*	_____ Determined whether our business requirements will allow or disallow the Configurator to select items in a configuration and to modify a configuration after the user adds a configured item. _____ Determined the date that Order Management must use when it determines the configure options that it displays for a configured item. _____ Determined how Order Management will display the Ship-to Address and the Bill-to Location for each customer during order entry. _____ Determined whether the Configurator stops processing when it encounters an error.

Task	Details	Prepare
		_____ Identified the item validation organization that Order Management must use to validate and display items. _____ Identified an Order Management user that the buyer can contact to resolve a problem that happens with a drop ship. _____ Determined whether the Configurator validates each order that includes a configured item during order import.
Manage Order Profiles	*Use Order Profiles to Control Order Management Behavior*	_____ Identified the value to use when converting a currency. _____ Identified the currency to display in the Order Management work area. _____ Identified the customer to use when filtering status data on the Overview page of the Order Management work area. _____ Determined whether use the source order number as the order number during transformation. _____ Determined the number of seconds to wait after an action finishes.

9. Administer pricing. For details, see *Overview of Oracle Pricing*.
10. Import source orders. If you use a source system, then you must import source orders.

Task	Details	Prepare
Not applicable	*Import Orders Into Order Management*	____ Acquired access to our source order data. ____ Identified a tool we can use to manipulate source order data, such as SQL (Structured Query Language), or ODI (Oracle Data Integrator).
Create Source System Manage Upstream and Fulfillment Source Systems	*Automatically Import Source Orders into Order Management*	____ Identified the attributes we will use in the Request Payload.

Task	Details	Prepare
Manage Standard Lookups Manage Item Relationships		

11. Transform source orders. If you use a source system, then you must transform source orders.

Task	Details	Prepare
Manage Product Transformation Rules	*Set Up Transformation*	____ Acquired the user and password that we need to sign into the Product Information Management work area.
Manage Product Transformation Rules	*Create Transformation Rules*	____ Described the structure of the source order data. ____ Identified the type of transformation that must happen so Order Management can support our source data. ____ Described requirements for pretransformation rules. ____ Described requirements for transformation rules. ____ Described requirements for posttransformation rules.
Manage Product Transformation Rules	*Create Transformation Rules*	____ Described requirements for advanced posttransformation rules, if necessary.

12. Set up statuses.

Task	Details	Prepare
Manage Task Status Conditions	*Manage Task Status Conditions*	____ Described our requirements for managing the statuses that our fulfillment systems provide.
Manage Status Values Manage Orchestration Process Definitions	*Add Status Conditions to Fulfillment Lines*	____ Described our requirements for managing the statuses for each fulfillment line.

Task	Details	Prepare
Manage Status Values	*Orchestration Process Status*	____ Described our requirements for grouping orchestration process statuses so they're meaningful.
Manage Orchestration Process Definitions	*Add Status Conditions to Orchestration Processes*	____ Described our requirements that specify when to set statuses for orchestration processes.

13. Set up processing constraints.

Task	Details	Prepare
Manage Processing Constraints	*Manage Processing Constraints*	____ Described requirements regarding who can change a sales order, what can change in the sales order, and when the change can happen.
Manage Constraint Entities	*Constrain Changes to Attributes*	____ Described requirements regarding the order attributes that can change.

14. Control change orders.

Task	Details	Prepare
Manage Order Attributes That Identify Change	*Manage Order Attributes That Identify Change*	____ Described our requirements regarding what changes Order Management will allow on an existing sales order, and when it allows the change, including order attributes, users, and timing.
Manage Orchestration Process Definitions	*Measure the Cost of Change*	____ Estimated the business cost associated with each change.
Manage Orchestration Process Definitions	*Compensate Sales Orders That Change*	____ Described our requirements regarding the order compensation we will allow Order Management to do.

15. Notify systems when orders change.

Task	Details	Prepare
Manage Business Event Trigger Points	*Overview of Sending Notifications from Order Management to Other Systems*	____ Described our requirements regarding when Order Management must notify another system that a sales order changed.

16. Use jeopardy to manage delays.

Task	Details	Prepare
Manage Jeopardy Priorities Manage Orchestration Process Definitions Manage Jeopardy Thresholds	*Set Up Jeopardy and Lead Time to Manage Delay*	____ Described our requirements regarding when Order Management must notify an order manager that a sales order is in jeopardy, including the threshold when to notify for each step in an orchestration process. ____ Described requirements regarding the amount of lead time that Order Management must use to finish each orchestration process step.

17. Manage task types.

Task	Details	Prepare
Manage Task Types	*Create Your Own Task Type*	____ Described our requirements regarding setting up task types.

18. Set up flexfields.

Task	Details	Prepare
Manage Order Extensible Flexfields	*Guidelines for Setting Up Extensible Flexfields in Order Management*	____ Described our requirements regarding what details we must display in fields, including the data and location in the user interface.

19. Set up orchestration processes.

Task	Details	Prepare
Manage Orchestration Process Definitions	*Guidelines for Setting Up Orchestration Processes*	____ Confirmed that the predefined orchestration processes don't meet our requirements. ____ Identified configuration details for each orchestration process that we must create, such as task type to use, change logic, process planning, jeopardy, statuses, and so on.
Manage Orchestration Process Definitions	*Set Up Orchestration Processes*	____ Described the IF/THEN rules that we need for each orchestration process. ____ Described orchestration planning and replanning that we need for each orchestration process. ____ Specified the behavior that we need for each orchestration process step, such as the task type to use, the service to call, lead-times, and so on. ____ Created a flowchart mock-up for each orchestration process.
Manage Orchestration Process Definitions	*Deploy Orchestration Processes*	Not applicable.
Manage Orchestration Process Definitions	Set Up Lead-Times for Orchestration Process Steps	____ Described the IF/THEN rules and conditions that determine the lead-time to use for each orchestration process step.
Manage Orchestration Process Definitions	*Select Fulfillment Lines for Orchestration Process Steps*	____ Described the IF/THEN rules that select the fulfillment line to run for each orchestration process step.
Manage Orchestration Process Definitions	*Pause Orchestration Processes Until Events Happen*	____ Specified the IF/Then rules that Order Management must use when it pauses each of our orchestration processes and each orchestration process step.
Manage Orchestration Process Assignment Rules	*Assign Orchestration Processes*	____ Specified the IF/Then rules that Order Management must use when it assigns each

Task	Details	Prepare
		of our orchestration processes and each orchestration process step.
Not applicable	*Resume Paused Orchestration Processes*	____ Identified the pause tasks we will release to resume a paused orchestration process.

Related Topics

- Quick Start for Setting Up Order-to-Cash
- Overview of Order Management

Guidelines for Setting Up Order-to-Cash

Identify the features you must set up to support the order-to-cash flow in your business environment, then estimate the effort required to set them up.

Estimate Effort Required to Set Up Features

Do the preparation described in *Roadmap for Setting Up Order-to-Cash*. This preparation will provide a solid understanding of the features you must implement, and the set up involved.

Create a rough estimate of the amount of effort your set up will require. A predefined feature usually needs less set up than a feature that isn't predefined. For example, if you need the configurator and order import, then include the setup effort for them in your project plans, and consider whether you need the privileges to support them.

Feature	Comes Predefined	Setup Required
Dashboard	Yes	No
Create Order	Yes	Yes
Revise Order	Yes	No
Return Order	Yes	Yes
Search and View Orders	Yes	No
Configurator	No	Yes
Order Import	No	Yes

Feature	Comes Predefined	Setup Required
Order Transformation	No	Yes
Order Status	No	Yes
Constraints	No	Yes
Change Order	No	Yes
Notifications	No	Yes
Jeopardy	No	Yes
Flexfield	No	Yes
Orchestration Process	No	Yes
Process Orders from Trading Partners	No	Yes
Advance Shipment Notice	No	Yes
Collaboration Messaging Framework	No	Yes
Approve Sales Orders	No	Yes
Screen Orders for Trade Compliance	No	Yes
Integrate Order Management with Transportation Management	No	Yes
Credit Check	No	Yes
Credit Cards	Yes	No
Internal Sales Orders	Yes	No
Attachments	No	Yes

Modify the Predefined Offering or Set Up a New One

Modify the Predefined Offering

Order Management comes predefined with most setup tasks already done for you, but you can modify the predefined offering.

1. Go to the Setup and Maintenance work area, then select the Order Management offering.
2. Notice the functional areas that display above functional area Orders, such as Initial Users, Enterprise Profile, and so on. These areas are common areas. For details about how to set them up, see *Implementing Common Features for Oracle SCM*.
3. In the Functional Areas list, click **Orders**.
4. In the Orders area, click **Required Tasks > All Tasks**.
5. In the Orders list, drill into and finish each task as necessary, depending on your business requirements.
   If you find that modifying the predefined offering doesn't meet your business requirements, then create a new implementation.

Create a New Implementation Project

If the predefined offering doesn't meet your business requirements for some reason, then create a new implementation project.

1. Go to the Setup and Maintenance work area.
2. On the Setup page, click **Tasks > Edit Implementation Projects**.
3. On the Implementation Projects page, click **Actions > Create**.
4. On the Create Implementation Project page, modify the attribute values, as necessary, then click **Next**.
5. On the Select Offerings to Implement page, in the Name list, expand **Order Management**.
6. In the Order Management row, and in the Pricing row, enable the Include option, then click **Save and Open Project**.
   You can create one or more implementation projects for the offerings and options. Each Oracle application creates the task list you must finish for each project you create. An Application Implementation Manager can set up these task lists, and assign and track each task that these lists contain.
7. On the Implementation Project page, in the Task list, expand **Order Management**, then do the tasks that your business flow requires.

Consider Which Scheduled Processes You Must Run

Scheduled Process	Description	Get Details
Collection Job Set	You collect organization information from your source system.	Collect Data section in *Quick Start for Setting Up Order-to-Cash*
Refresh and Start the Order Promising Server	Updates the Global Order Promising data.	Set Up Global Order Promising section in *Quick Start for Setting Up Order-to-Cash*
Load Interface File for Import *Import Sales Orders* *Delete Orders from Interface Tables*	Use these scheduled processes when you import source orders.	*Import Orders Into Order Management*
Transfer Invoice Details to Supply Chain Financial Flow Orchestration *Transfer Ownership Change Events to Receiving*	Send details about validated invoices, canceled invoices, and corrected invoices to Financial Orchestration.	*Indicate an Ownership Change During Drop Ship*

Scheduled Process	Description	Get Details
	Send details about the AP Invoice Match from Financial Orchestration to the receiving process.	
Update or Close Sales Orders	Order Management might display sales orders and fulfillment lines as open even if it closed all fulfillment lines that these orders and lines reference. You can use Update or Close Sales Orders to fix this problem.	*Update or Close Fulfillment Lines That Remain Open*
Generate Constraint Packages	Constrain the changes that your users can make.	*Manage Processing Constraints*
Publish Extensible Flexfield Attributes	Publish and deploy an extensible flexfield.	*Guidelines for Setting Up Extensible Flexfields in Order Management*
Plan Orchestration Processes	Update an orchestration process plan at regular intervals according to the frequency that your deployment requires.	*Guidelines for Setting Up Orchestration Processes*
Release Pause Tasks	Release a pause task so processing can continue.	*Pause Orchestration Processes Until Events Happen* *Resume Paused Orchestration Processes*
Generate Bucket Sets	Automatically keep bucket sets up to date with reference data and transactional data.	*Use Decision Tables and Bucket Sets in Business Rules*
Import AutoInvoice. For details, see *Update Intercompany Receivables Invoice Import Details*.	Import and validate transaction data from Order Management or financial systems that reside outside of Oracle to create invoices, debit memos, credit memos, and account credits in Oracle Receivables.	*Create One Invoice for Sales Orders with Items That Can and Can't Ship*

Consider How Processing Constraints Affect Your Setup

Order Management comes predefined with a variety of processing constraints that limit the changes you can make to attributes or what you can do in the Order Management work area. Examine them to make sure they won't cause problems with your custom set ups. For example, you can't write an order management extension that updates the Quantity attribute on a fulfillment line after Order Management already interfaces the line to billing.

Here are a few more examples of predefined constraints that might affect your custom set up.

Constraint	Description
Ordered Date Update	Prevent updates to the Ordered Date attribute on the order header.
Order Line Deletion	Prevent deleting an order line.

Constraint	Description
Fulfillment Line Bill-to Customer Update	Prevent updates to the Bill-to Customer attribute on the fulfillment line.
Payment Terms Are Missing	Prevent submitting a fulfillment line when the line doesn't have payment terms.

There are many predefined constraints, but you can filter them to reduce the ones you must examine.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, use Query by Example to filter the constraint entity. For example, if your set up affects something that happens:
    - On the order header before the user clicks Submit, set the filter for Constraint Entity to Order Header.
    - On the order line before the user clicks Submit, set the filter for Constraint Entity to Order Line.
    - After the user clicks Submit, set the filter for Constraint Entity to Order Fulfillment Line.
3. Refine the search results. Use Query by Example to filter the constrained operation. For example, if your set up:
    - Updates the value of an attribute, set the filter for Constrained Operation to Update.
    - Deletes something, set the filter for Constrained Operation to Delete.
4. Refine the search results. If you know the name of the attribute that your set up manipulates, use Query by Example to filter for it. For example, if your set up updates the value of the Ship-to Customer on the fulfillment line, then:
    - Set the filter for Constraint Entity to Order Fulfillment Line.
    - Set the filter for Constrained Operation to Update.
    - Set the filter for the Attribute Name to Ship-to Customer.
5. In the Details area, examine the conditions to determine whether one of them applies to the behavior you're encountering.

    Assume you create an order management extension that updates the Bill-to Customer attribute on the fulfillment line when the status is Awaiting Billing. At run time you encounter an error that prevents you from updating the line. You decide to examine the constraints.

    - Set the filter for Constraint Entity to Order Fulfillment Line.
    - Set the filter for Constrained Operation to Update.
    - Set the filter for the Attribute Name to Bill-to Customer.

    The Message attribute in the Details area describes that you can't update the Bill-to Customer because Order Management already fulfilled the fulfillment line.

Consider Which Licenses You Need

You need different licenses to access different set up tasks. For example, you need one license to set up Oracle Financials and a different license to set up Oracle CX Sales. Learn how to get the licenses that you need. For details, see *Whom To Contact For Licensing Queries? (Doc ID 1391957.1)*.

Quick Start for Setting Up Order-to-Cash

Do the minimum steps needed to set up Oracle Order Management when you don't use all the tasks that are available in the Order Management offering, such as setting up a test instance of Order Management.

> **CAUTION:** This topic doesn't include all the steps and security tasks that are required to fully set up Order Management. It includes only the steps needed so you can receive and fulfill test orders.

For details about how to use the predefined set up, see *Guidelines for Setting Up Order-to-Cash*.

Summary of the Steps

1. Prepare.
2. Set up common components.
3. Set up item organizations and product models, and import items.
4. Connect source systems and set up customers.
5. Collect data and set up Global Order Promising.
6. Set up Order Management and Pricing, and test your set up.

Prepare

1. Consider the data you will use in your test environment.

   For example:

   - Items you will add to a sales order
   - Customers who will order the items
   - Item Organization you will use as the source to fulfill each item
   - Units of measure (UOM) you will use
   - Currency you will use in each sales order
   - Users who will create sales orders

   As part of planning your test, create a list for each of these data, such as a list of items, list of customers, and so on. You can then use these details later during set up.
2. Get the URLs, User IDs, and passwords you need to access Order Management and other applications, such as the Security Console.

   Get these details from the Oracle provisioning team.
3. Get the user name and password for each user. Contact the person who installed the systems to get the user names and passwords they used or specified when they installed and provisioned the application.

User	Description
Super user for Oracle Applications	The default user name is FAADMIN.
System administrator for the Security Console	Not applicable.
System administrator for Oracle Identity Manager	The default system administrator is XELSYSADM.

4. Identify details for your test orders.
    - Identify the items that the test orders will contain and the customers who will order them.
    - Identify the item organizations that you will associate with these items.
    - Identify the units of measure (UOMs) and currencies that these test orders will use.

Security Tasks You Might Need to Do

Get a list of the tasks you might need to do. For details, see *Securing SCM*.

Here are the tasks you can do in the Setup and Maintenance work area after you acquire the super user.

- Manage Job Roles
- Manage Data Security Policies
- Manage Data Access for Users

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Set Up Common Components

1. Prepare the super user for user management and configuration.
2. Prepare the IT Security Manager Role.
3. Create the setup task list for the Order Management offering.

   This offering includes the tasks you must do.

   A large task list displays when you create the list for the Order Management offering. Make sure you do step 5, Set Up Enterprise Structures. Also do step 3, Set up Item Organizations, through step 10, Set up Pricing, then create or import orders.
4. Set up implementation users.

   For details, see *Set Up User Roles and Privileges in Order Management*.
5. Set up the enterprise structure.
    - Use the default enterprise structure for a pilot set up.

- The Order Management work area displays sales orders for the business units that the current user can access. You must create a separate business unit for each business unit that will receive sales orders.
- A set is a collection of business units. Order Management uses sets to restrict access to holds and orchestration processes. You must specify a default set when you create a business unit. You can use the predefined Common Set for the default set.

6. Set up users who will do functional testing.

   An implementation user can access a wide range of privileges. To test with users who have fewer privileges, set up users with job roles that are specific to Order Management.

Role	Name
Job	Order Entry Specialist
Job	Order Manager
Job	Order Administrator
Abstract	Error Recovery

   Examine how these roles implement security. Create at least one user for each of these roles.

   - Only the Order Manager role
   - The Order Manager role and the Error Recovery role
   - The Order Administrator role

   For details, see *Privileges That You Need to Implement Order Management*.

Set Up Organizations and Items, and Import Items

1. Set up item organizations.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Organization Structures
        - Task: Manage Item Organizations
    - On the Manage Item Organizations page, set up your item organizations so Product Information Management can use them.
        - You need at least one organization that represents a warehouse where your implementation can collect the supply data it uses to fulfill each order.
        - You must set up each warehouse from a fulfillment system as an item organization in Oracle.
        - You must not associate an item organization with a business unit.

    For details, see *What's an item organization?* and *How can I create items in both master and child organizations?*.
2. Set up your units of measure.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Items
        - Task: Manage Units of Measure
    - Use the Manage Units of Measure page to set up your Units of Measure so the Product Information Management work area can use them.

      For details, see *How Units of Measure, Unit of Measure Classes, and Base Units of Measure Relate to Each Other*.
    - Set up Product Information Management. Product Information Management must contain the items that each test order references. For details, see *How You Set Up Items in Product Information Management*.
3. Define item classes, items, and catalogs according to your test requirements.
4. Use an order import template to import the items that your test orders will reference. Transform orders, as necessary. For details, see *Import Orders Into Order Management*.

Connect Source Systems and Set Up Customers

1. Optional. Set up and connect the source system.

    Here's when you must set up and connect the source system.
    - You plan to use a source system. For details, see *Integrate Order Management with Source Systems*.
    - You plan to import customer data from a source system.
    - You plan to use cross-references. For details, see *Overview of Creating Cross-References in Order Management*.

    If you will use only the Order Management work area to create sales orders, then you don't need to set up a source system.

2. Optional. Import customers.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Customers
        - Task: Import Person and Organization
    - On the Manage Data Import Batches page, click **Actions > Create** to create an import batch.
    - Use an Extract, Transform, and Load tool to load your data into the interface tables.

    You can use this import process to import a batch from the interface tables into Trading Community Architecture. The batch you import must include the customers your orders will reference. For details, go to *Implementing Receivables Credit to Cash*, then search for Data Import for Customers and Consumers.

Get Customer Details when Not Integrating or Importing

If you don't integrate with a source system or fulfillment system, or if you don't import customers, then create customers in one of these ways.

- **Create them in Oracle Financials.**
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Financials
        - Functional Area: Customers
        - Task: Create Customer

    For details, click the help icon on the Create Organization Customer page. Also, see *Oracle Trading Community Architecture User Guide*.
- **Synchronize them in Trading Community Architecture.** If the customer and customer entities that Order Management needs to fulfill the sales order don't exist, then Trading Community Architecture synchronizes the customer master with customer details from the sales order. Important details include sold-to party, ship-to party, and bill-to account.

Collect Data and Set Up Global Order Promising

For details, see *Collect Planning Data for Order Management* and *Set Up Promising Rules and Sourcing Rules for Order Management*.

Set up Order Management and Pricing, and Test Your Set Up

You must set up connections to order capture systems and fulfillment systems, then deploy the predefined data that specifies how Order Management fulfills each order.

1. Set up parameters according to your testing requirements.

    For details, see *Manage Order Management Parameters*.

2. Do the minimum setup tasks that a test environment requires.
    - In the Setup and Maintenance work area, go to the functional area.
        - Offering: Order Management
        - Functional Area: Orders
    - Do the setup tasks.

Task	Description
Manage Order Profiles	Set up values for these profile options. - Display Currency - Currency Conversion Type For details, see *Use Order Profiles to Control Order Management Behavior*.
Manage Orchestration Process Definitions	Set up and deploy these predefined orchestration processes. - ShipOrderGenericProcess - ReturnOrderGenericProcess For details, see *Guidelines for Setting Up Orchestration Processes*.

    For a complete list of tasks, see *Roadmap for Setting Up Order-to-Cash*.
3. Deploy the required processing constraints.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Processing Constraints
    - On the Manage Processing Constraints, search for the constraint, then verify that the Enabled attribute for it contains a check mark.

Attribute	Value
Constraint Name	UPDATE SHIPPING REQUEST VALIDATION You must deploy this constraint.

Attribute	Value

The validation rule sets in this constraint prevent Order Management from attempting to update the validation for each shipping request when certain conditions are true.

Validation Rule Set	Makes Sure That
Ordered Quantity Isn't Zero	The ordered quantity on the fulfillment line isn't zero. If its zero, then the line is in Canceled status and we don't want to process it.
Update Shipping Request Validation	Data validation doesn't fail for the Update Shipping fulfillment service because the flow doesn't provide values for all the required attributes.
Fulfillment Organization ID Is Null	The Warehouse attribute on the fulfillment line contains a value. Order Management uses the Warehouse attribute to populate the fulfillment organization ID.
Fulfill Line Request Dates are Null	The requested ship date or the requested arrival date on the fulfillment line contains a value. We can't calculate shipping dates if we don't have a value in one of these attributes.
Scheduled Ship Date on the Fulfillment Line Is Empty	The scheduled ship date on the fulfillment line contains a value. We can't calculate shipping dates if we don't have a value in this attribute.

- Enable or add other constraints to meet your needs, as necessary. For details, see *Manage Processing Constraints*.
- Click **Generate Packages**.

  This action deploys all enabled constraints into your implementation.

4. Optional. Set up the orchestration that your test environment requires.
   - Set up and deploy orchestration processes.
   - Set up status codes.
   - Manage change orders.
   - Set up jeopardy and planning.
   - Release and deploy orchestration processes.
   - Create processing constraints.

   For details, see the chapters that start with *Overview of Orchestration Processes* and *Use Order Profiles to Control Order Management Behavior*.

5. Set up pricing.

   If you use Order Management to create sales orders, and don't import them or integrate to a source system that contains price details, then you must set up pricing. Order Management requires details about pricing entities, such as the pricing strategy, to determine price for each item that you add to the sales order.

   You can't submit a sales order in Order Management without the price. For details about how to set up pricing, *Overview of Oracle Pricing*.

   In addition, it might be necessary to:

   - Set up the Item Validation Organization parameter.
   - Get the privileges that you need to administer pricing segments, pricing strategies, price lists, and so on.

6. Test your set up.

   - Use a tool of your choice, such as SOAP (Simple Object Access Protocol), to simulate sending a test order. Note the order number.
   - In the Order Management work area, use the order number to search for the order.
   - Confirm that Order Management received the order and is processing it.

Setting Up Other Features

Feature	Description
Drop Ship	The set up for drop ship depends on your business requirements. For details, see the section that starts with *Overview of Drop Ship in Order Management*.
Internal Material Transfer	Internal Material Transfers comes predefined as available. The predefined job roles provide access to this feature, but you must set up your own job roles. You can set an option in Supply Chain Orchestration that determines whether inventory routes the transfer order to Order Management. For details, see *Overview of Supply Chain Orchestration*.
Back-to-Back Shipping	Back-to-back shipping can create supply only after Order Management successfully submits the sales order to order fulfillment. Back-to-back shipping can then create supply in these ways. • Procure supply from a supplier that resides outside of your organization. • Produce or assemble supply at an in-house manufacturing location. • Transfer material from another warehouse. • Reserve on-hand supply. The fulfillment warehouse receives the supply, then back-to-back shipping ships it to your customer. For details, see *Overview of Back-to-Back Fulfillment*.
Configure to Order	Use Configure to Order to efficiently fulfill each configured item. Here are the parameters that you can set to control the configurator. • Allow Changes Through Configurator Validation • Configuration Effective Date • Halt Configurator Validation on First Error • Use Configurator for Order Import Validation

Feature	Description
	You can also use Configure to Order to set up a kit. A kit is a configured item that includes one or more configure options, but the Order Management work area doesn't allow the Order Entry Specialist to modify the configure options of a kit. • You set the subtype for a kit to KIT-SMC when you set up the configuration model. • Order Management sends the shippable lines of a kit as a shipment set to Global Order Promising. • The Order Management work area doesn't display values for On Hand and Item Availability for kits because Global Order Promising doesn't support the quick availability check feature for kits. For details, see *Modeling Configurations for SCM* and *Manage Order Management Parameters*.
Integrate Order Management with Upstream Source Systems	You use a file or web service to import source orders from a source system. For details, see the sections that start with: • *Overview of Importing Orders Into Order Management* • *Web Services That You Can Use to Integrate Order Management* • *Integrate Order Management with Source Systems*
Integrate Order Management with Downstream Fulfillment Systems	Use a web service to allow Order Management to communicate with a fulfillment system. Use a predefined web service or create a new one. For details, see the section that starts with *Overview of Connecting Order Management to Your Fulfillment System*.

Related Topics

- Set Up User Roles and Privileges in Order Management
- How the Order Orchestration and Order Promising Processes Use the Collected Planning Data
- Overview of Security Console
- How Units of Measure, Unit of Measure Classes, and Base Units of Measure Relate to Each Other

Overview of Implementing Order Management

Get started with your Order Management implementation. Opt into the offerings that meet your business requirements.

For details about how to manage the opt-in and set up offerings, see *Using Functional Setup Manager*.

Order Management Offering

Use the Order Management offering to automate order fulfillment and process your sales orders so they're timely, complete, and accurate.

Here are the functional areas that you can use with the offering.

Functional Area	Description
Orders	Do a wide variety of set up tasks for Order Management, from managing how you import your source orders, transforming source orders, integration, application behavior, business rules, orchestration, and so on.
Pricing	Set up pricing for Order Management. For example, define pricing parameters, totals, elements, bases, messages, matrix types, lookups, and so on.

To get the complete list of functional areas and features, use the Associated Features report when you plan your implementation.

Upgrade or Update

Get important details that you need to upgrade or update. For details, see *Performing Your Quarterly Update (Doc ID 2337485.1)*.

Related Topics

- Plan Your Implementation
- Overview of Functional Setup Manager

Business-to-Business Messaging

Overview of Business-to-Business Messaging in Order Management

Use Order Management as your central location when you interact with more than one channel.

Improve order capture and order fulfillment in the order-to-cash flow when you communicate a sales order between businesses.

The business-to-business flow uses Collaboration Messaging Framework to automate message flow so Order Management can receive and process each source order from a trading partner, then reply with an advance shipment notice after shipping successfully finishes.

Web services use Open Applications Group Integration Specification (OAGIS) messages in the payload that it uses to handle interactions that happen between Oracle Applications and each trading partner. You can use your existing Electronic Data Interchange (EDI) infrastructure with OAGIS. Use this configuration to receive each source order, then your trading partner and supplier can process it through order fulfillment.

Use the business-to-business flow to achieve results.

- Process purchase order
- Change purchase order
- Cancel purchase order
- Acknowledge purchase order

- Acknowledge a change in the purchase order

Use the business-to-business flow to realize benefits.

- Reduce cost
- Increase processing speed and accuracy
- Improve relationships between business partners
- Simplify setup and management

If a delay in supply happens, then Supply Chain Orchestration might send an update for the scheduled ship date to Order Management. Order Management updates the scheduled ship date but doesn't update the scheduled arrival date on the fulfillment line.

Order Management updates the scheduled ship date, scheduled arrival date, and shipping method only if you set up Global Order Promising to calculate shipping, such as how to calculate transit time. If you don't set it up, the scheduled dates and shipping method might not contain a value.

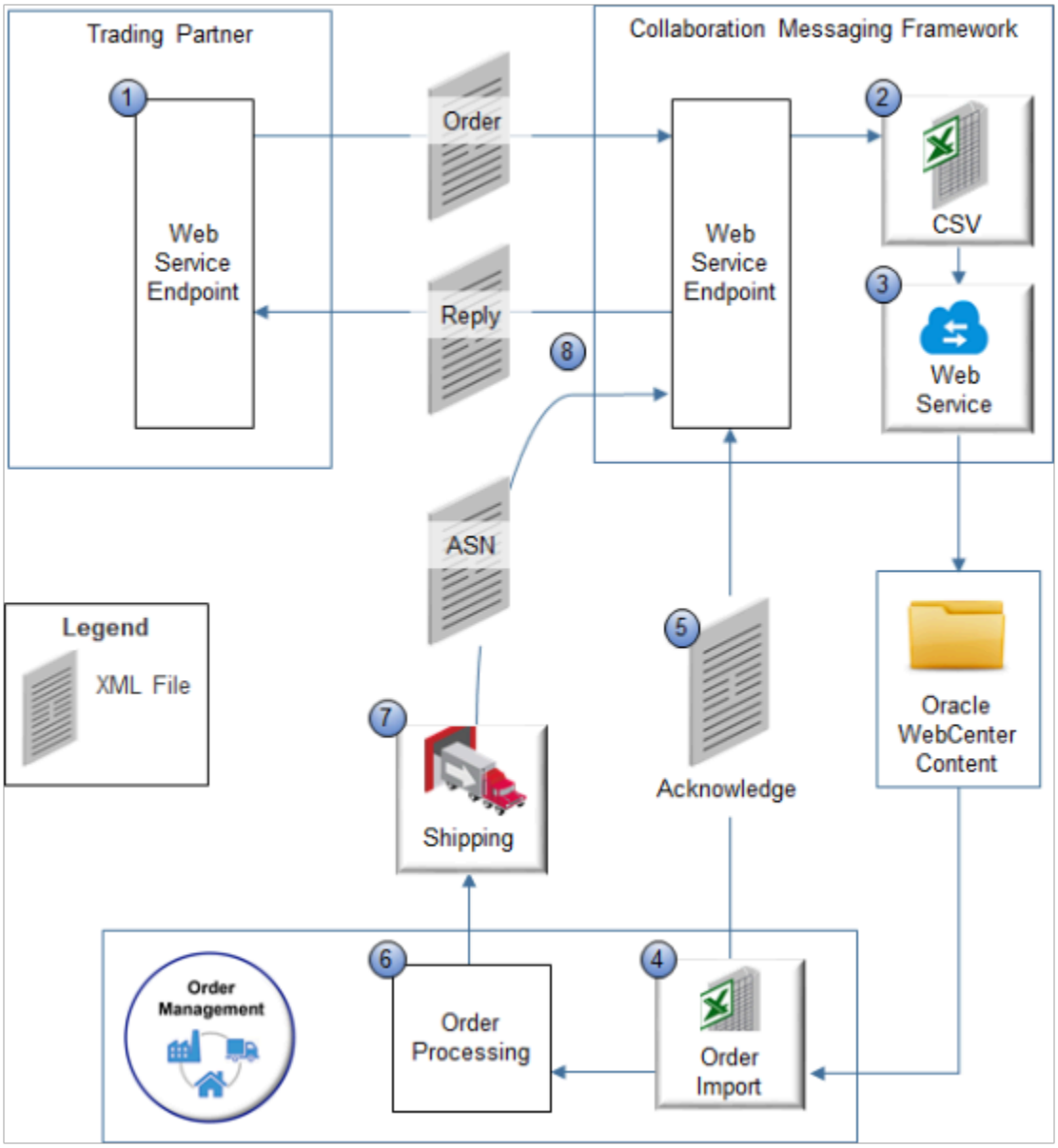
This behavior affects fulfillment, including choices that your end-users make when they override the schedule. For details, see *Schedule Fulfillment Lines Manually*.

Related Topics

- How Business-to-Business Messaging Works in Order Management
- Set Up Business-to-Business Messaging in Order Management

How Business-to-Business Messaging Works in Order Management

Collaboration Messaging Framework uses a web service to communicate sales order details between your trading partner and Order Management. It communicates these details in XML payloads.



Note

1. A trading partner uses a web service endpoint that they set up and enabled on their server to send an XML document that contains source orders to the CollaborationMessage web service on Collaboration Messaging Framework.
2. Collaboration Messaging Framework converts each source order to a CSV format (comma separated value) that Order Management supports.
3. Collaboration Messaging Framework uses a web service to upload the CSV files to a folder in Oracle WebCenter Content.
4. You use the order import template and a scheduled process to convert CSV files into sales orders.
5. Collaboration Messaging Framework subscribes to a business event that the scheduled process uses when it's done running. Order Management recognizes the event, then sends an order acknowledgment in an OAGIS message (Open Applications Group Integration Specification) through Collaboration Messaging Framework to the trading partner.
6. Order Management processes the sales order, then sends it to shipping for order fulfillment.

7. Oracle Shipping creates and processes the shipment.
8. Shipping finishes delivery, then sends the ship confirm and advance shipment notice in an OAGIS message, through Collaboration Messaging Framework, then in a reply to the trading partner.

Types of Messages That Collaboration Messaging Can Send and Receive

Collaboration Message	Description
OAGIS_10.1_ACK_PO_COLLAB_MSG_OUT	Send acknowledgment to the trading partner that Collaboration Messaging received the purchase order.
OAGIS_10.1_ACK_CHANGE_PO_COLLAB_MSG_OUT	Send acknowledgment to the trading partner that Collaboration Messaging received the purchase order change.
OAGIS_10.1_PROCESS_SHIPMENT_COLLAB_MSG_OUT	Send details to the trading partner about shipments.
OAGIS_10.1_PROCESS_RCV_DEL_COLLAB_MSG_OUT	Send details to the trading partner about purchase order deletions.

Here are the types of messages that Collaboration Messaging can receive from the trading partner.

Collaboration Message	Description
OAGIS_10.1_PROCESS_PO_COLLAB_MSG_IN	Process purchase orders.
OAGIS_10.1_CHANGE_PO_COLLAB_MSG_IN	Change purchase orders.
OAGIS_10.1_CANCEL_PO_COLLAB_MSG_IN	Cancel purchase orders.

Related Topics

- Overview of Business-to-Business Messaging in Order Management
- Set Up Business-to-Business Messaging in Order Management

Set Up Business-to-Business Messaging in Order Management

Use the Collaboration Messaging work area to set up the business-to-business flow in Order Management.

Assume you must create a relationship between your Computer Service and Rentals customer, who resides in Oracle Applications, and your Computer Associates trading partner.

- Allow Computer Service and Rentals to receive sales orders, updates, and cancellations from Computer Associates
- Allow Computer Service and Rentals to send acknowledgments and shipments to Computer Associates

Summary of the Steps

1. Identify the trading partner and messages to send and receive.
2. Create a relationship between trading partner and customer.
3. Simulate collaboration messaging.
4. Import source orders.
5. Process the sales order in shipping.
6. Examine the results.

This topic uses example values. You might need different values, depending on your business requirements.

Identify the Trading Partner and Messages to Send and Receive

1. Get the privileges that are in the B2B Messaging Administration Duty role, then sign into Oracle Applications.
   These predefined job roles have the privileges.
   - Order Entry Specialist
   - Order Manager
   - Order Administrator
   - Warehouse Manager

   This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.
2. Go to the Collaboration Messaging work area.
3. Click **Tasks > Manage Trading Partners**.
4. On the Manage Trading Partners page, click **Actions > Create**.
   Use Manage Trading Partners to communicate with a trading partner directly or through a service provider. For example, communicate through a service provider to set up a solution where the trading partner must use EDI (Electronic Data Interchange), a modified XML format, a proprietary format, and so on.
   In this example, you set up the configuration to communicate directly with the trading partner.
5. In the Create Trading Partner dialog, set the values, then click **Save and Close**.

Attribute	Value
Service Provider	None Computer Service and Rentals will communicate directly with Computer Associates instead of going through a service provider, so set this attribute to None.
Trading Partner ID	ComputerAssociates

Attribute	Value
Partner ID Type	Generic

6. Specify how to send messages to the trading partner.

   In the Delivery Methods tab, click **Actions > Add Row**, set the values, then click **Save and Close**.

Attribute	Value
Name	CMKDelivery00
Delivery Method	Web Service
Service Name	CollaborationMessage.Process
Not applicable	This integration uses the Process method of the CollaborationMessage web service. Make sure the trading partner already deployed this web service on their server.
Security Policy	None
Endpoint	Enter the URL that locates the server and port that the trading partner uses at their location as the end point for their web services. For example: `http://ComputerAssociates.com:7012`
User Name	Enter the user name that the trading partner server requires to access the endpoint.
Password	Enter the password that the trading partner server requires to access the endpoint.

7. Set up the messages that Collaboration Messaging sends to the trading partner.

   Click **Outbound Collaboration Messages**, then add these rows.

Name	Collaboration Message
CMKORDERACK001	OAGIS_10.1_ACK_PO_COLLAB_MSG_OUT
CMKORDERACK002	OAGIS_10.1_ACK_CHANGE_PO_COLLAB_MSG_OUT

Name	Collaboration Message
CMKSHIPCONFIRM001	OAGIS_10.1_PROCESS_SHIPMENT_COLLAB_MSG_OUT

Note

- Set the Status to Active for each message.
- Enter text that describes the message contents.

8. Click **Save**
9. Set up the messages that Collaboration Messaging receives from the trading partner.

   Click **Inbound Collaboration Messages**, then add these messages.

Name	Collaboration Message
CMKORDERIN001	OAGIS_10.1_CANCEL_PO_COLLAB_MSG_IN
CMKORDERIN002	OAGIS_10.1_CHANGE_PO_COLLAB_MSG_IN
CMKORDERIN003	OAGIS_10.1_PROCESS_PO_COLLAB_MSG_IN

   Set the Status to Active for each message.
10. Click **Save and Close**.

Create a Relationship Between Trading Partner and Customer

You will create a relationship between customer Computer Service and Rentals and the trading partner, Computer Associates. You also identify the documents that you must enable for the partner.

1. In the Collaboration Messaging work area, on the Overview page, click **Tasks > Manage Customer Collaboration Configuration**.
2. On the Manage Customer Collaboration Configuration page, search for the customer who will receive communications from the trading partner.

   For this example, enter `Computer Service and Rentals`. Assume you already set up Computer Service and Rentals as a customer. If you haven't, then you must do so now. For details, see *Import Customer Data for Your Sales Orders*.
3. In the search results, if more than one row exists for this customer, then click the row that includes a check mark in the Ship to Party option and the Collaboration Configured option, then click **Edit Collaboration Configuration**.

4. On the Edit Customer Collaboration Configuration page, in the Associated Service Providers area, click **Actions > Add Row**, then set the values.

Attribute	Value
Service Provider	None
Trading Partner ID	ComputerAssociates Note that you specified this ID earlier in this topic when you identified the trading partner.
Order Processing Business Unit	Specify the business unit that processes each sales order that Order Management receives. For this example, Computer Service and Rentals resides in the Vision Operations business unit, so select **Vision Operations**.
Application Partner Code	Accept the default value.

5. In the Collaboration Documents for Service Provider area, add these documents.
    - PROCESS_PO_IN
    - CHANGE_PO_IN
    - CANCEL_PO_IN
    - ACKNOWLEDGE_PO_OUT
    - ACKNOWLEDGE_CHANGE_PO_OUT
    - PROCESS_SHIPMENT_OUT

    Note
    - You use this area to specify the documents that this customer will communicate with the trading partner.
    - Set the Association Status to Enabled for each document.
6. Click **Save and Close**, then click **Done**.
    The customer can now communicate with the trading partner.

Simulate Collaboration Messaging

1. Click **Tasks > Validate Inbound Collaboration Messaging Setup**.
2. On the Validate Inbound Collaboration Messaging Setup page, set the values.

Attribute	Value
Service Provider	None
From Partner ID	ComputerAssociates

Attribute	Value
Not applicable	You will be testing a flow that sends a new source order from the trading partner, Computer Associates, so you use the Partner ID that you specified earlier in this topic.
External Message ID	Enter the number that identifies the message you're testing, such as `08192016_001`.
External Message Name	OAGIS_10.1_PROCESS_PO_COLLAB_MSG_IN
Processing Service	CollaborationMessage.Process

3. Click **Create Message Payload**.

   Notice that the Message Payload area uses the settings that you specified in step 2 to create, then display the XML that this test will use to communicate the message. You can modify the XML, if necessary.
4. Click **Process**.

   A separate web service in Collaboration Messaging simulates communication from the trading partner. It receives the XML document, validates it, then displays a Processing Confirmation dialog that includes the XML result of the test. If the test is successful, then the XML will include this code.

   ```
   <ProcessingResultCode><Success>/<ProcessingResultCode>
   ```
5. Make a note of the line that includes the message ID, such as.

   ```
   <BODID>IN_8001</BODID>
   ```

   In this example, `IN_8001` is the message ID.

Import Source Orders

The simulation created a simulated source order. Next, you import it order into Order Management and view it.

1. Use a scheduled process to import the source orders. For details, see *Import Orders Into Order Management*.
   - When you run the *Load Interface File for Import* scheduled process, set the Data File to the file that Messaging Framework created when you simulated collaboration messaging. The data file will include a concatenation of the document name plus the message ID. For example:

     ```
     PROCESS_PO_IN-1.0-IN_8001
     ```

   - Set the Source System to ORA_ELECTRONIC_DOCUMENTS when you run the *Import Sales Orders* scheduled process. Order Management uses this predefined source system for each source order it receives from the trading partner.
   - As an option, in the Batch Name parameter, specify the unique ID of the message you received, such as 08192016_001.
   - Order Management uses the setup from the Pricing work area to determine pricing. It will ignore any pricing data that the source order contains.

- When the Import Sales Orders scheduled process finishes, Order Management can process the sales orders through order fulfillment.

2. In the Order Management work area, on the Overview page, search for the simulated order, such as Demo_Order.
3. In the search results, note that the value in the Source Order attribute includes the order number you simulated, such as Demo_Order_0819_001.
4. In the Order column, click the **order number**.
5. On the Order page, notice that the order status is Processing, and the fulfillment line status is Awaiting Shipping, which indicates that Order Management released the sales order to order fulfillment.

   The messaging framework creates an acknowledgment message, then publishes it to the trading partner.

Process the Sales Order in Shipping

1. In the Shipments work area, on the Overview page, search for the sales order you noted in the Source Order attribute earlier, such as Demo_Order_0819_001.
2. On the Edit Shipment Line page, click **Cancel**.
3. On the Manage Shipment Lines page, click **Actions > Pick Release**, then click **Save and Close**.
4. On the Overview page, search for Demo_Order_0819_001.
5. On the Edit Shipment Line page, notice that the Line Status is Staged, then click the **link** next to Shipment.
6. On the Edit Shipment page, click **Ship Confirm**, then click **Save and Close**.

   Shipping creates an event. Collaboration Messaging subscribes to this event.
7. On the Manage Shipments page, in the row for your shipment, notice that the ASN Status is Sent.

   This status indicates that Shipping sent the ASN status to Collaboration Messaging.

Examine the Results in Collaboration Messaging

1. In the Collaboration Messaging work area, click Tasks, then click **Manage Collaboration Messaging History**.
2. In the Search area, set the value, then click **Search**.

Attribute	Value
Document	PROCESS_PO_IN

3. In the Messages area, in the External Message ID column, click the **row** that references the message you simulated earlier, such as 08192016_001.
4. Click **Actions**, then click a menu item.

Menu Item	Description
View Source Document	View the input XML that Collaboration Messaging received from the trading partner.
View Output Document	View the output XML that Collaboration Messaging converted from the input XML, and then sent to Order Management.

5. Click **Done**.
6. Click **Tasks > Manage Collaboration Messaging History**.
7. Examine the results of the acknowledgment.

   In the Search area, set the value, then click **Search**.

Attribute	Value
Document	ACKNOWLEDGE_PO_OUT

8. Examine the results of the shipment.

   In the Search area, set the value, then click **Search**.

Attribute	Value
Document	PROCESS_SHIPMENT_OUT

Related Topics

- Import Orders Into Order Management
- Overview of Business-to-Business Messaging in Order Management
- How Business-to-Business Messaging Works in Order Management

Various Setups

Security

Privileges That You Need to Implement Order Management

You need privileges to access the web services that Order Management uses to communicate with other systems.

> **Note:** This topic describes predefined job roles and duty roles to illustrate how job roles, duty roles, and privileges work together. However, you must create your own roles to meet your specific requirements. For details, see *Guidance for Assigning Predefined Roles* and *Security Reference for Order Management*.

You must make sure the job roles that you create can reference the duty roles that Order Management needs.

- A job role allows each user to access the application features that the user needs to do their job in their organization, and to access the data that these features reference. Here are some example, predefined job roles.
  - Order Entry Specialist

    - Order Manager
    - Order Administrator

- A duty role is a group of functions and privileges that represent one duty of a job. Each duty role is specific to an application.

Assume one of your users releases a sales order for shipping. The job role that this user uses references the duty roles that allow Order Management to call the shipping system and to receive details about the shipment from the shipping system. For example:

- Shipment Request Processing Web Service Duty
- Orchestration Order Shipping Web Service Duty

Identify the Privileges That You Need

Get details about the privileges that you need to do various administrative set up and run time tasks, and end user tasks in Oracle Order Management, Oracle Pricing, Oracle Global Order Promising, Oracle Supply Chain Orchestration, and Oracle Configurator.

Privileges control the work areas that you can access. They also control the elements that you can see and use, and the data that you can create, read, update, or delete in work areas and during import through channels.

Predefined job roles have many of the privileges that you need to fulfill each role. However, you must not use a predefined job role. Instead, you can create a copy of a predefined role, then modify the copy to meet your needs.

If you don't have all the privileges you need, you might still be able to access a work area but not see or use specific elements or tasks. You might be able to see some but not all data, or no data at all. So, if you copy a predefined job role and then edit it, you need to make sure you don't delete the privileges that your users need to fulfill their job function.

Try it.

1. Get the privileges that you need to use the security console.
2. Go to the Security Console work area.
3. Search for the job roles and privileges that you need. Here are some tips.

I Need Privileges To	Search For
Administer Oracle Order Management	- DOO_ - ORA_DOO_ORDER_ADMINISTRATOR_JOB
Use Oracle Order Management	- FOM_ - ORA_FOM_ORDER_ENTRY_SPECIALIST_JOB
Administer Oracle Pricing	- QP_ - ORA_QP_PRICING_ADMINISTRATOR_JOB
Administer or Use Global Order Promising	MSP_
Administer or Use Supply Chain Orchestration	DOS_
Administer Configurator Models	CZ_

4. Examine the search results. The Order Administrator job role has over 100 privileges. For example, here are 10 of them:
    - DOO_ADMINISTER_JEOPARDY_THRESHOLD_DEFINITION_PRIV
    - DOO_ADMINISTER_ORCHESTRATION_INFRASTRUCTURE_JEOPARDY_THRESHOLD_DEFINITION_PRIV
    - DOO_ADMINISTER_ORCHESTRATION_INFRASTRUCTURE_WEB_SERVICE_SOURCING_RULE_PRIV
    - DOO_ADMINISTER_TASKS_FOR_OPTIN_FEATURES_PRIV
    - DOO_ADMINISTER_WEB_SERVICE_SOURCING_RULE_PRIV
    - DOO_APPLY_ORCHESTRATION_ORDER_FULFILLMENT_LINE_HOLD_PRIV
    - DOO_APPLY_ORCHESTRATION_ORDER_HOLD_PRIV
    - DOO_APPLY_ORCHESTRATION_ORDER_LINE_HOLD_PRIV
    - DOO_ASSIGN_ORCHESTRATION_ORDER_LINE_TO_PROCESS_PRIV
    - DOO_CANCEL_ORCHESTRATION_ORDER_TASKS_PRIV

    Each privilege gives you access to some specific functionality. For example, DOO_APPLY_ORCHESTRATION_ORDER_HOLD_PRIV allows you to apply a hold on a sales order.

For more, see *Order Management Offering*.

Predefined Job Roles

Some job roles come predefined with Order Management.

Job Role	Responsibilities
Order Entry Specialist	• Create new sales orders, update existing sales orders, and create return sales order. • Create sales orders in the Order Management work area or modify sales orders that you import from a source system. • Monitor order fulfillment. • Work directly with customers who purchase items. This job role includes duty roles. • FSCM Load Interface Administration Duty • Item Inquiry Duty
Order Manager	• Manage sales orders that you create in the Order Management work area or that you import from a source system. Makes sure Order Management submits each sales order so it can proceed to order fulfillment. • Approve sales orders that require approval. • Work with other professionals in your organization, such as pricing administrator, customer contract manager, credit manager, and accounts receivable manager, to help determine set up requirements and troubleshoot problems. This job role includes duty roles. • Orchestration Order Monitoring Duty • Orchestration Order Management Duty • Orchestration Order Scheduling Duty

Job Role	Responsibilities
Order Administrator	• Set up and maintain Oracle Order Management so it supports order entry and order fulfillment. • Set up rules, policies, constraints, and so on. For example, set up defaulting rules, order entry preferences, order entry privileges, orchestration processes, change order rules, process planning, jeopardy conditions, order statuses, and hold definitions. • Set up orchestration. Each of these job roles include duty roles. • Orchestration Order Administration Duty • Orchestration Infrastructure Administration Duty

Duty Roles

Duty roles allow Order Management to communicate with a system that resides outside of Order Management. The Description column describes the communication that the web service provides.

Duty Role	Description
Order Orchestration Decomposition Web Service Duty	Communicate with source systems so Order Management can separate source orders during transformation.
Orchestration Order Activity Management Web Service Duty	Communicate with a fulfillment system that can process an activity.
Orchestration Order Billing Web Service Duty	Communicate with a system that processes the billing for each sales order.
Orchestration Order External Integration Web Service Duty	Allow a system that resides outside of Oracle Order Management to call Oracle Order Management.
Orchestration Order Fulfillment Web Service Duty	Allow a fulfillment system to send status updates to Order Management through a task service.
Orchestration Order Inquiry Web Service Duty	Extract order details from a sales order.
Orchestration Order Receiving Web Service Duty	Receive a return for a sales order.
Orchestration Order Shipping Web Service Duty	Process the shipment of a sales order.
Orchestration Order Template Web Service Duty	Allow a fulfillment system to send status updates for fulfillment tasks through a task service.

Job Roles That You Use to Administer Order Management

The predefined Order Administrator job role can do most set ups. The job roles that are available to you depend on how you set up security for your organization. Some organizations create a super user that allows you to access a variety of Oracle Application work areas. Other organizations prefer to keep this access separate, depending on your organization's security goals.

Here are some predefined roles and the work areas that they can access.

Work Area	Job Role	Details
Product Information Management	Product Manager	*Security Reference for Product Management*
Order Promising	Order Administrator	*Security Reference for Order Management*
Inventory Management	Inventory Manager	*Security Reference for Manufacturing and Supply Chain Materials Management*
Work Definitions	Manufacturing Engineer	*Security Reference for Manufacturing and Supply Chain Materials Management*
Suppliers	Supplier Manager	*Security Reference for Oracle Procurement Cloud*
Purchase Orders	-	*Security Reference for Oracle Procurement Cloud*
Shipping	Shipping Agent	*Security Reference for Order Management*
Financials offering in the Setup and Maintenance work area. For example, to integrate Order Management with Oracle Receivables.	Financial Application Administrator	*Security Reference for Financials*
Accounts Receivable	Accounts Receivable Manager	*Security Reference for Financials*

Other Roles and Privileges

What You Need to Do	Description
Integrate Order Management with Global Trade Management	The DOO_MANAGE_ORCHESTRATION_ORDER_TRADE_COMPLIANCE_INTERFACE_WEB_SERVICE privilege can set up the integration. For details, see *Integrate Order Management with Global Trade Management*.
Limit access in Oracle Transactional Business Intelligence to the Order Management, Fulfillment Lines Real Time subject area.	The Order Transaction Analysis Duty and Orchestration Process Transaction Analysis Duty role can access this functionality.

What You Need to Do	Description
Use the Create Document action. For example, go to the Order Management work area, then, on the Create Order page, click **Actions > Create Document > View**.	Make sure you have the FOM_PRINT_ORDER_PRIV_OBI privilege. It allows you to use the Create Document action. The predefined Order Entry Specialist job role and the Order Manager job role include this privilege.

Set Up User Roles and Privileges in Order Management

Set up user roles and privileges to manage the authentication and authorization that Order Management uses to secure Order Management processing, including web service usage.

> **Note:** This topic describes predefined job roles and duty roles to illustrate how job roles, duty roles, and privileges work together. However, you must create your own roles to meet your specific requirements. For details, see *Security Reference for Order Management*

Here's how Order Management implements security.

- Uses authentication through a user name and password during sign in to allow each user to access the Order Management work area
- Uses authorization through user roles and privileges to allow each user to do different tasks according to job outcome in the Order Management work area

This topic describes how to examine privileges and job roles that come predefined with Order Management, and how to add an Order Management user. For background details, see:

Book	Details
Securing SCM	Job roles, privileges, duty roles, and how to set up security, including values that you set for each user.
Security Reference for Order Management	Job roles that come predefined with Order Management.

The examples in this topic use predefined job roles. You must create your own job roles, depending on your security requirements.

Here are some example roles.

Role	Description
Pricing Administrator- All Business Units, which is QP_PRICING_ADMINISTRATOR_ALL_BUSIN	Administers pricing.
Product Manager, which is EGP_PRODUCT_MANAGER_JOB	Sets up organizations and items so you can add items to your sales orders.

Summary of the Set Up

1. Create users and assign job roles.
2. Create a job role.
3. Manage data access for users.

This topic uses example values. You might need different values, depending on your business requirements.

Create Users and Assign Job Roles

Create two users and assign job roles. One user can use administrative privileges. The other user can use only view privileges.

1. Make sure you have the privileges that you need to manage job roles.

   If you don't sign in with these privileges, then various actions will be grayed out when you do the Create Implementation Users task, such as copying a job role, and you won't be able to add privileges to a job role.
2. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Initial Users
   - Task: Create Implementation Users
3. On the User Accounts page, click **Add User Account**, enter values, then click **Add Role**.

Attribute	Value
First Name	Diane
Last Name	Cho
Email	`diane.cho@yourComany.com`

4. In the Add Role Membership dialog, enter `Order Manager`, then click **Search**.
5. In the search results, click the **row** that contains the values.

Attribute	Value
Name	Order Manager
Code	ORA_DOO_ORDER_MANAGER_JOB

6. In the Confirmation dialog, click **Add Role Membership > OK**, then click **Done**
7. On the Add User Account page, add passwords for the user, then click **Save and Close**.

   Each user can use these passwords the first time the user signs in. Instruct your users to change passwords immediately after sign in.

Create another user and assign a job role so the user can only view sales orders, but not create, update, or delete them.

1. Identify the role that provides only view access to sales orders.
    - Go to *Security Reference for Order Management*.
    - Examine the roles, duties, privileges, and policies until you locate one that meets your needs. For this example, the Order Entry Specialist is the most likely role.
    - In the Order Entry Specialist section, scroll through the Privileges area until you locate the Item Inquiry granted role.

Granted Role	Description	Privilege
Item Inquiry	Queries and views items in the enterprise.	Manage Item Attachment
Item Inquiry	Queries and views items in the enterprise.	Manage Item Catalog
Item Inquiry	Queries and views items in the enterprise.	Manage Item Global Search
Item Inquiry	Queries and views items in the enterprise.	Manage Trading Partner Item Reference
Item Inquiry	Queries and views items in the enterprise.	View Item
Item Inquiry	Queries and views items in the enterprise.	View Item Organization Association
Item Inquiry	Queries and views items in the enterprise.	View Item Relationship

    - For this example, you must provide only read access, so you will use the View Item privilege.
2. On the User Accounts page, click **Add User Account**, set the values, then click **Add Role**.

Attribute	Value
First Name	Yu
Last Name	Li
Email	`yu.li@yourComany.com`

3. In the Add Role Membership dialog, enter the role you located earlier in this procedure, Order Entry Specialist, then click **Search**.

4. In the search results, click the **row** that contains the values.

Attribute	Value
Name	Order Entry Specialist
Code	ORA_FOM_ORDER_ENTRY_SPECIALIST_JOB

5. In the Confirmation dialog, click **Add Role Membership > OK**, then click **Done**.
6. On the Add User Account page, add passwords for this user, then click **Save and Close**.

Create Job Role

As an option, you can create a job role to meet your business requirements. In this example, you create a job role that allows Yu to view sales orders but not edit them.

1. On the User Accounts page, click **Roles**.
2. On the Roles page, in the Search window, enter `Order Entry`, then click **Search**.
3. In the search results, in the row that contains these values, click **Actions > Copy Role**.

Attribute	Value
Name	Order Entry Specialist
Code	ORA_FOM_ORDER_ENTRY_SPECIALIST_JOB

> **Tip:** Reduce your work load. Modify the copy of a predefined role rather than create a new one.

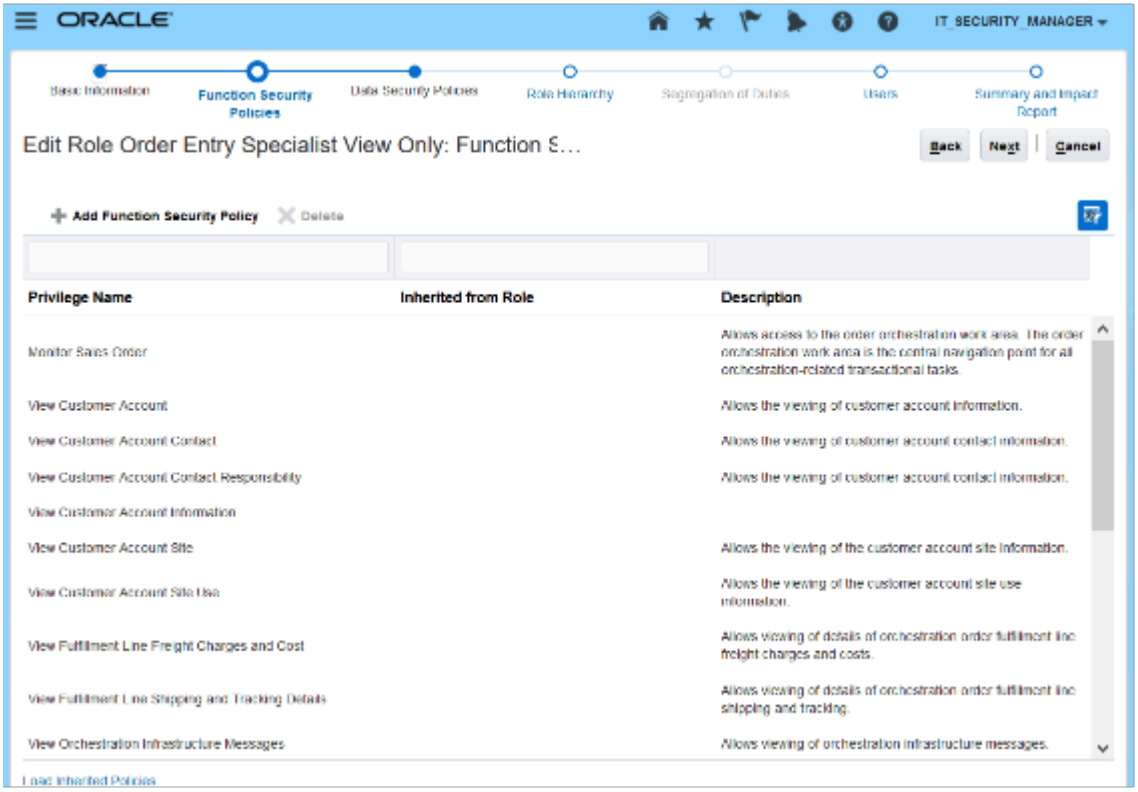
4. In the Copy Options dialog, select Copy Top Role, then click **Copy Role**.
5. On the Basic Information page, enter values, then click **Next**.

Attribute	Value
Role Name	Order Entry Specialist View Only
Role Code	FOM_ORDER_ENTRY_SPECIALIST_JOB_VIEW_ONLY
Description	Search for and view sales orders, including sales order header, order lines, price details, and price totals. Don't allow user to create, update, or delete any part of the sales order.

6. On the Function Security Policies page, delete all rows except rows that contain these privileges.
    - Monitor Sales Order
    - View Customer Account

- View Customer Account Contact
- View Customer Account Contact Responsibility
- View Customer Account Information
- View Customer Account Site
- View Customer Account Site Use
- View Fulfillment Line Freight Charges and Cost
- View Fulfillment Line Shipping and Tracking Details
- View Item
- View Item Organization Association
- View Item Relationship
- View Orchestration Infrastructure Messages
- View Orchestration Order Fulfillment Line Hold
- View Orchestration Order Hold
- View Orchestration Order Line Hold
- View Orchestration Process Details
- View Orchestration Process Hold
- View Order Orchestration Request Messages
- View Orders
- View Planning Supply Availability
- View Supply Availability Report
- View Supply Chain Financial Orchestration System Options
- View Supply Orders

For example:



Note

- If you must add a privilege, then click **Add Function Security Policy**.
- If you must add all privileges, for example you select to not copy a predefined role, then, in the Add Function Security Policy dialog, enter the first characters that are similar across a group of privileges, such as View Customer, click Search, then add each privilege from the search results.

7. On the Data Security Policies page, delete each row that includes these values.

Attribute	Values
Policy Name	- Grant on Collaboration Document Header - Grant on Business Unit

To delete a row, click the **down arrow** in the row, then click **Remove Data Security Policy**.

8. Verify that the Data Security Policies page displays these policies.

Policy Name	Policy Description	Privilege
Grant on Trading Community Customer Account Site Use	Order entry specialist can view customer account site use.	Read, View Customer Account Site Use

Policy Name	Policy Description	Privilege
Grant on Trading Community Relationship	Order entry specialist can view trading community relationship.	Read, View Trading Community Relationship
Grant on Trading Community Organization Party	Order entry specialist can view trading community organization.	Read, View Trading Community Organization
Grant on Application Reference Data Set	Order entry specialist can view customer account site use.	View Customer Account Site Use
Grant on Application Reference Data Set	Order entry specialist can view customer account site.	View Customer Account Site,
Grant on Trading Community Party	Order entry specialist can view trading community person.	Read, View Trading Community Person
Grant on Trading Community Customer Account	Order entry specialist can view customer account.	Read, View Customer Account
Grant on Application Reference Data Set	Order entry specialist can view customer account relationship.	View Customer Account Relationship
Grant on Trading Community Customer Account Site	Order entry specialist can view customer account site.	Read, View Customer Account Site

Specify the Create, Read, Update, and Delete Actions

Specify the actions that each policy allows. For example:

- In the Policy Name column, in the row that contains this value, click **Actions > Edit Data Security Policy**.

Attribute	Value
Policy Name	Grant on Trading Community Customer Account Site Use

- In the Edit Data Security Policy dialog, next to Actions, click the **down arrow**, then add or remove the check mark next to each of the actions you will allow or disallow the user to do.

  For example, View Customer Account Site, Manage Customer Account Site, Read, Delete, Update, and so on. For this example, you're setting up a read-only view, so make sure only the view actions and read actions contain a check mark.

Specify Access According to a Group of Business Units

A business unit set as a group of business units that you can use for a specific setup. For example, assume you add business unit 1 and business unit 2 to business unit set x, and then attach `Payment Term NET30` to set x. You can then use this payment term for business unit 1 and business unit 2.

The Set Id identifies the business unit set. For details, go to *Implementing Sales*, then search for Overview of Multiple Business Units in Sales.

You can specify the actions that each policy allows according to Set Id. For example:

- In the Policy Name column, in the row that contains this value, click **Actions > Edit Data Security Policy**.

Attribute	Value
Policy Name	Grant on Application Reference Data Set

- In the Edit Data Security Policy dialog, next to Actions, click the **down arrow**, then notice you can specify a wide range of views and manage actions that the user can perform.

Specify Access According to Business Unit

You can specify the actions that each policy allows according to business unit. For example:

- Click **Create Data Security Policy**.
- In the Create Data Security Policy dialog, click **Search**.
- In the Search Database Resources dialog, enter `Business Unit`, click **Search**, wait for the results to display, then click **OK**.
- Set the values.

Attribute	Value
Policy Name	Grant on Business Unit
Data Set	Select by Instance Set You can also use Select by Key to specify the business unit according to BU_ID.
Condition Name	Specify how to filter according to business unit. For most situation, select **Access the Business Unit for Which the User is Explicitly Authorized**.

Attribute	Value
Actions	Select the actions that you must allow the user to do for the business unit.

9. Click **Next**.
10. On the Role Hierarchy page, delete all role hierarchies except for these:

Role Name	Role Code
Item Inquiry	ora_egp_item_inquiry_duty
Item Inquiry	ora_egp_item_inquiry_duty_hcm
Item Inquiry	ora_egp_item_inquiry_duty_obi
Item Inquiry	ora_egp_item_inquiry_duty_crm
Manage Item Catalog	egp_manage_item_catalog_priv_obi
Print Order	fom_print_order_priv_obi

Use the Role Hierarchy page to specify other job roles that the job role you're creating can access. A role hierarchy is a hierarchy that specifies other job roles that a job role references.

For example, here's the predefined role hierarchy that the Order Entry Specialist job role uses.

```
Order Entry Specialist

 B2B Messaging Administration

 Collaboration Messaging Manager

 Collaboration Messaging Setup

 SOA Infra Designer

 FSCM Load Interface Administration

 Item Inquiry

 Upload data for Source Sales Order Import
```

For details about the role hierarchy that each predefined job role uses, see the Security Reference for Order Management book.

11. Click **Next**.
12. On the Users page, click **Add User**.

13. In the Add User dialog, search for Yu Li, wait for the results to display, click **Add User to Role > OK** in the confirmation dialog, then click **Cancel**.
14. Click **Next > Save and Close**.

Manage Data Access for Users

Manage data access for Yu.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Initial Users
    - Task: Manage Data Access for Users

    For details, see *Implementing Common Features for Oracle SCM*.
2. On the Manage Data Access for Users page, enter a value, then click **Search**.

Attribute	Value
User Name	`yu.li` You must search according to dot notation, which is firstName.lastName.

    The search results display the data access you set up for Yu, including for the Order Entry Specialist role where you added Yu as a user on the User Accounts page, and the other job roles you specified when you created the Order Entry Specialist View Only job role, and then assigned to Yu.
3. Click **Authorize Data Access**.
4. In the Opening SecurityDataAccessTemplate.xls dialog that displays, accept the Open With option, then click **OK**.

    Microsoft Excel opens.
5. Edit in Microsoft Excel.
    - In Microsoft Excel, in the Connect dialog, click **Yes**.
    - On the Login page, sign in with the privileges that you need to manage IT security.
    - In the Authorize Data Access for Users template that displays, verify that the template includes the security contexts that Yu needs for view access.

Security Context	User Name	Role
Business unit	`li.yu@yourComany.com`	Order Entry Specialist
Data access set	`li.yu@yourComany.com`	Order Entry Specialist View Only
Asset book	`li.yu@yourComany.com`	Order Entry Specialist View Only

Security Context	User Name	Role
Business unit	`li.yu@yourComany.com`	Order Entry Specialist View Only
Control budget	`li.yu@yourComany.com`	Order Entry Specialist View Only
Cost organization	`li.yu@yourComany.com`	Order Entry Specialist View Only
Intercompany organization	`li.yu@yourComany.com`	Order Entry Specialist View Only
Ledger	`li.yu@yourComany.com`	Order Entry Specialist View Only
Manufacturing plant	`li.yu@yourComany.com`	Order Entry Specialist View Only
Inventory organization	`li.yu@yourComany.com`	Order Entry Specialist View Only
Project organization classification	`li.yu@yourComany.com`	Order Entry Specialist View Only
Reference data set	`li.yu@yourComany.com`	Order Entry Specialist View Only

- In the Security Context Value column, in the first row that contains data, right-click the **cell**, then click `Invoke Action`.

  > **CAUTION:** Use this action instead of manually entering text. This action searches the Oracle database for the data access sets you can use. If you manually enter text, and if your text doesn't exactly match text that the database contains, then the upload will fail.

- In the Select Security Context Value dialog, set the value, then click **Search**.

Attribute	Value
Business Unit	Vision Operations

- In the search results, click the **row** that includes Vision Operations, then click **OK**.

  Notice that Excel adds Vision Operations to the cell you selected in the Security Context Value column.
- Repeat the above steps for each of the other rows that contain data.

  For example, for the row that contains Asset Book, set value Security Context to an asset book.
- In the command ribbon that displays across the top of Excel, click **Authorize Data Access for Users > Upload**.

- Wait for the upload to finish, then verify that the Status column displays `Successfully Uploaded for each row.`
- Click **Status Viewer**, then verify that the Status View displays `No Error`.
- Sign out.

6. Go back to Oracle Applications.
7. Go to the Scheduled Processes work area.
8. On the Scheduled Processes page, click **Schedule New Process**, then run the scheduled process.

Scheduled Process Name	Description
Retrieve Latest LDAP Changes	Synchronizes users, roles, and role grants with the definitions that exist in LDAP (Lightweight Directory Access Protocol) that Order Management uses to determine who can access the Order Management work area.

Examine Role Usage in Your Implementation Project

Your implementation project specifies the roles that can do for each task in the project. You will examine how a predefined implementation project allows the Order Administrator role to manage source systems where you typically use web services to communicate data.

1. Go to the Setup and Maintenance work area.
2. On the Setup page, click **Tasks**, then click **Manage Implementation Projects**.
3. On the Manage Implementation Projects page, click **Actions > Create**.
4. On the Create Implementation Project page, click **Next**.
5. In the Order Management row, add a check mark in the Include column, then click **Save and Open Project**.
6. In the Task list, expand **Order Management > Define Orders**, then, in the Manage Upstream and Fulfillment Source Systems row, in the Authorized Roles column, click **Details**.
7. Notice that the Authorized Roles dialog includes the Order Administrator role.

Related Topics

- Overview of Multiple Business Units in Sales

General

Opt Into Features in Order Management

Get details about features you can enable or disable according to your business requirements.

1. Go to the Setup and Maintenance work area, then go to the offering.
   - Offering: Order Management
2. Click **Change Feature Opt In**.

Examine the features that you can use with Order Management.

Feature	Description	Detail
Maintain Common Reference Objects	Support common functionality, such as data security, reference data sets, and other preferences. Leave this feature enabled for most deployments.	Go to *Implementing Applications*, then search for Overview of Common Reference Objects
Governance, Risk, and Compliance	Manage governance, risk, and compliance across processes and systems.	*Oracle Governance, Risk, and Compliance*
Local Installation of Help	Write your own help.	Go to *Implementing Common Features for SCM*, then search for Set Up Help.
Transformation Rules	Transform source orders that you create in Order Management or that you import from a source system to optimize order fulfillment. For most implementations, leave this feature enabled.	*Create Transformation Rules*
Order Holds	Allow the Order Entry Specialist to place a hold on a sales order. For most implementations, leave this feature enabled.	*Guidelines for Setting Up Holds on Sales Orders*
Drop Ship	Leave this feature disabled unless you implement drop ship.	*Indicate an Ownership Change During Drop Ship*
Enterprise Structures Guided Flow	Leave this feature disabled unless you implement enterprise structures.	Go to *Implementing Common Features for Oracle SCM*, then examine the Enterprise Structures chapter.
Application Toolkit Component Maintenance	Leave this feature disabled unless you use Application Toolkit to modify various components, such as application help, reports and analytics, the Watchlist, and so on.	Go to *Implementing Common Features for Oracle SCM*, then examine the Application Toolkit Configuration chapter.
Click to Dial	Use Click to Dial to place a call to a contact from a hyperlink on the phone number or phone icon.	Not applicable
Enterprise Scheduler Job Definitions and Job Sets	Leave disabled unless you run scheduled processes according to Oracle Enterprise Scheduler Services.	Go to *Implementing Common Features for Oracle SCM*, then examine the Enterprise Scheduler Job Definitions and Job Sets chapter.
Enable Custom Payloads for Downstream Integration	Use view objects to integrate with systems that reside downstream of Order Management.	*Use a Service Mapping to Integrate Order Management with Other Oracle Applications*
Specify Business Unit for Selling Profit Center for Goods and Services Tax	Set the selling profit center on a sales order line that's different from the business unit on the order header.	*Set Up Business Units for Selling Profit Centers*

Feature	Description	Detail
Enable Coverage and Subscription	Include coverage items and subscription items in sales orders.	*Set Up Coverages for Sales Orders*
Update Selected Lines in Sales Order	Allow the Order Entry Specialist to select one or more order lines in a sales order, specify the attributes and values to update, then update all selected lines. You must make sure your user has the Update Selected Lines on Sales Orders privilege.	*Update More Than One Order Line*
Return Items or Cancel Services Without a Reference Order	Allow the Order Entry Specialist to return an item without specifying the sales order that originally ordered the item.	*Allow Users to Return Items Without the Original Sales Order*

Enable or disable other features.

Feature	Description	Details
Pricing	Use pricing parameters to specify some of the pricing behavior for sales orders.	*Overview of Oracle Pricing*
Order Management Business Intelligence Analytics	Enable analytic reporting in the Order Management work area. Leave this feature enabled.	*Use Reports and Analytics with Order Management*

Related Topics

- Plan Your Implementation
- Configure Offerings

Identify Hosts and Ports for Order Management

Identify the hosts and ports that Order Management uses to communicate data. You use these details during setup and maintenance.

1. Go to the Setup and Maintenance work area.
2. On the Setup page, click **Tasks > Review Topology**.
3. On the Review Topology page, click **Detailed**.
4. Get details for your Service Oriented Architecture (SOA).
    - Expand **SCMDomain**.
    - In the SCM-SOA row, examine the values for External Server Host and External Server Port.

5. Get details for services you can use with Oracle Applications.
    - Expand **FADomain**, then examine values.

Name	Values
FSCMServices	Host and port for Application Development Framework (ADF) services.
FASOA	Host and port for SOA services.

    - In the SCM-SOA row, examine the values for External Server Host and External Server Port.

Manage Order Management Parameters

Set up parameters that affect behavior across all of Order Management.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Parameters
2. On the Manage Order Management Parameters page, set your values.

Parameters

Parameter	Description
Activate Credit Check on Order Submit	Specify whether to run credit check when the Order Entry Specialist submits the sales order. For details, see *Guidelines for Setting Up Credit Check*.
Allow Changes Through Configurator Validation	Allow the configurator to select items in a configuration and to modify configuration options after the Order Entry Specialist adds a configured item to a sales order, and then saves the sales order as a draft, but before submitting it. For example: 1. The Order Entry Specialist adds a configured item to a sales order, then saves the sales order as a draft. 2. A configuration manager modifies the configuration model and configuration rules for the configured item in such a way that it affects the configuration that the Order Entry Specialist added. 3. The Order Entry Specialist returns to the draft several days later, then submits the sales order. In this example, if you set Allow Changes Through Configurator Validation to:

Parameter	Description
	• **Yes.** The configurator uses the modified configuration model and configuration rules to update the configuration that the Order Entry Specialist created. It allows the submit to proceed. It doesn't report these modifications to the Order Entry Specialist. This setting is useful when your business process doesn't require the Order Entry Specialist to understand and agree to modifications that the configurator makes. • **No.** The configurator doesn't modify the configuration that the Order Entry Specialist created. Instead, it doesn't allow the submit to proceed, and displays a validation error when the Order Entry Specialist attempts to submit or validate the sales order. This setting is useful when your business process requires that the Order Entry Specialist understand and agree to modifications that the configurator makes. Note • This parameter doesn't apply to a source order that includes a configured item. For details, see *Import Orders Into Order Management*. • This parameter affects all business units. See *Configurator Models*.
Allow User to Remove Project Details on Referenced Returns	A referenced return order gets attribute values from the original sales order that it references. If you set this parameter to Yes, then Order Management allows the Order Entry Specialist to remove these original values from project attributes on a referenced return order, but only if the return order is in Draft status.
Automatically Set Values on Sales Agreement Attributes	Control default values that Order Management sets for sales agreement attributes when you create or update the sales order. If you set this parameter to Yes, and if: • Only one sales agreement exists for the customer and business unit, then Order Management sets the agreement on the order header to this one sales agreement. • More than one agreement exists, then Order Management leaves the agreement on the order header empty. • The agreement on the order header is empty, and if only one sales agreement exists for the item on the order line, then Order Management sets the Sales Agreement, Sales Agreement Line, and Sales Agreement attributes on the order line to values from the agreement. Otherwise, it leaves this order line attributes empty. If you set this parameter to No, then Order Management doesn't set any default values. It leaves them all empty. Order Management constrains the customer, business unit, and currency on the order header. It also constrains the item on the order line.
Business Unit for Selling Profit Center	See *Set Up Business Units for Selling Profit Centers*.
Cancel Supply When Canceling Backorders	Use this parameter to control behavior in your back-to-back flow. Set a value:

Parameter	Description
	• **Yes**. If Shipping canceled only part of the shipment line, then the *Process Responses from Order Fulfillment* scheduled process will update Oracle Supply Chain Orchestration's supply order. It will also cancel or close any transfer order, purchase order, or work order that's attempting to fulfil the remaining quantity. • **No**. Don't update or cancel the supply order, and don't cancel or close the transfer order, purchase order, or work order. Set to No when you want to manually review and cancel the supply order. If Shipping finishes the shipment, and if Shipping shipped all the quantity on the order line, then the scheduled process will close the supply order even if you set this parameter to No. See *Cancel Backorders and Enforce Single Shipments*.
Check for Trade Compliance When User Submits Sales Order	Set a value. • **Yes.** Order Management will verify each sales order that the Order Entry Specialist submits in Order Management or that you import from a source system. • **No.** Order Management won't verify any sales orders. See *Overview of Setting Up Trade Compliance*.
Compare Change Order to Fulfillment Values	See *Compare Change Order to Fulfillment Values*.
Configuration Effective Date	See *Manage Your Validations*.
Configuration Effective Date for Exploding Included Items	Use this parameter with or without setting Configuration Effective Date. See *Control Explosion Dates for Configuration Models*.
Coverage Start Date	Select a value. • **Shipment Date**. Set the start date of the coverage to the date when Order Management ships the covered item. • **Delivery Date**. Set the start date of the coverage to the date when you shipping system delivers the covered item at your customer's site. For details, see *Set Up Coverages for Sales Orders*.
Credit Check Failure at Order Submit	Specify how to proceed if credit check fails. • **Save Order in Draft Status.** Save the sales order in Draft status and don't proceed to fulfillment. • **Submit the Order with Hold on Lines That Failed Credit Check.** Save the sales order in Submitted status, place a hold on each order line that fails credit check, then proceed with fulfillment for order lines that aren't on hold. See *Guidelines for Setting Up Credit Check*.

Parameter	Description
Customer Relationship Type	Specify the values that the Order Entry Specialist can set for the Ship-to Customer attribute and the Bill-to Customer attribute. Set a value. • **Single Customer.** Allow the Order Entry Specialist to select only the bill-to customer and the ship-to customer that's the same as the sold-to customer. For example, if the Order Entry Specialist creates a sales order for Company x, then Order Management sets Ship-to Customer to Company x and Bill-to Customer to Company x. The Order Entry Specialist can't modify these values. • **All Customers.** Allow the Order Entry Specialist to select any bill-to customer or ship-to address, regardless of the sold-to customer. For example, if the Order Entry Specialist creates a sales order for Company x, then Order Management sets Bill-to Customer to Company x and Ship-to Customer to Company x, by default, and allows the Order Entry Specialist to search and select any bill-to customer, or any ship-to customer and address.
Days to Retry Recovery for Business Events	Specify the number of days to recover from errors that involve business events when you run the Recover Errors scheduled process. The Recover Errors scheduled process attempts to retry errors that fail in the background. These errors might add up over time and cause a performance problem. You can use this parameter to put a limit on the number of days that you want to retry recovery. For example, set it to 6, and the scheduled process will only try to recover errors that are up to 6 days old, and will ignore any error that's more than 6 days old. This parameter comes predefined with a value of 5 days, by default. You can use a maximum value of 30 days. If you don't want to retry recovery, the set it to 0. Increase the value only if you find you need more time to recover from an error that involves a business event. For example, assume you have a fulfillment line that's in error because you used the wrong URL when you set up your connector 9 days ago. You now need to process that line, so you set the parameter to 10. For details, see *Fix Errors in All Sales Orders* and *Overview of Using Business Events with Order Management*.
Enable Orchestration Process Planning and Calculate Jeopardy	Specify whether Order Management will plan your orchestration process and calculate jeopardy on each sales order when it runs your orchestration process. See *Guidelines for Setting Up Orchestration Processes*. Querying data for the Fulfillment lines in Jeopardy infolet and for the Orders in Jeopardy infolet on the Overview page in the Order Management work area can sometimes affect performance. If you don't use jeopardy, then set this parameter to No, and Order Management won't query for and won't display data in these infolets.

Parameter	Description
Filter Ship-To Addresses by Ship-to Usage	Filter addresses that the Order Entry Specialist can select when setting the Ship-to Address attribute on the order header. Set a value. • **Yes**. Display only ship-to addresses. • **No**. Display all addresses. Order Management comes predefined to use No, by default. Examine the behavior when Filter Ship-To Addresses by Ship-to Usage is No. 1. Create a sales order in the Order Management work area. 2. On the order header, click **Ship-To Address**, then click **Search**. 3. In the Search and Select dialog, don't set any values, then click **Search**. Notice that the search returns addresses that include Ship-to and Bill-to values in the Usage attribute. Note • In some situations, you might need to prevent the Order Entry Specialist from using a Ship-to Address when the usage for the address is Bill-to. For example, if you use Oracle Receivables, then you must set Filter Ship-To Addresses by Ship-to Usage to Yes because Receivables come predefined to ship only to a ship-to address. In another example, it might not be appropriate for a bill-to location, such as an accounting center, to receive shipment. • If you import a source order, and if the Filter Ship-To Addresses by Ship-to Usage parameter is Yes, and if the value of the Site Usage attribute for the ship-to address in Trading Community Architecture isn't Ship-to, then the import will fail. • To specify the value that the Usage attribute displays, navigate to Trading Community Architecture, then set the Site Usage attribute to one of these values. o Ship-to o Bill-to o Ship-to and Bill-to See *Oracle Trading Community Architecture Technical Implementation Guide*.
From Address for Email Messages	Specify the From email address that the Universal Messaging Service uses when it sends an email from the order document or from an automatic notification. The From address comes predefined as `no-reply@orderreport.com`. You can modify this value to use an address in the domain that your company uses. You must use a valid email address format. For example, something@somethingelse.xxx. For details, see *Administer Email Format in Order Management*.
Halt Configurator Validation on First Error	Set a value.

Parameter	Description
	• **True.** Stop processing on the first error that the configurator encounters during order entry. • **False.** Don't stop processing if the configurator encounters an error during order entry. Instead, allow the configurator to continue to run until it finishes processing. This setting allows the configurator to identify and report all errors that the configuration contains. Halt Configurator Validation on First Error affects only order entry. It doesn't affect order import.
Inventory Transaction Date for Order Lines	See *Reduce Inventory Without Picking or Shipping*.
Item Validation Organization	See *Use Item Validation Organizations with Order Management*.
Notify Frequency	Specify an integer that represents the number of hours to wait before consolidating, and then sending a notification. Notify Frequency comes predefined with a value of 1. You can set it to any value that's greater than or equal to 0 (zero). If you set it to 0, then Order Management won't consolidate notifications, and will send each notification when the event that it references happens. See *Send Notifications from Order Management to Other Systems*.
Number of Processes for Order Import	Set the maximum number of instances that Order Management can run of the *Load Interface File for Import* scheduled process at the same time when you import source orders. The default value is 4. Order Management divides the total number of orders that you import by the value that you set for the parameter, then uses the result to determine how many orders to process in each instance. Adjust this value as necessary depending on how it impacts performance. As a starting point, we recommend that you use the default value of 4. If you want to user a higher value, contact Oracle for assistance. Assume this parameter contains a value of 1, you change it to 10, you import 1,000 orders, and each order has the same number of order lines. Order Management will create 10 separate instances of the Load Interface File for Import scheduled process, assign 100 orders to each instance, and then process all 10 instances at the same time. In this example, you will improve performance by a factor of 10 when compared to using a value of 1 for this parameter. The balance across instances affects performance. Assume each order that's running on instance 1 has 1,000 order lines, and each order that's running on all the other instances have 1 order line. You won't realize the same improvement in performance because instance 1 will still take a long time to finish. See *Import Orders Into Order Management*.
Number of Times to Retry Pause	Specify the number of times to retry a pause task that pauses until time elapses. Adjust this value according to performance. See *Pause Orchestration Processes Until Time Elapses*.

Parameter	Description
Preparer for Procurement	Specify an Order Management user who the buyer can contact to help resolve a problem with a fulfillment line that involves a drop ship supplier. The buyer is a procurement application user. Order Management sends details to purchasing: • If you don't specify Preparer for Procurement, and if an error happens, then the procurement system will reply with an error that the preparer is missing. • You must specify at least one value so Order Management can successfully send a purchase request to Oracle Procurement. • The default value applies across all business units. • You can specify a separate preparer for each selling business unit. • You can specify a preparer as the default value for all business units. Order Management will use this default value only for business units where you don't specify a preparer. • Order Management doesn't use the value for any other attribute as the preparer. For example, it doesn't use the Created By attribute. It uses only the value that you specify in Preparer for Procurement. See *Set Up Drop Ship in Order Management*.
Process Inventory Transactions Immediately Process Inventory Transaction Lines as a Group	See *Reduce Inventory Without Picking or Shipping*.
Response Time in Seconds Response Time in Seconds for Copy Action	See *Response Time in Seconds* and *Response Time in Seconds for Copy Action*.
Return Control to Users After Requests to Save Sales Orders	See *Return Control to Users After Requests to Save Sales Orders*.
Send Discount Details to Billing Systems	Specify whether to send the list price and discounts, or only the net price, to your downstream billing system.
Specify Default Values for Tax Determinants	See *Specify Default Values for Tax Determinants*.
Start Approval Process for Sales Orders	Specify whether to get sales order approval. • If you must enable approval, then you must set this parameter to Yes even if you must enable approval only for Order Management. • If you set this parameter to No, then Order Management won't route sales orders for approval even if you create and activate an approval rule. • If you set this parameter to Yes but you don't create and activate an approval rule, then Order Management won't route the sales order for approval, but will instead send it to order fulfillment. For details, see *Set Up Approval Rules for Sales Orders*.

Parameter	Description
Use Configurator for Order Import Validation	Set a value. • **True.** The configurator will validate each source order that includes a configured item during order import. • **False.** The configurator won't validate.
Use Price on Imported Orders That Oracle Fusion Already Priced	Set a value. • **Yes.** Specify this value if your source system already used Oracle Pricing to price the source order. The import won't reprice it. If you set this parameter to Yes, then make sure your import payload includes the charge details. For details, see *Freeze Price on Sales Orders*. • **No.** The import will price the source order. This parameter applies only when you import through REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.
Use Pricing Algorithms to Calculate Totals for Sales Order	Set a value. • **Yes.** A pricing algorithm will calculate the total for each sales order that you import. Use this value only if the predefined logic doesn't meet your requirements. • **No.** Predefined logic will do the calculation. Order Management has tuned the predefined logic for performance.

Compare Change Order to Fulfillment Values

Use this parameter to help manage revisions that you import through more than one channel.

Set the parameter to:

- Yes when you revise through more than one channel. This setting will help to make sure the data in your sales order is consistent even when you import a revision through more than one channel. Assume you use FBDI to create and revise sales order 66684 from your order entry channel, and then use a web service to revise it from your shipping channel. The sales order data in your order entry channel might not match the data in your shipping channel. You can use this parameter to make sure you're using the current fulfillment data from Order Management.
- No when you revise through only one channel.

Assume you create sales order 66684, submit it, and then some time later you import a revision.

If you set this parameter to Yes:

1. You must use the GET action of the salesOrdersForOrderHub REST API to get all the attribute values for the sales order that's currently in fulfillment.
2. Make sure the import payload that you use for the revision includes a value for each attribute that you aren't revising. Use the values from step 1 to get these values.
3. Order Management will compare attribute values in your revision's import payload to attribute values in sales order 66684 that's currently in fulfillment to determine whether you revised any values.

4. If Order Management finds that you revised any attribute values, then it will replace the fulfillment line values in order 66684 with the revised values.

If you set this parameter to No:

- You don't need to populate values for attributes that you aren't revising in your payload.
- Order Management will use the order details that it received from the web service or file based data import when you initially created the order, and then compare them to the data that you send in the revision. Order Management will only update values that you revised.

Guidelines

If you set this parameter to Yes:

- And then deploy your set up, then you are committed to using it. Don't change it back to No after you deploy.
- You must use REST API to get the latest attribute values each time you revise the sales order.

For example, assume you:

1. Use file-based data import to create sales order 66684, the order has one order line, and you submit it to fulfillment.
2. Use the Order Management work area to revise sales order 66684. You add another order line to it, then submit it.
3. Use the Order Import web service to make other changes on sales order 66684. You must use the salesOrdersForOrderHub REST API to get the attribute values that you will use in your web service payload. You must modify only the attributes that you need to revise for both order lines.

Example

Assume you set this parameter to Yes.

1. You create sales order 37564 with two lines, don't set a value in the Warehouse attribute, then submit the order.

Order Line	Quantity	Warehouse
1	10	Empty
2	12	Empty

2. Order Promising schedules the sales order and the line moves to the Awaiting Shipping status.

Order Line	Quantity	Warehouse	Status
1	10	M1	Awaiting Shipping
2	12	M1	Awaiting Shipping

3. You revise the order, change only the quantity, and then submit the revision.

Order Line	Quantity	Warehouse
1	5	Empty
2	7	Empty

Here's what happens next.

- Order Management compares the revision's attribute values in step 3 to the original order's values in step 2 to determine whether you changed any attribute values.
- The warehouse on the original order in step 2 is M1 but it's empty on the revision in step 3, so Order Management processes the change for the quantity and for the warehouse, then submits it to fulfillment.

Assume you modified the warehouse by mistake and need to set it back to its original value of M1. Here's what you need to do.

- Use the GET action of the salesOrdersForOrderHub REST API to get the latest values for all of the original order's attributes. It will have a value of M1 in the Warehouse attribute.
- Copy the attribute values from the REST response into the import payload that you're using to revise the order.
- Modify only the attributes that you need to change in your payload, then redo the import. For example:

Order Line	Quantity	Warehouse
1	5	**M1**
2	7	**M1**

Now let's assume you set this parameter to No.

- You import a source order, your import doesn't include a value for the Scheduled Ship Date attribute, and you submit the order to fulfillment.
- Order Management schedules the order line and processes it to the Awaiting Shipping status.
- Some time later, you import a revision that changes the quantity on the order line, and then submit the revision. You don't need to provide a value for the Scheduled Ship Date in your import payload even if a value already exists for this attribute in the original order.

Response Time in Seconds

Use the Response Time in Seconds parameter to specify the number of seconds to wait before giving control back to the user after the user submits a sales order that has a large number of order lines.

Improve your user experience. Sales orders that have a large number of order lines might take a few minutes to process. You can specify the number of seconds to wait before giving control back to the user for sales orders that have a lot of order lines.

- You can specify a value of 5 seconds to 240 seconds.
- If you set a value, then Order Management returns control according to the time it takes to finish processing the submit, not according to the number of order lines.
- If you don't set a value, then Order Management gives control back to the user only after it finishes submitting all the order lines in the order.
- If Order Management finishes submitting the order to fulfillment before the time that you specify elapses, then it displays the same success message or error message that it displays when you don't use Response Time in Seconds.
- If Order Management doesn't finish submitting the order to fulfillment before the time that you specify elapses, then the Order Management work area displays a dialog that you can use to navigate away from the sales order while Order Management continues to process the submit. If you're still on the order page, then you can refresh the page to get an updated order status. If you navigate away from the order page, then you can search for the sales order and get a status update. If you encounter an error or warning, you can edit the order or discard the draft just like you can if you don't set a value in the parameter.
- This feature applies only for sales orders that you submit in the Order Management work area. It doesn't apply for source orders that you import.
- Response Time in Seconds doesn't come predefined with a value.
- This feature works only when you click Submit in the Order Management work area to create, revise, or cancel a sales order. It doesn't apply when you import an order.
- This feature applies to all users and all business units.

Response Time in Seconds for Copy Action

Improve your user experience. It might take Order Management a few minutes to copy a sales order that has a large number of order lines. Use this parameter to specify the number of seconds to wait before allowing the user to navigate away from the order while Order Management creates the copy.

- This feature works only when you click Copy in the Order Management work area. It doesn't work during order import.
- It applies to all users and all business units.
- This parameter doesn't come predefined with a value. You can specify a value of 5 seconds to 240 seconds.

Set a Value

If you set a value, then you can navigate away from the original sales orders while Order Management creates a copy of it.

If Order Management finishes copying all lines in the order before the time that you specify elapses, then the Order Management work area displays the Edit Order page for the new copy.

If Order Management doesn't finish copying all lines in the order before the time that you specify elapses, then:

- The Order Management work area displays a dialog that has the order number for the new copy.
- You can close the dialog and remain on the Order page for the original sales order.
- The View Order page displays a lock icon next to the new order number while Order Management creates the copy. You can refresh the page to get the latest details.

- Order Management doesn't lock the original order, but you must not modify it while Order Management is creating the copy.
- You can use the order number from the dialog to search for the new copy. If the search isn't successful after some time, then you can go to the Manage Order Orchestration Messages page to search for it and examine the results.
- You can work on other sales orders while you're waiting.

Don't Set a Value

If you don't set a value, then you can't navigate away from the original sales order until Order Management finishes copying all lines in the original order. Order Management finishes copying the order, then displays the new copy on the Edit Order page.

Return Control to Users After Requests to Save Sales Orders

Improve your user experience. Give your user control to navigate away from a sales order while Order Management saves the sales order instead of the user having to wait for the save to finish.

A sales order that has a large number of order lines might take a few minutes to save when you click Save or click Save and Close. You can use this parameter to specify whether to give your user control to navigate away from the sales order while Order Management saves the order. If you don't set the parameter, then Order Management gives control to the user only after it saves all lines in the order.

Set the parameter to a value.

- **No**. You can't navigate away from the sales order until Order Management finishes saving all lines in the order. You remain on the Edit Order page after Order Management finishes saving the order. The predefined value is No.
- **Yes**. You can navigate away from the sales order while Order Management saves it.
    - You submit or save the sales order, the Order Management work area displays a dialog, then displays the View Order page.
    - If Order Management is still saving the order, then the View Order page displays a lock icon next to the order number. You can refresh the page to get the latest details.
    - If you navigate away from the View Order page, then you can search for the sales order to get the latest details.
    - If an error or warning happens, then you can click the error or warning icon to view the detail.

This feature works only when the user clicks Save or Save and Close in the Order Management work area. It doesn't apply when you import an order.

Specify Default Values for Tax Determinants

Use the Specify Default Values for Tax Determinants parameter to specify when to apply default values on attributes that affect tax. This parameter gives you a lot flexibility for the values that you use to calculate tax.

- Improve performance when you have really big sales orders and you don't want to calculate tax for them.
- Use this parameter with sales orders that you create in the Order Management work area or with orders that you import. You can also use an order management extension or transformation rule to automatically set default values.
- Use default values when you create the sales order or submit it in the Order Management work area, or only use values that you import.

- You can apply default values when you save the sales order, submit the order, or when you modify an order header attribute or order line attribute that affects tax.
- You can specify to not apply default tax values at all. For example, when you already applied tax on a sales order that you import.
- You can turn off the default behavior and instead set default values when you save or submit the sales order. You can also have Order Management set default values when you update attributes on the order header or the order.

Here are the values that you can set for the Specify Default Values for Tax Determinants parameter.

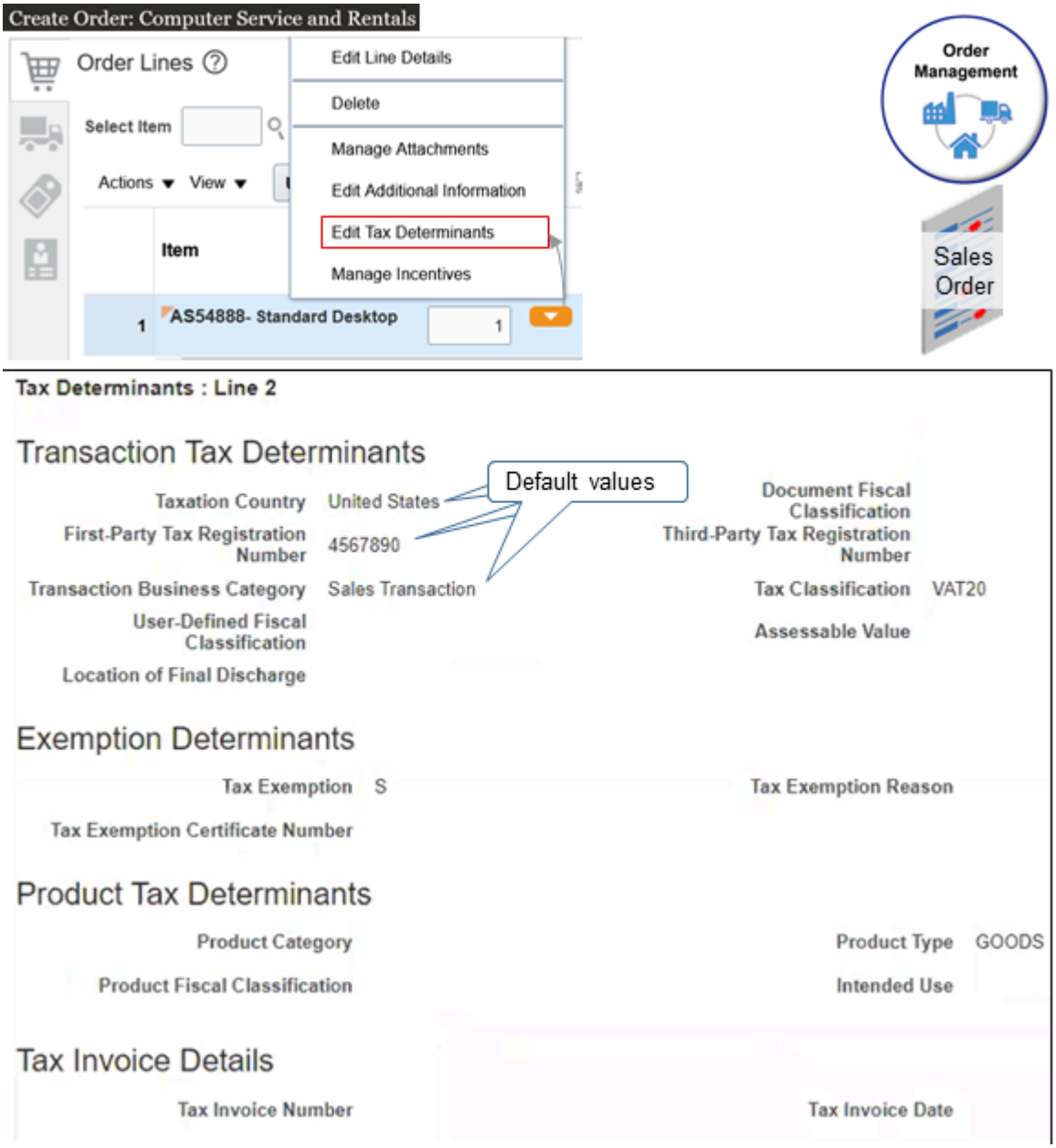
Value	Description
Set Defaults When You Modify Attributes and Order Management	Set default values when you modify an attribute that affects tax on the order header or order line.
Set Default Values When Saving the Order	Set default values only when you save the sales order.
Set Default Values When Submitting the Sales Order	Set default values only when you submit the sales order. Order Management won't set default tax values while the order is in draft status. Order Management uses this behavior for orders that you create in the Order Management work area and for orders that you import.
Do Not Set Default Values	Don't set the default value for any tax determinant. Use this setting when you want to calculate tax completely outside of Oracle Applications or when you import tax values into Order Management and you want to use those imported values. You might also consider using this setting if you have really big sales orders and you need to maintain a high level of performance. Calculating tax on very large orders might impact performance.
On Demand	Don't use. Its for a future update.

For more details about tax, see:

- The *Tax* section
- *Edit Tax on Order Lines*
- *Troubleshoot Tax Set Up*

Order Management Extensions

If you use an order management extension to automatically create order lines when you import sales orders through Oracle Application Development Framework, then the extension will set the default values even if you set the parameter to Set Defaults When You Modify Attributes. Here's an example that sets some default values, such as setting the default value for the Taxation Country attribute to United States:



Order management extensions didn't automatically set these values in update before Update 22D.

Order Management sets default tax values after the On Save event. For example, it sets the Product Fiscal Classification after it saves the sales order, not before the save. So, if you're currently using an extension with the On Save event to validate a tax determinant, then we recommend that you modify your extension so it uses the On Start of Submission Request event, which happens after On Save. This way, your extension will use the defaulted value.

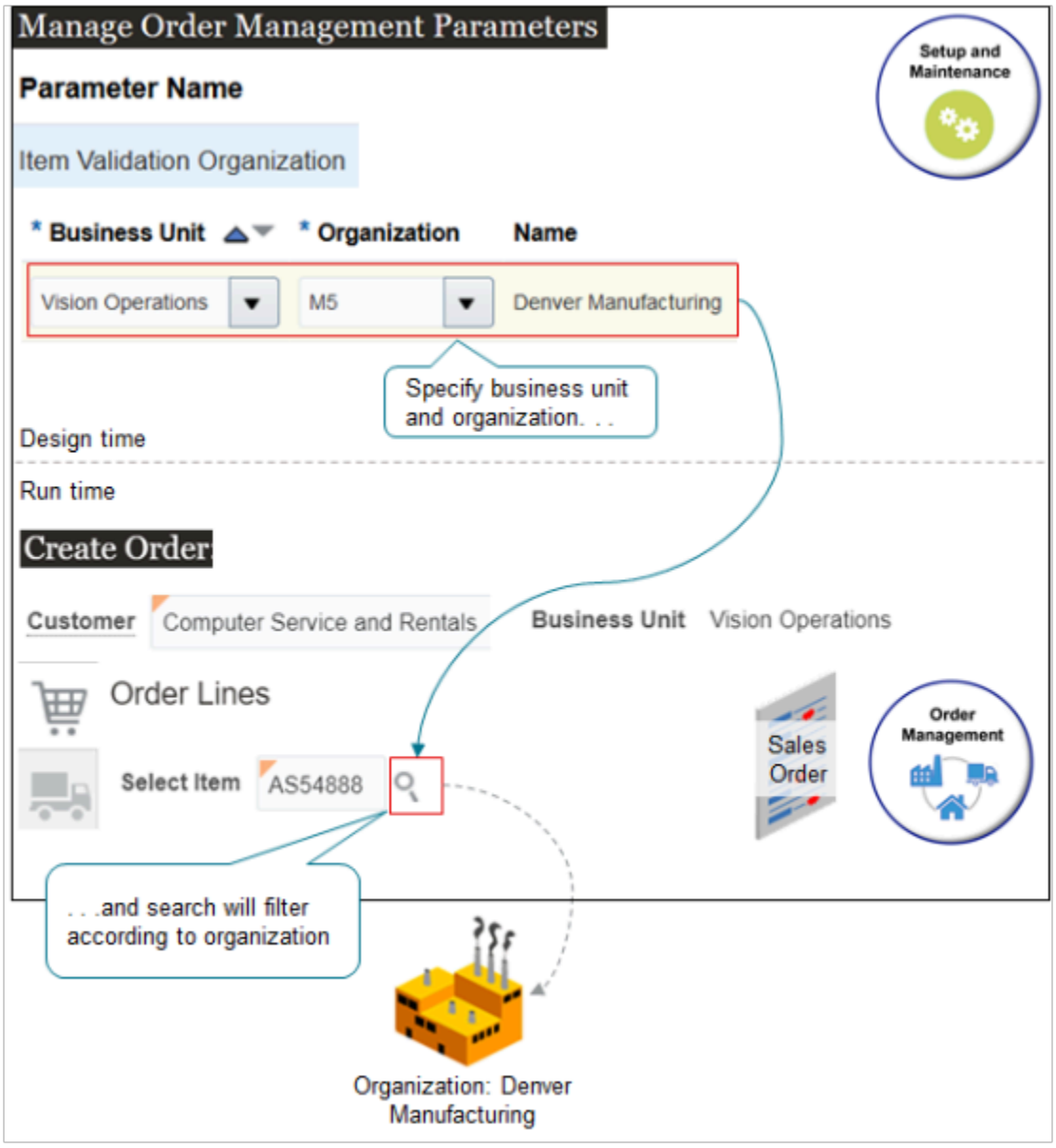
For details, see *Overview of Creating Order Management Extensions*.

Use Item Validation Organizations with Order Management

Use the Item Validation Organization parameter to specify the organization that validates whether fulfillment for the item is correct, and to make sure it meets your users' and customers' needs.

For example, you can specify the organization that Order Management uses to filter the items that you search for on a sales order, according to your business unit. Order Management will display only the items that it associates with the organization that you specify.

Assume your Vision Operations business unit only sells items that the Denver Manufacturing organization builds, so you must filter search so it only returns items from Denver Manufacturing.



Try it.

1. Go to the Manage Order Management Parameters page and click the row that contains the value.

Attribute	Value
Parameter Name	Item Validation Organization

Attribute	Value

2. In the Values area, click **Add Row**, then set the values.

Attribute	Value
Business Unit	Vision Operations
Organization	Denver Manufacturing The Organization attribute only displays values that Product Information Management associates with the business unit that you select in the Business Unit attribute. You can use Product Information Management to set up an inventory organization and create these associations.

3. Go to the Order Management work area, create a sales order, and set the Business Unit to Vision Operations.
4. Search for an item on the catalog line on the sales order.
5. Verify that the search returns only the items that Denver Manufacturing builds.

Note

- You must specify an organization for each business unit. If you don't, then at run time, Order Management disables search for the item in each sales order that references a business unit that you don't specify.

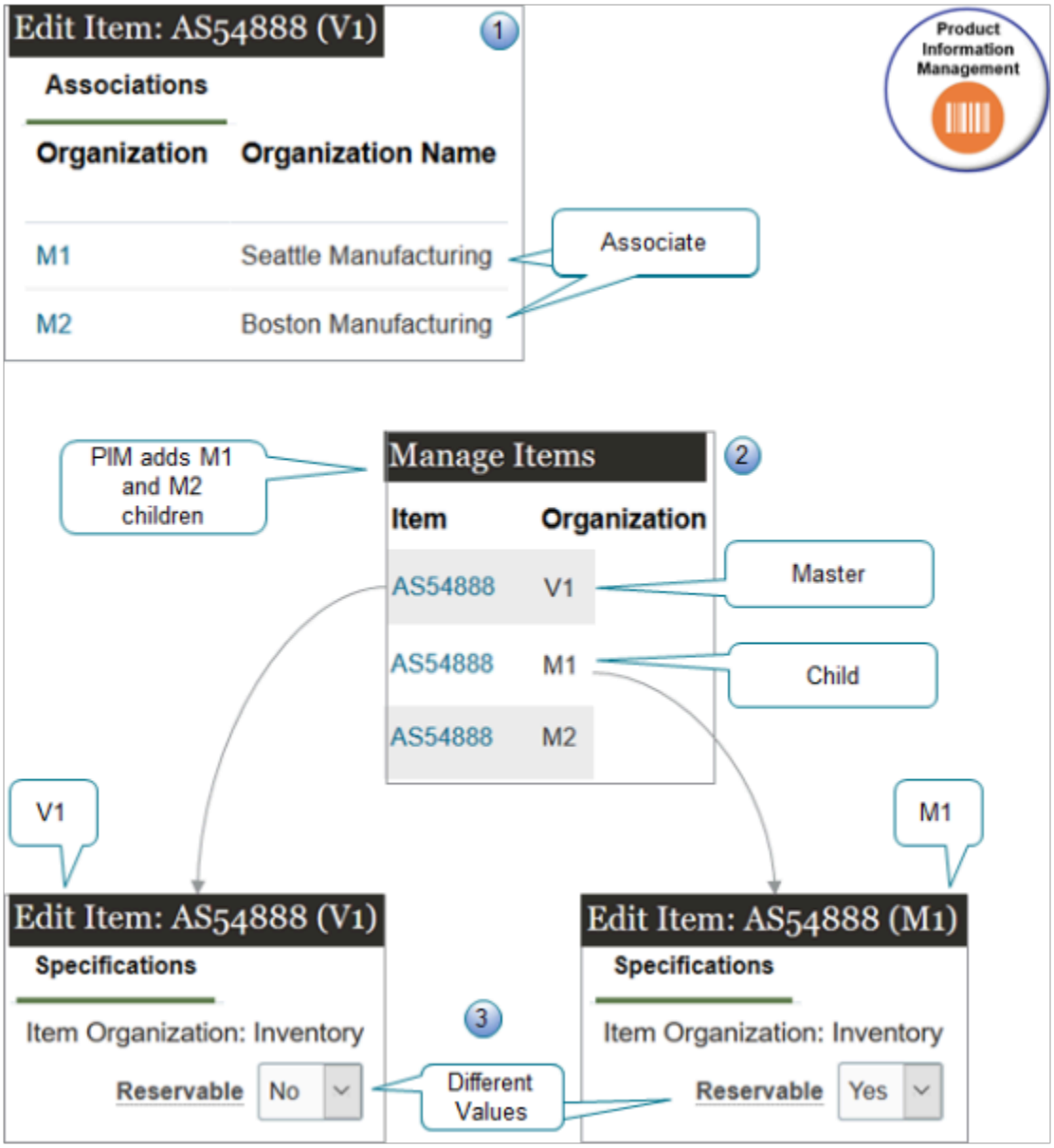
  As an alternative, if you use the same master organization for every business unit, then you can specify these values.

Attribute	Description
Business Unit	Set it to All Business Units.
Organization	Specify the master organization.

- You can set the Item Validation Organization parameter only for business units that Order Management associates with your sign in responsibility. If it associates your sign in responsibility with only one business unit, then it uses this business unit as the item validation organization.
- If you change the value of the Item Validation Organization parameter, then the change that you make doesn't affect the sales orders that your orchestration process is currently processing. Orchestration will apply your change only to sales orders that you create after you make the change.

There's a Master Organization, and it Has Children

You can set attributes for an item in Product Information Management at the master level and the child level.



Note

1. You create the master first and set its attributes.

   You then click Associations to add an inventory organization as a child of the master. This way, the downstream flow can create a work definition during manufacturing.
2. Product Information Management automatically creates a duplicate of the master when you add the child, but with a few important differences, such as the organization.

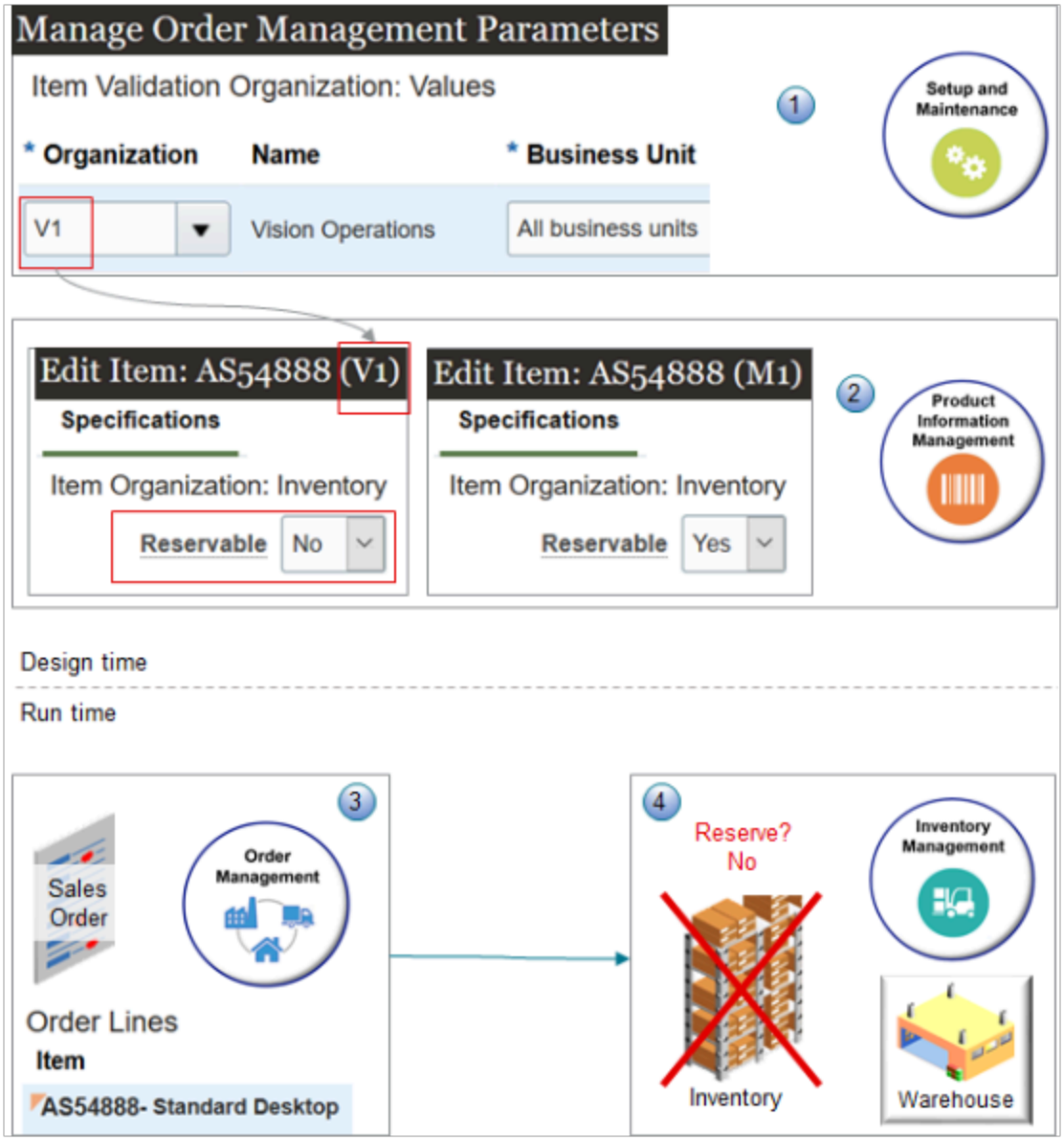
   - Assume you add M1 to V1. To see them, go to the Manage Items page and query for your item. You will see V1 and M1 on separate rows, but with different values in the Organization column.
   - V1 is the master. V1 means Vision Operations, which is headquarters for Vision Corporation. They keep all the masters.
   - M1 and M2 are children. M1 means Seattle Manufacturing, and M2 means Boston Manufacturing.

3. You can edit V1 and M1 separately. M1 and M2 inherit most of their attribute values from V1, but you might need to set attributes for M1 and M2 differently from what you set for the master to meet the unique needs that each manufacturing plant has. For example, you can set the Reservable attribute to No on the master and then set it to Yes on the child.

Specify the Item Validation Organization

Order Management uses attribute values that you set for the item in Product Information Management during fulfillment depending on how you set the Item Validation Organization parameter.

Consider an example.



Note

1. You set the Item Validation Organization parameter to V1 for all business units.
2. You set the Reservable attribute in Product Information Management to No for the V1 master, and you set it to Yes for the M1 child.

3. At run time, you create a sales order for an item that's in the V1 organization. Order Management will use the values that you specify for V1 in Product Information Management. It doesn't use values from the child, and because you usually specify the inventory organization as the child, that means it won't use values from the inventory organization when it fulfills your sales order.
4. The orchestration process won't reserve inventory for the item during fulfillment because it uses values from the master, not the child.

The same behavior applies for other attributes. For example:

- If you set the Invoice Enabled attribute to Yes for the master, then Accounts Receivable will create an invoice for the item even if you set Invoice Enabled to No for the children.
- If you set the Default Shipping Organization to Seattle Manufacturing on the master, but then set the Supply Warehouse on the order line to Boston Manufacturing, Order Management will fulfill the item from Seattle, not Boston.
- If you set the Output Tax Classification Code on the child but not on the master, then Order Management sets the Tax Classification Code differently depending on when you add the order line.
    - . Order Management sets the Tax Classification Code on the order line according to the value that you set for the Output Tax Classification Code on the master.
    - . Order Management sets the Tax Classification Code on the order line according to the value that you set for the Output Tax Classification Code on the child.

Specify Master Organizations and Inventory Organizations

You can use the Product Information Management work area to associate an inventory organization with a business unit.

If you don't set the Item Validation Organization parameter, and if Product Information Management associates more than one inventory organization with the same business unit, and.

If Product Information Management Does This	Then Order Management Does This
Associates the same master organization with these inventory organizations.	Sets Item Validation Organization to the master, then informs the Order Entry Specialist of this setting. The Order Entry Specialist can accept this value and continue entering the sales order, or reject it and contact the Order Administrator with a request to use the Item Validation Organization parameter to specify the inventory organization.
Associates different master organizations with these inventory organizations.	Informs the Order Entry Specialist that the Order Administrator must first set up the inventory organization before the Order Entry Specialist can create the sales order.

For details, see *Inventory Organizations*.

Make Sure Your Item is Part of a Customer Order

If you create a sales order in the Order Management work area and submit it, or if you import it, then Order Management validates that the Customer Ordered attribute and the Customer Orders Enabled attribute contain Yes for the item in Product Information Management. It does this for each item on each order line in your sales order.

- If either attribute contains No, then Order Management doesn't submit the sales order and instead displays an error message.

- If you use the catalog line on the Create Order page or the Revise Order page, then Order Management only displays items that pass validation when you search for the item.
- If you use the Order Management work area to add an item to an order line, save the order in Draft status and the item passes validation, but you then set the Customer Ordered attribute or the Customer Orders Enabled attribute to No, then you can't submit the order.
- If you import an order line, and if the line fails the validation, then Order Management imports the order but sets the order status to Draft status and doesn't submit it to order fulfillment.

Here's how to make sure your item will pass validation.

1. Look at the Item Validation Organization parameter and identify the organization that you use in your business unit. For details, see *Manage Order Management Parameters*.
2. Go to the Product Information Management work area, click **Tasks > Manage Items**, then search for and open your item in the organization that you identified in step 1.
3. Click **Sales and Order Management**, then set the Customer Ordered attribute and the Customer Orders Enabled attribute to Yes.

Order Management doesn't do this validation for lines:

- It already shipped
- It already sent to Account Receivables
- It already cancelled or closed
- It adds through a product transformation rule
- That are part of a return
- That are a child of a configured item
- That are part of an internal material transfer

Related Topics

- Manage Order Management Parameters
- What You Can Do in Product Master Data Management
- Inventory Organizations

Time Zone Differences in Order Management

Oracle Global Order Promising uses the time zone where the inventory organization resides when it calculates promising for the supply that Oracle Order Management needs to fulfill each sales order, then sends the scheduled ship date and scheduled arrival date to Order Management.

Global Order Promising uses date attributes from Order Management to optimize supply, such as Earliest Acceptable Date or Latest Acceptable Date.

Using the time zone where the inventory organization resides provides more accurate values in date attributes, and helps to avoid undesirable time shifts between demand and supply. For example, to prevent the scheduled ship date from occurring before the requested date or the ordered date.

- Order Management applies this behavior for sales orders that you create in the Order Management work area and for source orders that you import order from a source system.
- This behavior doesn't apply for drop shipments.

Example of Calculating Time Zone Differences

Assume:

- You're in Denver, Colorado, USA, in Mountain Standard Time (MST), which is UTC-7.
- The inventory organization also uses UTC-7.
- The warehouse uses a scheduling calendar that's open Monday through Friday, so Global Order Promising can set the Scheduled Ship Date attribute for the shipment on any weekday, but not on the weekend.
- The on-hand supply is always available.

For example:

1. You use the Order Management work area to set the requested date on the sales order to 3/2/24 5:00 PM, which is a Friday, then submit the sales order.
2. Order Management interacts with Global Order Promising in UTC (Coordinated Universal Time), so it sends 3/3/24 00:00AM (UTC) as the requested date.
3. Promising converts the requested date to the time zone where the inventory organization resides, which is 3/2/24, Friday, 05:00 PM (UTC-7).
4. Promising schedules the shipment at the inventory organization to the end of the day, which is 3/2/24, 11:59 PM local time (UTC-7), because this day is a working day.
5. Promising sends schedule details to Order Management as Saturday, 3/3/24, 6:59 AM (UTC).

   06:59 is a seven hour offset from UTC. The calculation is `Friday, 11:59 PM local time (UTC-7) plus seven hours equals Saturday, 3/3/24 06:59 AM (UTC).`
6. The Order Management work area now displays the scheduled ship date in UPTZ, which is UTC-7.

   The scheduled ship date in UTC was 3/3/18 minus 06:59AM (UTC). The time and date in UPTZ is 3/2/24 minus 11:59 PM (UTC-7).

   Promising uses UTC. Order Management displays dates in UPTZ (User Preference Time Zone) so they're meaningful to the person who uses the Order Management work area.

UTC (Coordinated Universal Time) uses this nomenclature:

`UTC-x`

where

- - means minus
- ***x*** equals hours

For example, `UTC-7` means `UTC minus 7 hours.`

Handle Time Zone Differences When You Import Orders

You can use the DooDecompReceiveOrderComposite web service to import a source order from your source system. You use various date attributes in the web service payload, such as requested ship date, requested arrival date, expected arrival date, and so on.

You use the `yyyy-mm-ddTHH:mi:ssZ` format for each date attribute.

Requirements for how you format dates are different depending on how you import your source order.

How You Import	Description
You use the ReceiveOrder web service. For details, see *Use Web Services to Import Orders*.	Dates in your import payload can use UTC with or without a negative or positive offset.
You use the order import template. For details, see *Import Orders Into Order Management*. You use the business-to-business flow. For details, see *Overview of Business-to-Business Messaging in Order Management*.	Dates in your import payload must use UTC.

Here's an example payload for the ReceiveOrder web service. It includes a UTC offset for the Earliest Acceptable Ship Date attribute. This value indicates that the end of day is April 3, 2018, and the time is 1:10:10 AM in the time zone that's 8 hours behind UTC.

```
<ns2:SupplierAddressLine3/>
<ns2:SupplierAddressLine4/>
<ns2:SupplierAddressCity/>
<ns2:SupplierAddressState/>
<ns2:SupplierAddressZipCode/>
<ns2:SupplierAddressProvince/>
<ns2:SupplierAddressCountry/>
<ns2:FulfillmentMethodCode/>
<ns2:Comments/>
<ns2:ReferenceTransactionLineId/>
<ns2:InterfaceStatus/>
<ns2:UnitListPrice>5</ns2:UnitListPrice>
<ns2:UnitSellingPrice>5</ns2:UnitSellingPrice>
<ns2:ExtendedAmount>5</ns2:ExtendedAmount>
<ns2:BatchIdentifier/>
<ns2:DestinationShippingOrganizationIdentifier/>
<ns2:DestinationShippingLocationIdentifier/>
<ns2:EarliestAcceptableShipDate>2018-04-03T01:10:10.0-08:00</ns2:EarliestAcceptableShipDate>
<ns2:LatestAcceptableShipDate>2018-04-05T01:10:10Z</ns2:LatestAcceptableShipDate>
<ns2:EarliestAcceptableArrivalDate>2018-04-06T01:11:10Z</ns2:EarliestAcceptableArrivalDate>
<ns2:LatestAcceptableArrivalDate>2018-04-08T01:11:10Z</ns2:LatestAcceptableArrivalDate>
<ns2:PromiseShipDate/>
<ns2:PromiseArrivalDate/>
<ns2:SubInventoryCode/>
<ns2:SubInventory/>
<ns2:ShipSetName/>
<ns2:TaxExempt>S</ns2:TaxExempt>
<ns2:TaxClassificationCode/>
<ns2:TaxExemptionCertificateNumber/>
<ns2:TaxExemptReasonCode/>
<ns2:DefaultTaxationCountry/>
<ns2:FirstPartyTaxRegistration/>
<ns2:ThirdPartyTaxRegistration/>
<ns2:DocumentSubType/>
<ns2:FinalDischargeLocationIdentifier/>
<ns2:ProductFiscalCategoryIdentifier/>
```

You must the `yyyy-mm-ddThh:mi:ssz` format.

where

This Variable	Specifies This Value
yyyy	Calendar year.
mm	Month of the calendar year.

This Variable	Specifies This Value
dd	Numeric day of the month.
T	Time designator.
hh	Number of hours after midnight, in the 24 hour format.
mi	Minutes after the beginning of the hour.
ss	Seconds after the beginning of the minute.
z	Time zone indicator with or without a UTC offset. For example, 0-8.00 is eight hours behind UTC.

Consider an example.

```
2024-04-03T01:10:10.0-08:00
```

where

Value	Description
2024	Calendar year.
04	Month of the calendar year, which is April.
03	Numeric day of the month, which is April 3rd.
t	Time designator.
01	One hour after midnight.
10	Ten minutes after the beginning of the hour.
10	Ten seconds after the beginning of the minute.
0-08:00	where • **0**. Specifies 0 UTC. • **-**. Delimiter between 0 UTC and the UTC offset. • **08:00**. Specifies 8 hours after 0 UTC.

Guidelines for Setting Up Units of Measure

Use guidelines to help you set up the units of measure that you use in Order Management.

- Set the Primary Unit of Measure attribute and the Secondary Unit of Measure attribute when you create the item in the Product Information Management work area. Order Management uses the values that you specify to set the default value for each of these attributes on the sales order.
- Make sure the unit of measure is appropriate for the item. For example, Quart is appropriate for a liquid, but Amperage isn't because amperage measures electrical current.

Import

If you import a source order through FBDI or a web service:

- Collect units of measure from your order capture system so Order Management can validate the units of measure it receives later when you import source orders from your capture system. For details, see the Collect Data subtopic in *Quick Start for Setting Up Order-to-Cash*.
- Use the Ordered UOM attribute to specify the unit of measure. Use the OrderedUOMCode attribute to specify the abbreviation for the measure. For example, Ea is an abbreviation for Each.
- The import uses the value that you import instead of the value that you specify in Product Information Management.
- If you use the Pricing Administration work area to set up pricing for your item, then the import validates the unit of measure according to how you set up the price list. For example, if you set the Pricing UOM attribute for the item on the price list to Ea, and if your import doesn't use Ea for the unit of measure, then the import fails and displays an error message.
- If you encounter an error during order import, like `Cross-Referenced Value Not Found for UOM_CODE`, see the Cross-Reference Error subtopic in *Troubleshoot Problems With Order Import*.

Fulfill

- If you don't encounter an error during import but the order gets stuck during fulfillment, then consider creating a conversion rule.
- If your shipping system uses a unit of measure to represent shipping that's different from the unit of measure that Order Management uses in the sales order, then the shipment service converts the unit of measure back to the unit of measure that the sales order uses, then communicates the shipped quantity to Order Management. For details, see *Task Services*.
- If you use the Create Inventory Reservations fulfillment task to reserve supply, then you must provide the item, quantity, unit of measure, and warehouse. For details, see *Guidelines for Reserving Inventory*.
- If you use Oracle Inventory Management, then create a conversion rule that converts the UOM from the sales order into a UOM that Oracle Inventory Management can understand. If you use your own fulfillment system, then make sure it accepts the UOM that you set on the sales order.

Create an Interclass Conversion

Order Management doesn't limit what UOMs your fulfillment system can use according to UOM class. If your fulfillment system uses a specific UOM class, then you must set up a UOM interclass conversion for each UOM that's eligible on the order line according to your pricing setup.

Consider an example:

- Assume you create an All Items price list. You specify the Pricing UOM attribute on the price list so Oracle Pricing prices the item according to the item's UOM and the order line's type. This setup allows your users to select any UOM that you have set up for the item on the order line at run time.
- Your fulfillment system uses the Count UOM class.
- You create a sales order, add a return line to the order, set the UOM on the line to Kilogram, then submit the order.
- Order Management sends the return line to your fulfillment system to process the return request.
- Kilogram is in the Weight UOM class but you haven't set up an interclass conversion between the Count UOM class and the Weight UOM class. Your fulfillment system can't process the request and displays an error.

  ```
  The value provided for the Unit of Measure attribute is not valid for this transaction.
  ```

To fix this problem, you can use the Manage Units of Measure for Interclass Conversion task to create a UOM interclass conversion between Count and Weight. For an example that uses this task, see *Set Up Dual Units of Measure*.

For more, see *Set Up Units of Measure for Pricing*.

Related Topics

- Quick Start for Setting Up Order-to-Cash
- Troubleshoot Problems With Order Import
- Task Services
- Guidelines for Reserving Inventory
- How Units of Measure, Unit of Measure Classes, and Base Units of Measure Relate to Each Other

Display Data from Different Offerings

Display data from different offerings on the Overview page in the Order Management work area.

The Overview page displays data in infolets. An infolet is a graphic representation of data, such as a bar chart or pie chart. It displays order status, such as the number of orders on backorder, orders past due, orders on hold, orders in jeopardy, and so on.

The Overview page comes predefined to only display data from the Order Management offering. However, you can set it up so it displays data from other offerings.

1. In the Setup and Maintenance work area, go to the Order Management offering.
2. Click **Actions > Edit Implementation Status**, set status to Implemented, then click **Done**.
3. Repeat steps 1 and 2 for each offering that must display data in infolets.

   For example:

   - Order Promising is part of Supply Chain Planning. To display promising data, in step 1, go to Supply Chain Planning.
   - Shipping is part of Manufacturing and Supply Chain Materials Management. To display ship data, in step 1, go to Manufacturing and Supply Chain Materials Management.

Your users must sign in with the FOM_VIEW_ORDER_TO_CASH_INFOLET_PAGE privilege to view infolets.

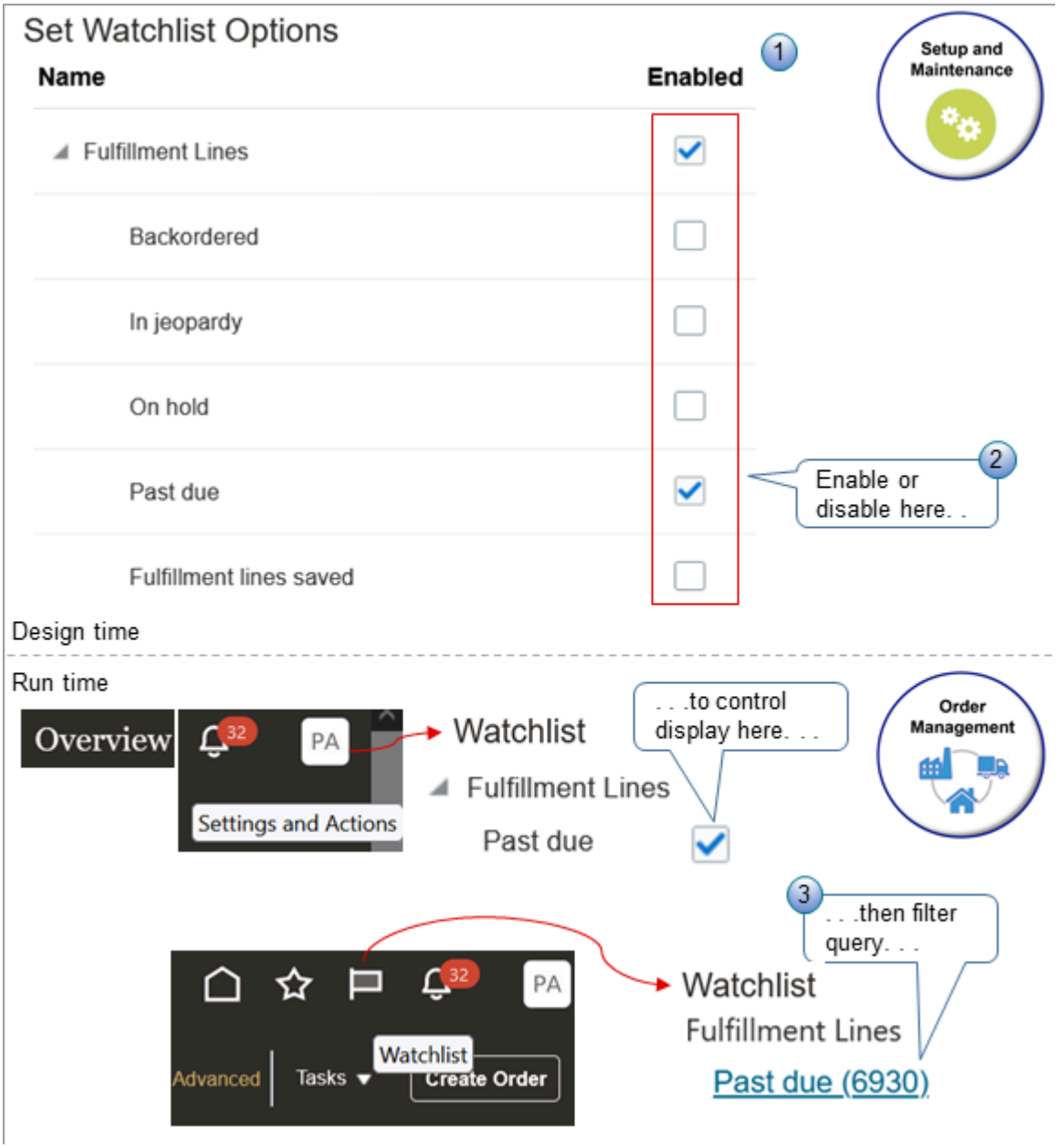
Here are more privileges you need to view infolets from each offering.

Infolet	Offering	Privilege
Order Exceptions Orders in Process Order Billing Status	Order Management	FOM_VIEW_ORDERS_PRIV
Scheduling Performance	Supply Chain Planning	MSP_VIEW_PLANNING_SUPPLY_AVAILABILITY_PRIV
Inventory Valuation	Manufacturing and Supply Chain Materials Management	CST_RUN_INVENTORY_VALUATION_REPORT_PRIV
Open Shipments Open Shipments by Priority Shipment Exceptions	Manufacturing and Supply Chain Materials Management	WSH_MANAGE_DELIVERY_PRIV

Manage Your Watchlists

To improve performance, you can control the options that your users can select for the watchlist.

Control the options that your users can select for the watchlist.



Note

1. Order Management comes predefined to allow your users to display all fulfillment lines that are backordered, in jeopardy, on hold, and past due in the watchlist. It also comes predefined to display lines that are part of a search that your user creates. This might affect performance if you process a lot of sales order.
2. You can use the Setup and Maintenance work area to remove some or all of this data from the watchlist for all of your users.
3. At runtime, Order Management will filter the data that it queries according to the watchlist options that your user enables.

Try It

Assume you want to allow your users to only display fulfillment lines that are past due in the watchlist.

1. Go to the Setup and Maintenance work area.
2. Click **Tasks > Search**.
3. Search for, then open the Set Watchlist Options task.
4. On the Set Watchlist Options page, expand **Fulfillment Lines**.
5. Disable these options.
    - Backordered
    - In Jeopardy
    - On Hold
    - Fulfillment Lines Saved
6. Make sure these options are enabled.
    - Fulfillment Lines
    - Past Due
7. Click **Save and Close > Done**.
8. Test work your.
    - Go to the Order Management work area.
    - On the Overview page, in the banner, click **Settings and Actions**.
    - In the Settings and Actions dialog, click **Set Preferences**.
    - On the Preferences page, click **Watchlist**.
    - On the Watchlist page, expand **Fulfillment Lines**, then verify that you can enable only the Past Due option.
    - In the banner, click **Watchlist** (the flag icon).
    - Verify that the Fulfillment Lines area of the Watchlist dialog displays only the Past Due link.

Other Settings

Here are some more options that affect Order Management. You can control them on the Set Watchlist Options page in Setup and Maintenance:

- Change Orders
- Orchestration Processes
- Orders
- Sales Orders

You can also use the Refresh Interval in Sections attribute on each option to control how often Order Management runs the query for each option. Refresh less often to improve performance.

Collect Data

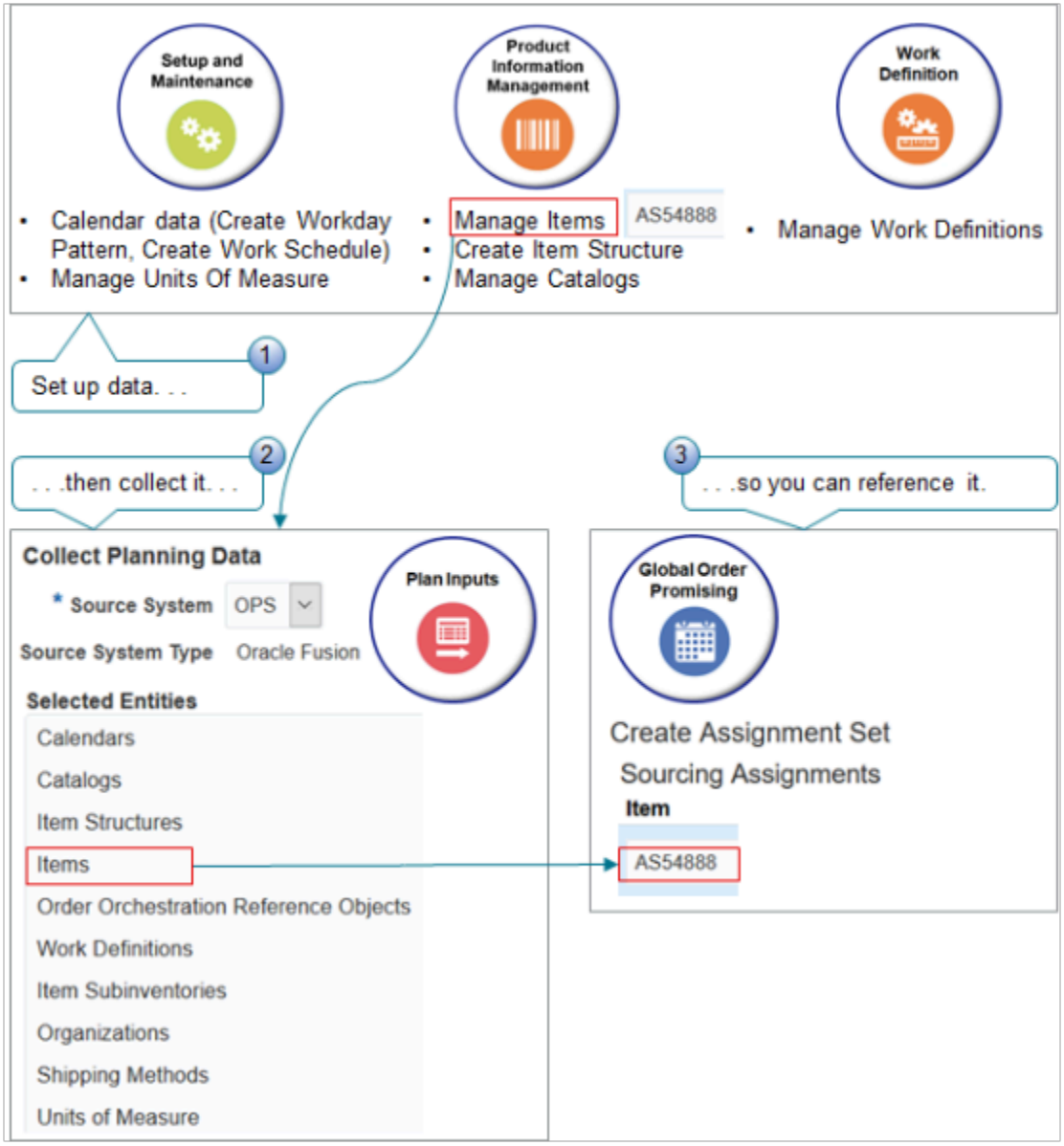
Overview of Collecting Promising Data for Order Management

Collect data for order promising at various points of your set up process, and also after you finish set up.

Global Order Promising promises orders that it receives from Order Management. It uses data from your supply chain network and supplies that you collect. You set up rules in Promising that specify how to plan and promise.

You must collect data into the Global Order Promising repository. The repository stores details about the setups that you make in the Global Order Promising work area, such as rules for sourcing, available-to-promise, and allocating supply. Promising and sourcing work together to determine what to deliver to your customer within the time frame that the sales order requests. Collecting also makes sure that you can search for and select data in the Order Management work area, such as adding an item to an order line, or selecting a value in an attribute, such as the Warehouse.

For example, collect the catalog that you assign to an item in the Product Information Management work area so you can use it in an available-to-promise rule that you create in the Global Order Promising work area.



Data You Must Collect

You must collect data several times during setup. For example, collect each time you.

- Create or modify an item in the Product Information Management work area.
- Modify the item, item structure, catalog, or work definition.

Use the Collect Planning Data task to collect data each time that you do the work described in the Source column. You must collect the entity listed in the Entity column.

Source	Entity
Manage Suppliers task in the Suppliers work area.	Approved Supplier List
Create Workday Pattern task and Create Work Schedule task in the Setup and Maintenance work area.	Calendars You must collect calendars from your fulfillment system so Order Management can use them during scheduling. A calendar specifies when a facility, such as a warehouse, is open or closed.
Manage Catalogs task in the Product Information Management work area.	Catalogs
Manage Currencies task in the Setup and Maintenance work area.	Currencies
Manage Customers task in the Setup and Maintenance work area.	Customer
-	Demand Class
Manage Geographies task in the Setup and Maintenance work area.	Geographies
Manage Items task in the Product Information Management work area.	Items
Create Item Structure task in the Product Information Management work area.	Item Structures
Manage Subinventories task in the Product Information Management work area.	Item Subinventories
Manage Item Relationships task in the Product Information Management work area.	Item Substitution Relationships
This entity includes. • Freight terms	Order Orchestration Reference Objects

Source	Entity
• FOB points • Invoicing rules • Accounting rules • Shipment priorities • Payment terms • Return reason • Tax classification code • Tax exemption reason • Sales credit type • Activity type • Document categories • Payment methods • Receipt methods	
Item Organization area of the Manage Items task in the Product Information Management work area.	Organizations
Manage Resources task in the Work Definition work area.	Resources
Manage Carriers task in the Setup and Maintenance work area.	Shipping Methods If Order Promising must consider the transit time that happens during shipping between the warehouse and a destination, then you must collect shipping methods. If you don't, then Order Management uses a transit time of zero days.
-	Subinventories
Manage Suppliers task in the Suppliers work area.	Suppliers
Manage Units of Measure task in the Setup and Maintenance work area.	Units of Measure Order Management validates the units measures and currencies against data that it already collected from your order capture system when it receives source orders from this system.
Manage Work Definitions task in the Work Definition work area. You must include this entity for a make flow. If you don't, then Global Order Promising won't provide a recommendation for the make flow, and Supply Chain Orchestration won't create a supply order.	Work Definitions

Source	Entity
Order Management still processes the order but fulfillment won't benefit from the recommendation and supply order.	

Note: You need different privileges access different work areas. You might need to create a super user that has privileges. For details, see *Privileges That You Need to Implement Order Management*.

Notes

- A pilot set up expects your test orders to use the same values for the unit of measure, currency, and currency conversion that you collect into the data repository.

  During a full set up, if more than one order capture system uses different values for these entities, then you must do more set up. For example, if one order capture system uses Ea for each item, and if another order capture system uses Each for each item, then you must set up cross-references for these order capture systems.
- Your can collect reference data from more than one system. However, a reference data entity is a global object, so Order Management and Global Order Promising use the most recent data that your set up collects.

  So, identify the source system that contains the master data list, then collect data from the source system after you collect data from all other source systems. For example, if source system x contains the master list of currencies, then collect currencies from source system y so Order Management can cross-reference currencies to system y, then collect currencies from system x.

Related Topics

- Privileges That You Need to Implement Order Management
- Overview of Data Collections for Supply Chain Planning
- Manage Planning Source Systems for Data Collections
- How You Collect Different Data Types for Supply Chain Planning

Collect Planning Data for Order Management

You collect planning data so you can select the data that you need during set up. You can collect planning data for organizations, items, structures, routings, suppliers, transit times, and so on.

Assume you create a new item, the AS54888, and you must add the AS54888 to the Item attribute in your Sourcing Rule, but how do you get the AS54888 to show up in the pick list that you use when you set the Item attribute? You must collect it first.

Try it.

1. Do your setups in various work areas. For example:

Work Area	Description
Product Information Management	○ Use the Manage Items task and Create Item Structure task to set up the AS54888 item.

Work Area	Description
	○ Use the Manage Catalogs task to set up catalogs for the AS54888.
Setup and Maintenance	○ Use the Manage Units of Measure task to set up the units of measure you need for the AS54888. ○ Use the Create Workday Pattern task and the Create Work Schedule task to set up calendar data for the AS54888.
Work Definition	Use the Manage Work Definitions task to set up the work definitions that manufacturing will use to build the AS54888.

2. Go to the Plan Inputs work area, then click **Tasks > Collect Planning Data**.
   - You use the Collect Planning Data task to collect the set ups you made in step 1. This task makes the objects that you set up available to other Oracle Applications.
   - Don't use the Plan Inputs task that's available in the Setup and Maintenance work area. Use the Plan Inputs work area instead.
3. In the Collect Planning Data dialog, set the values.

Attribute	Value
Source System	Select the source system that your implementation uses for planning, such as OPS (Oracle Order Orchestration and Planning). If you're not sure, here's how you identify the source system that your implementation uses. a. In the Setup and Maintenance work area, go to the task. - Offering: Order Management - Functional Area: Orders - Task: Manage Upstream and Fulfillment Source Systems b. On the Manage Upstream and Fulfillment Source Systems page, in the Destination System area, notice that you can use a destination system where the Enable for Oracle Fusion Distributed Order Orchestration option contains a check mark. c. In the Source Systems area, notice that the Collect Planning Data dialog gets values for the Source System attribute from source systems where the Order Orchestration attribute contains Order Orchestration.
Collection Type	Set a value. ○ **Targeted**. Delete all data in the repository for the entities that you select, collect new data for these entities, then save it in the repository. ○ **Net Change**. Collect data in increments, and collect only changed or new data. This choice is faster than using Targeted. Use Net Change when you already performed a targeted collection and now must keep your planning data current with data in your run time environment. ○ **Automatic Selection**. Let the server decide.

Templates are available, such as Static Data for Supply Planning. For this example, to help you learn how it works, do the steps instead so you can more clearly visualize the flow.

4. On the Reference Data tab, move entities from the Supply Entities list to the Selected Entities list.

Select the entities you need. Here are the entities that you typically collect for an Order Management implementation.

- Calendars
- Currencies
- Items
- Item Structures
- Order Orchestration Reference Objects
- Organizations
- Sub Inventories
- Units Of Measure

For the complete list, see *Overview of Collecting Promising Data for Order Management*.

5. Optional. Click Schedule, then set a schedule to collect automatically.

   You might need to collect some entities periodically. For example, the conversion rate between currencies, such as the Euro and the Dollar, might change each day. If you sell in markets that use different currencies, and if you don't use a currency conversion list, then you might need to collect the Currencies entity one time each day so Order Management can use the current conversion rate.
6. Click **Submit**.

   The Plan Inputs work area automatically starts the *Collection Job Set* scheduled process.
7. In the Status dialog, note the process number. For this example, assume its 50465.
8. Go to the Scheduled Processes work area.
9. Locate the value 50465 in the Process ID column. Monitor the process until the Status column displays Succeeded.

   Collection Job Set might automatically start other scheduled processes, such as Extract Oracle Fusion Entity. You can examine the log for Collection Job Set to monitor them too. For example, here's the log for a Collection Job Set that finished successfully.

```
Is Attribute Based Planning Enabled = 1
Is User Defined Attribute Based Planning Enabled = 1
The Extract Oracle Fusion Entity process started for entity ITEM_ORGS with refresh number 262023.
The Extract Oracle Fusion Entity process processed the following number of records: 1319.
```

10. Use Oracle Applications to continue your set up.

    For example, use the Create Assignment Set task in the Global Order Promising work area to specify the AS54888 item that you set up in the Product Information Management work area.

Related Topics

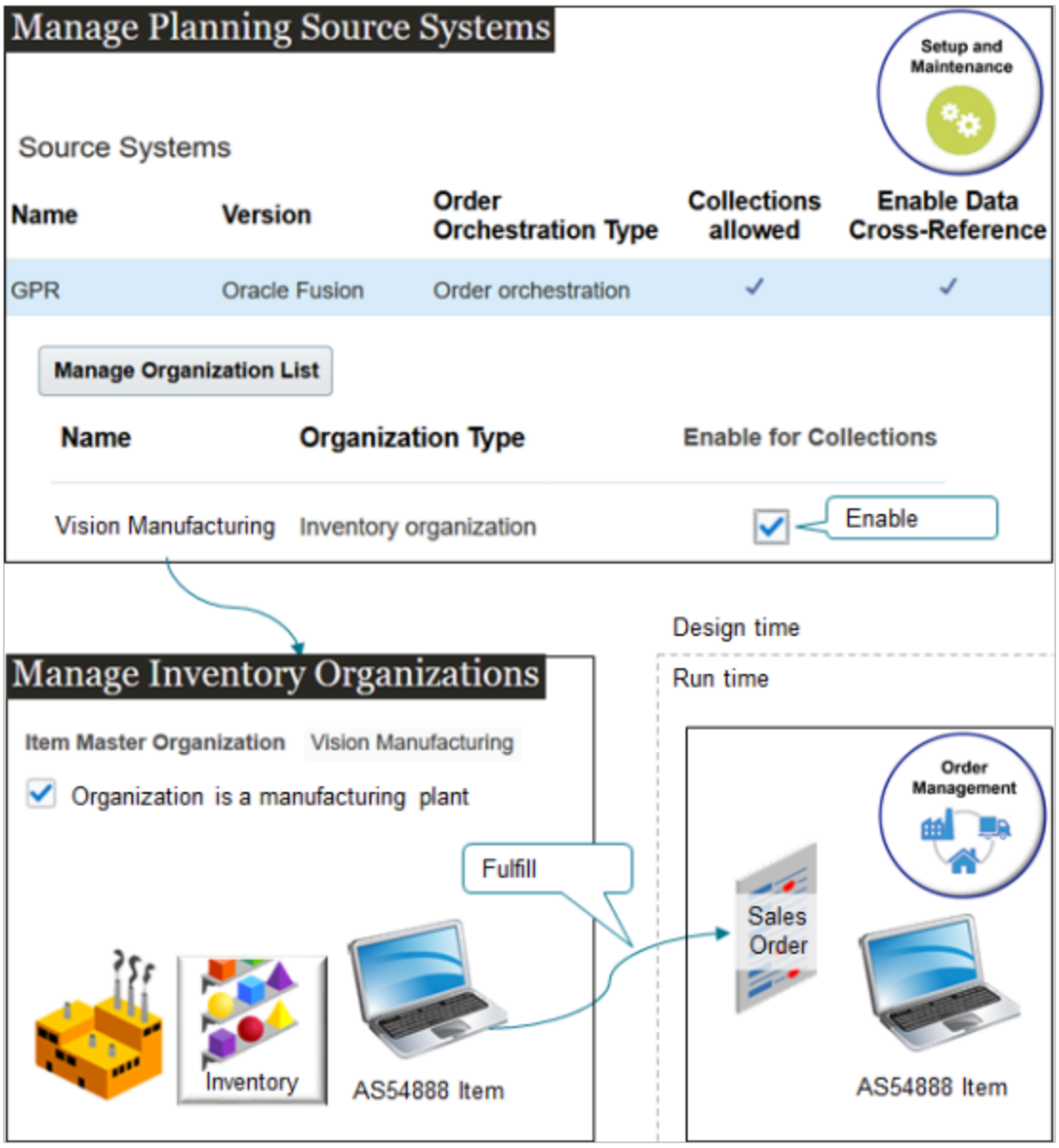
- Refresh the Order Promising Server for Order Management
- Overview of Data Collections for Supply Chain Planning
- Manage Planning Source Systems for Data Collections
- How You Collect Different Data Types for Supply Chain Planning

Collect Source System Data

Order Management uses various data, such as units of measure, currency, and currency conversions, from the Order Orchestration and Planning Data Repository. You must collect data for these entities so your set up can receive source orders.

Sourcing rules in Global Order Promising reference the Organization Parameter entity. You must collect organization parameters before you set up your sourcing rules.

Assume you must set up collections for the AS54888 Desktop Computer.



Note

- You use the Manage Planning Source Systems task to specify Oracle Order Orchestration And Planning as your source system, to specify Vision Manufacturing as the inventory organization for the AS54888, and to enable Vision Manufacturing for collections.

- You use the Manage Inventory Organizations task to specify Vision Manufacturing as a manufacturing plant.
- At run time, Order Management uses the inventory in Vision Manufacturing to fulfill the AS54888.

Summary of the Set Up

1. Set up the item.
2. Set up the inventory organization.
3. Collect data from source systems.

Set Up the Item

Set up your item so it supports order promising.

1. Go to the Product Information Management work area, then open item AS54888 for editing.
2. Click **Specifications**, click **Planning**, then set the value.

Attribute	Value
Planning Method	Set it to MPR Planning or MPS Planning. You must use one of these values. If you use any other value, then the item won't be available for promising and fulfillment will likely fail.

3. Click **Sales and Order Management**, then set the values.

Attribute	Value
Check ATP	Material and Resource
Shippable	Yes

4. Refresh the order promising server.

Entities	Value
Items	Contains a check mark.

For details, see *Refresh the Order Promising Server for Order Management*.

Set Up the Inventory Organization

Set up your inventory organization so it supports order promising.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Manufacturing and Supply Chain Materials Management
    - Functional Area: Facilities

- Task: Manage Inventory Organizations

2. On the Manage Inventory Organizations page, search for your inventory organization.

Attribute	Value
Organization Name	Vision Manufacturing

3. In the search results, click that **row** that has your organization, then click **Manage Organization Parameters**.
4. On the Manage Inventory Organization Parameters page, in the Additional Usages area, set the value.

Attribute	Value
Organization is a Manufacturing Plant	Contains a check mark.

5. Go to the task.
   - Offering: Manufacturing and Supply Chain Materials Management
   - Functional Area: Manufacturing Master Data
   - Task: Manage Plant Parameters
6. Verify that the value in the Manufacturing Calendar attribute is the correct calendar.

   For details, see *How to Create a Calendar used by Order Management and Global Order Promising (Doc ID 2204151.1)*.
7. Collect planning data and refresh the order promising server.
   - Select all entities when you do the Collect Planning Data task.
   - Select all entities when you refresh the server.

   For details, see *Collect Planning Data for Order Management*.

Collect Data From Source Systems

1. Collect data from your source system.
   - In the Setup and Maintenance work area, go to the task.
     - Offering: Supply Chain Planning
     - Functional Area: Supply Chain Planning Configuration
     - Task: Manage Planning Source Systems
   - On the Manage Planning Source Systems page, in the Source Systems list, locate your Oracle system, then set the values.

Attribute	Value
Version	Fusion

Attribute	Value
Collections Allowed	Contains a check mark
Order Orchestration Type	Order Orchestration
Enable Data Cross-Reference	Contains a check mark

- Set the values for each system that your set up must integrate, such as an order capture system or fulfillment system.

Attribute	Value
Version	Others
Collections Allowed	Contains a check mark
Enable Data Cross-Reference	Contains a check mark

For details, see *Manage Planning Source Systems for Data Collections*.

2. Collect data for Vision Manufacturing, which is the organization that contains the inventory you plan to use to fulfill the item.
   - Select your source system in the Source Systems list, then click **Manage Organization List**.
   - In the Manage Organization List dialog, click **Refresh Organization List**.
   - Add a check mark to the Enable for Collections option for each organization that you use to fulfill the item, then click **Save and Close**.

Attribute	Value
Name	Vision Manufacturing
Enable for Collections	Contains a check mark

3. Collect planning data. You must do the Collect Planning Data task each time after you do the Manage Planning Source Systems task. Here are your values for this example.

Parameter	Value
Source System	OPS

Parameter	Value
Collection Type	Targeted
Reference Data	Move the Items entity and the Organizations entity to the Selected Entities window.

For details, see *Collect Planning Data for Order Management*.

4. Refresh the Order Promising Server for Order Management.

Parameters	Value
Items	Contains a check mark.
Organizations	Contains a check mark.

For details, see *Refresh the Order Promising Server for Order Management*.

Related Topics

- Refresh the Order Promising Server for Order Management
- Collect Planning Data for Order Management
- Overview of Data Collections for Supply Chain Planning
- Manage Planning Source Systems for Data Collections

Collect Runtime Data

Collect data that changes dynamically at run time, such as on-hand supply and purchase orders.

Global Order Promising makes available-to-promise and sourcing decisions when your user creates a sales order at run time. It allows the user to explore different scenarios to increase margin, improve delivery, and so on. Promising needs the most up-to-date supply and demand data to make these decisions.

Its important to periodically collect data for each new item that your users add when they create a sales order because collecting gets the on-hand quantity for return orders and canceled orders, which makes that quantity available for other sales orders. The quantity affects planning.

Try it.

1. Go to the Plan Inputs work area, then click **Tasks > Collect Planning Data**.

2. In the Collect Planning Data dialog, set a value in the Source System attribute, click **Supply Planning Data**, then move entities from the Supply Entities list to the Selected Entities list.

Entity	Value
On Hand	Promising uses this entity to get the inventory that's in stock for the item.
Purchase Orders and Requisitions Transfer Orders Work Order Supplies	Add these entities so Global Order Promising can get availability across the entire supply chain.

3. Click **Submit**.
4. Refresh the Global Order Promising server.

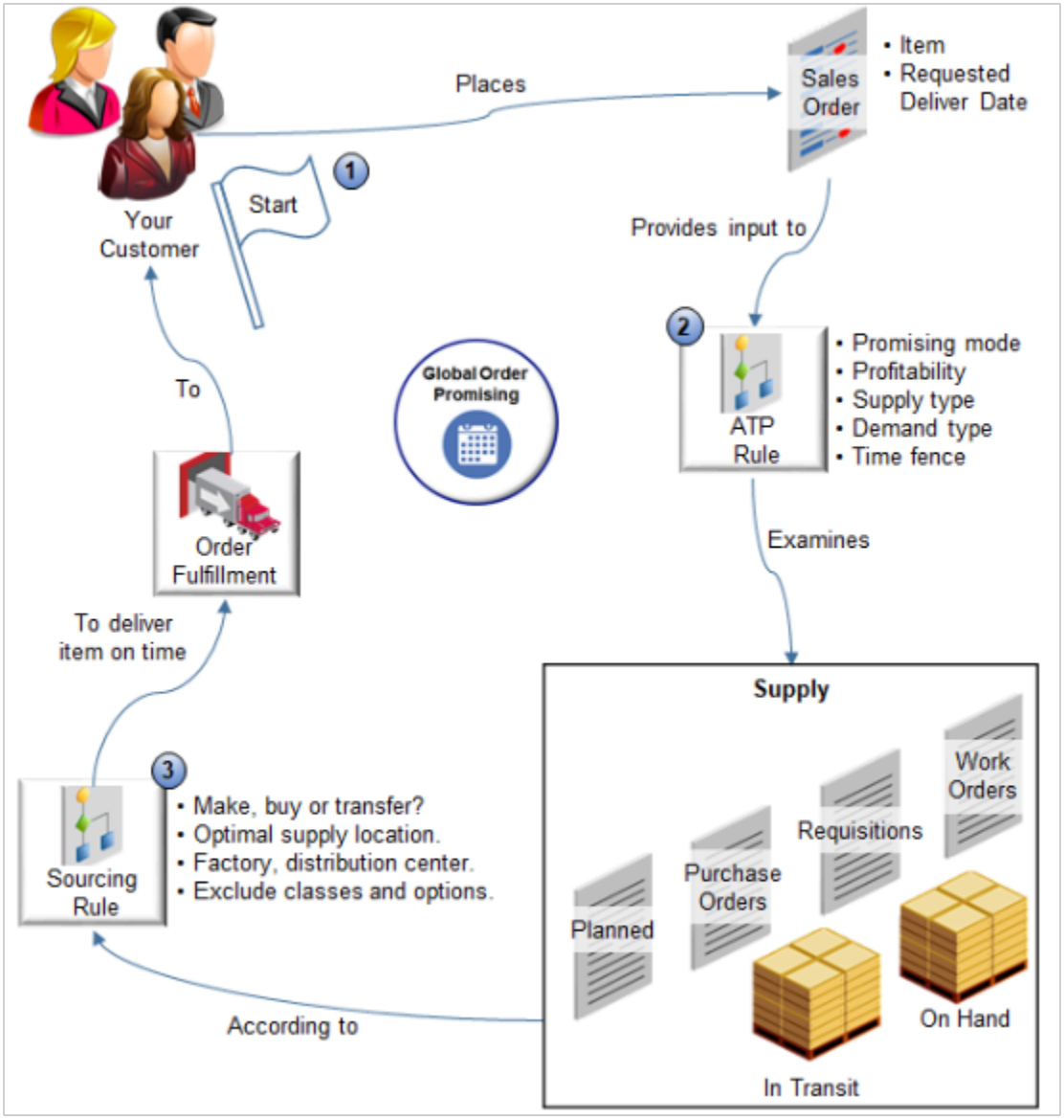
Related Topics

- Collect Planning Data for Order Management
- Collect Data for Global Order Promising
- Overview of Data Collections for Supply Chain Planning
- Manage Planning Source Systems for Data Collections
- How You Collect Different Data Types for Supply Chain Planning

Set Up Promising Rules and Sourcing Rules for Order Management

Use an available-to-promise (ATP) rule and a sourcing rule to promise your sales order in different ways.

Global Order Promising uses sourcing rules, assignment sets, and available-to-promise rules when it determines availability and schedules an order line for Order Management.



Note

1. Your customer places a sales order that includes the item and requested delivery date.
2. You create an available-to-promise rule, which is a set of instructions you specify that tells Order Promising how to analyze supply that's available in your supply chain so it can promise the item and meet the delivery date. You specify the supply type to consider, such as supply that's on hand or in transit. You can also specify supply that various documents create, such as purchase orders, requisitions, or work orders.
3. You create a sourcing rule that specifies the supply sources to consider when promising, such as whether to consider make, buy, or transfer sources. You can also specify the optimal location that can supply the demand.

Summary of the Setup

1. Create your sourcing rule.
2. Assign your sourcing rule.
3. Create your available-to-promise rule.
4. Manage the administrator profiles.
5. Refresh the server.

This topic uses example values. You might need different values, depending on your business requirements.

Create Your Sourcing Rule

Assume you must set up a relatively simple sourcing rule for the AS54888 item. You transfer it from an organization that stores inventory, such as Vision Manufacturing. For details about how to set up sourcing in other contexts, see the Manage Sourcing Rules section in *Set Up Drop Ship in Order Management* and *Create Sourcing Rules for Your Configuration Model*.

1. Go to the Global Order Promising work area, then click **Tasks > Manage Sourcing Rules**.
2. On the Manage Sourcing Rules page, Click **Actions > Create**, then set the values.

Attribute	Value
Name	Sourcing Rule for the AS54888 Item You can use any text.
Organization Assignment Type	Global Set to. - **Global** when you must specify where to fulfill and ship the sales orders. You don't specify an organization to create supply. Instead, you specify a transfer or buy source. - **Local** when you must specify how to create supply and the organization that creates it. You create a global sourcing rule that specifies the warehouse that Order Management uses when it fulfills the sales order. For details, see *Source Your Supply Chain*.

3. In the Sourcing Rule Effective Dates area, click **Actions > Add Row**, then set the start date.
   - You must set a start date.
   - As an option, you can also set an end date. If you don't set an end date, then the rule never expires.
   - If the runtime date occurs before the start date or after the end date, then you might encounter a runtime error that states order promising can't schedule the fulfillment line.
   - If the requested ship date or the requested arrival date on the order line happens before your rule's start date or after your rule's end date, then you might encounter a runtime error that states order promising can't schedule the fulfillment line.
4. In the Sources area, Click **Actions > Add Row**, then set the values.

Attribute	Value
Type	Transfer From Use Transfer From to transfer from an inventory organization. Global Order Promising enables Make At only when you set assignment type to Local.
Organization	Vision Manufacturing

Attribute	Value
Allocation Percent	100 In this example, you add only one source, so specify 100%. If you add more than one source, then you can allocate demand across sources. For example, if you add a row for Vision Manufacturing and set allocation to 70%, add another row for Vision Distribution and set allocation to 30%, then Promising will use Vision Manufacturing to promise 70% of the orders.
Rank	1 If you add more than one source, then you can specify the rank order to use for sources. For example, if you add a row for Vision Manufacturing and set Rank to 1, add another row for Vision Distribution and set Rank to 2, then Promising will use Vision Manufacturing to promise the order first. If Promising determines that Vision Manufacturing can't fulfill the order, then Promising will consider Vision Distribution.
Shipping Method	Global Order Promising disables Shipping Method for a local rule. You can't edit it. Leave it empty for a global rule. If you set a value for a global rule, you might get an error. `The value provided for the Shipping Method attribute is invalid.` Don't set shipping method in this context because it specifies where supply originates, not how to ship it to the customer.
Exclude for Options and Option Classes	Exclude options and option classes when promising a sales order. Exclude them for a Make At or Buy From sourcing type. For example, exclude an item from planning when you know your source can't make it because it includes toxic chemicals that the source isn't authorized to handle, or your company limits production to only one specific site. For another example, assume you know Seattle Manufacturing created a large oversupply of the CTO_474100 screen option class from a prior marketing campaign. You already know supply is available. To improve planning performance, you decide to exclude it from planning.

5. Click **Save > Save and Close**.

Assign Your Sourcing Rule

1. Click **Tasks > Manage Assignment Sets**.
2. On the Manage Assignment Sets page, click **Actions > Create**.
3. On the Create Assignment Set page, set the values.

Attribute	Value
Name	Assignment Set for Sourcing Rules
Catalog	GOP_Catalog Use the same catalog that you use for your item in Product Information Management and in your available-to-promise rule.

4. In the Sourcing Assignments area, click **Actions > Add Row**, then set the values.

Attribute	Value
Assignment Level	Item Note o Promising fulfills your sales order only from the source that you assign to the assignment set. o Assign at least one sourcing rule at the global level so Global Order Promising can use it to identify a ship-from location. If you don't, then your users must manually set a value in the Warehouse attribute on each sales order.
Item	AS54888
Sourcing Type	Sourcing Rule
Sourcing Rule or Bill of Distribution	Sourcing Rule for the AS54888 Item

For details, see *Set Your Assignment Levels*.

Create Your Available-To-Promise Rule

1. Click **Tasks > Manage ATP Rules**.
2. On the Create ATP Rule page, set values.

Attribute	Value
Name	ATP Rule for the AS54888 Item
Description	Rule that specifies how to determine availability for the AS54888 item.
Promising Mode	Infinite Availability

Attribute	Value

3. Click **ATP Rule Assignment**, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Assignment Basis	Item and Organization
Assigned-to Organization	Vision Manufacturing Use the same value that you use in the Organization attribute in your sourcing rule.
Assigned-to Item	AS54888

Manage the Administrator Profiles

1. In the Setup and Maintenance work area, click **Search**, search for, then open the Manage Administrator Profile Values page.
2. On the Manage Administrator Profile Values page, set the value, then click **Search**.

Attribute	Value
Application	Global Order Promising

3. In the search results, click the row that has MSP_DEFAULT_ASSIGNMENT_SET in the Profile Option Code column.
4. In the Profile Values area, click **Actions > New**, set the values, then click **Save and Close**.

Attribute	Value
Profile Level	Site
Profile Value	Pick the name of the assignment set that you created earlier in this topic. In this example, pick Assignment Set for Sourcing Rules.

5. On the Manage Administrator Profile Values page, set the value, then click **Search**.

Attribute	Value
Profile Option Code	MSC_SRC_ASSIGNMENT_CATALOG

Attribute	Value

6. In the search results, in the Profile Values area, set the value.

Attribute	Value
Profile Value	Set to the same value that you use with the assignment set. For this example, set it to GOP_Catalog.

Refresh the Server

For details, see *Refresh the Order Promising Server for Order Management*.

Related Topics

- Refresh the Order Promising Server for Order Management
- Create Sourcing Rules for Your Configuration Model
- How the Order Orchestration and Order Promising Processes Use the Collected Planning Data
- Overview of Security Console
- Assignments and Promising Rules

Refresh the Order Promising Server for Order Management

You must refresh the Order Promising Server each time you collect data for Order Management. You must also refresh each time you create or update a promising rule or sourcing rule.

1. Go to the Scheduled Processes work area, then click **Action > Schedule New Process**.
2. In the Schedule New Process dialog, set the value.

Attribute	Value
Name	*Refresh and Start the Order Promising Server*

3. Add a check mark to the parameters that you must update, then click **Submit**.
   Here are the parameters you refresh for most set ups.
   - On Hand. You need the on-hand inventory to replenish each item in an inventory transaction or to process a return order.
   - ATP Rules.
   - Sourcing.
   - Items.
   - Organizations. Include only when you create a new inventory organization.
   - Resources. Include only when you create a new resource.

4. Click **Actions > Refresh**, then verify the status is Succeeded. Wash, rinse, and repeat, as necessary.

Related Topics

- Collect Data for Global Order Promising
- Overview of Data Collections for Supply Chain Planning
- Manage Planning Source Systems for Data Collections
- How You Collect Different Data Types for Supply Chain Planning

Display Customer Details

Overview of Displaying Customer Details on Sales Orders

Control how Order Management displays customer details on sales orders, such as ship-to address, bill-to address, contacts, and payment terms.

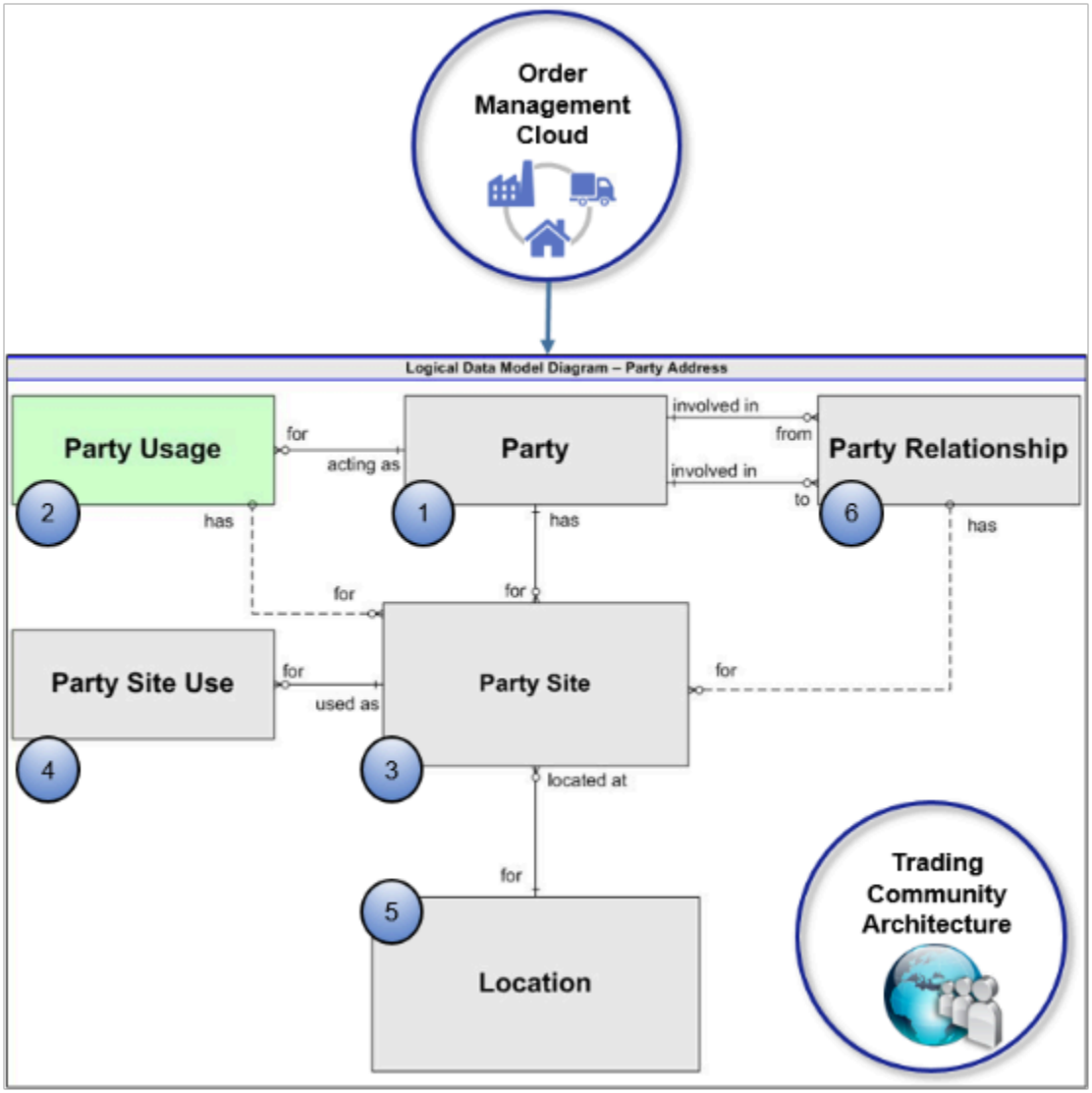
For example:

- Set the default value that the Order Management work area displays for each customer attribute and address attribute.
- Use an address from a related customer to set the default value for ship-to address.
- Use data from the customer master to set the default value for preferences on each sales order.

Order Management gets customer details from Oracle Trading Community Architecture (TCA). Trading Community Architecture is a data model you can use to manage details about customers who belong to your community, such as organizations, locations, and the relationships that define your community. For details, see *Oracle Trading Community Architecture User Guide*.

Party Object

The party object in Trading Community Architecture contains customer data. Order Management uses it to get the customer data it displays.



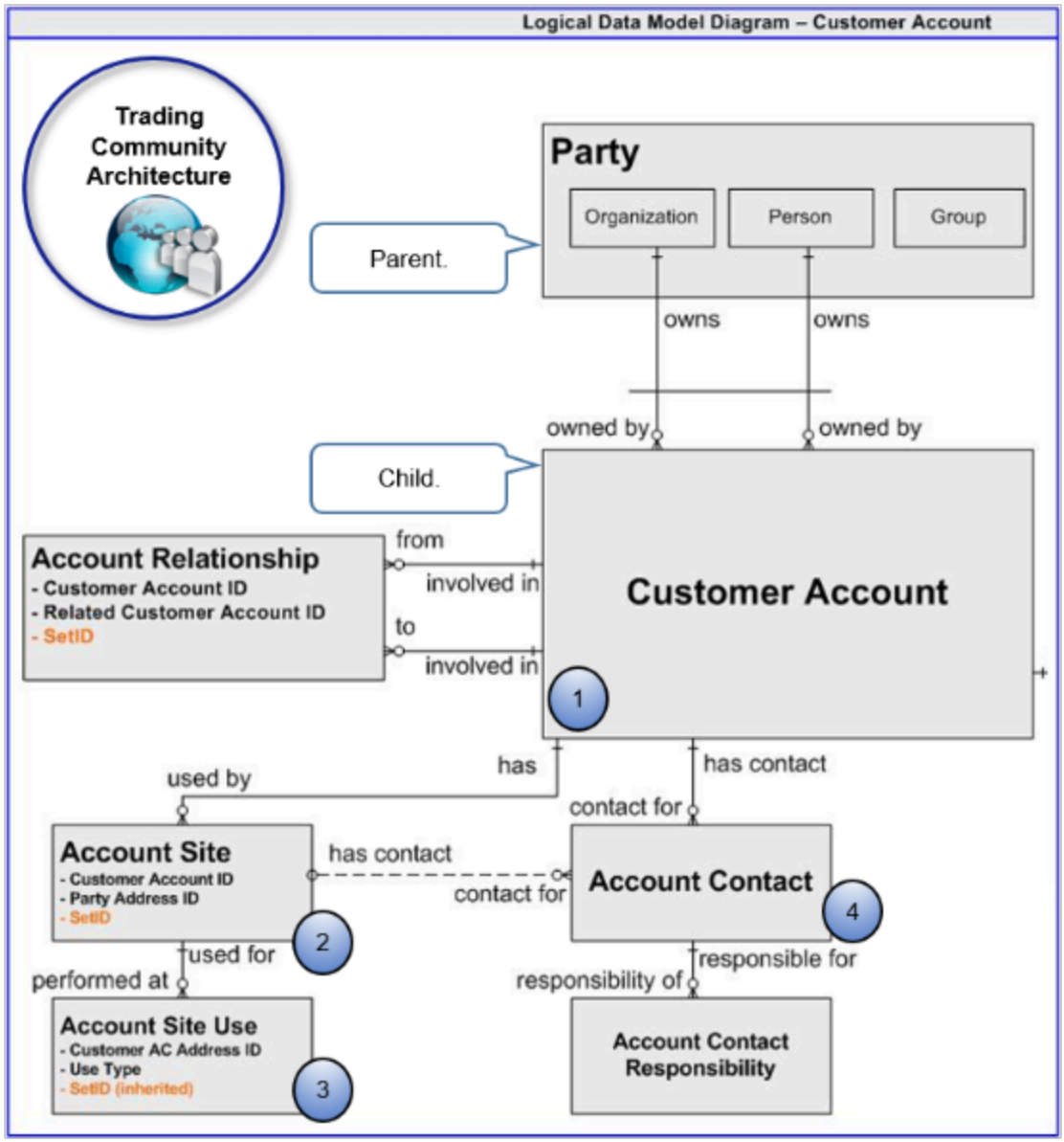
Note

Object	Description
1. Party.	A trading partner. Each party can reference one or more party usages.
2. Party Usage.	How you use the party, such as customer, supplier, prospect, and so on. Order Management uses only the customer party usage. For example, searches and lists of values in the Order Management work area display only customer party usage.
3. Party Site.	An address that the party uses. Each party can reference one or more party sites.
4. Party Site Use.	How the party uses the address. For example, ship-to, bill-to, and so on. Each party site can reference one or more party site usages.
5. Location.	Physical address where the party site resides. Each party site can reference only one location.

Object	Description
6. Party Relationship.	Each party can reference one or more other parties. You can use the party relationship to establish a relationship between two different parties.

Account Object

Order Management uses the customer account object from Trading Community Architecture.



Note

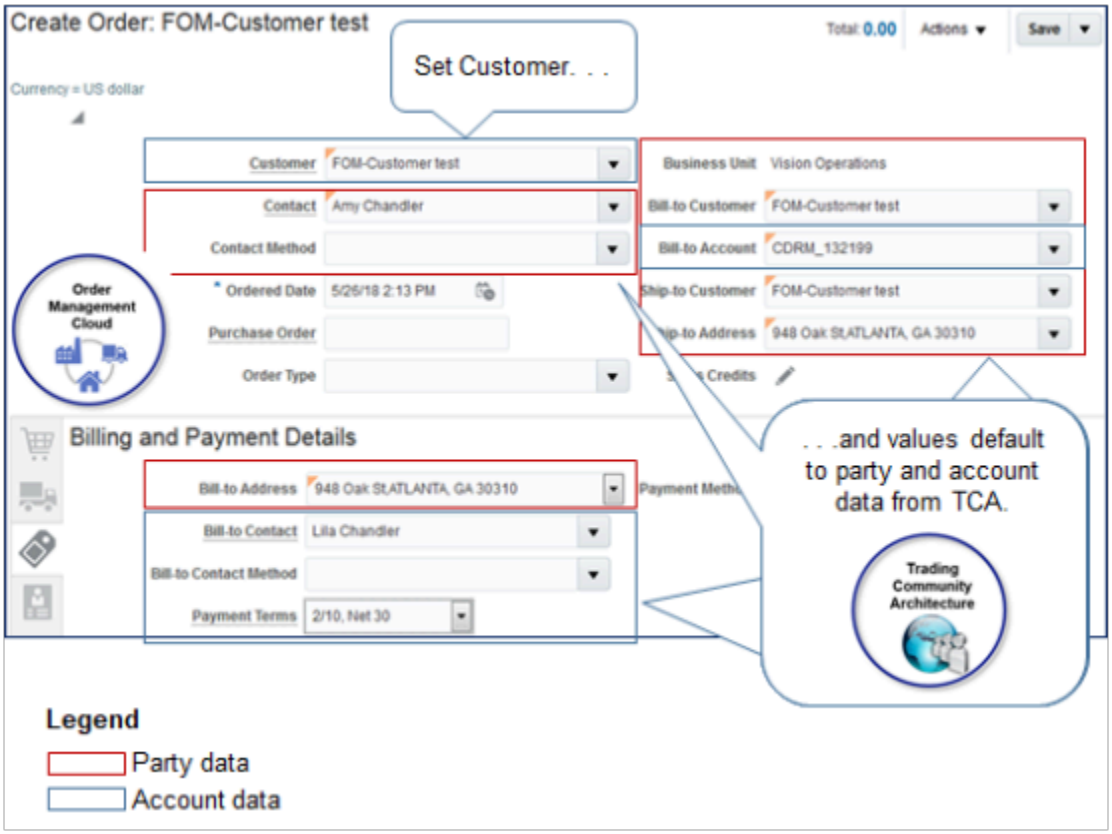
Object	Description
1. Customer Account.	The customer account is a child of the party. It includes financial details that Order Management uses to communicate with a financial application. Each party can reference one or more customer accounts, however each party typically references only one customer account.
2. Account Site.	Set of party addresses that the account uses. Each customer account can reference one or more account sites.
3. Account Site Use.	How the account site uses the address. Each account site can reference one or more account site usages.
4. Account Contact.	Person in the account. Each account can reference one or more contacts. Account contact is a subset of party contacts.

Where Order Management Displays Customer Details

Here's how Order Management displays customer data.

- Uses the party object to display data for customer attributes, such as Customer, Bill-to Customer, or Ship-to Customer.
- Uses the account object to display data for account attributes, such as Bill-to Account or Bill-to Address.

If the Order Entry Specialist sets the value for the Customer attribute on the order header, then Order Management automatically sets the value for other attributes to the data it gets from the party and account, such as contact, ship-to customer, address, bill-to customer, account, payment terms, and so on.



Note

- In Oracle Applications, the party object contains data for each customer, and the account object contains data for the bill-to account and bill-to address.
- In Oracle eBusiness Suite, the account object contains data for each customer. For example, the account contains the ship-to address, sold-to address, and bill-to address.

Collect Data

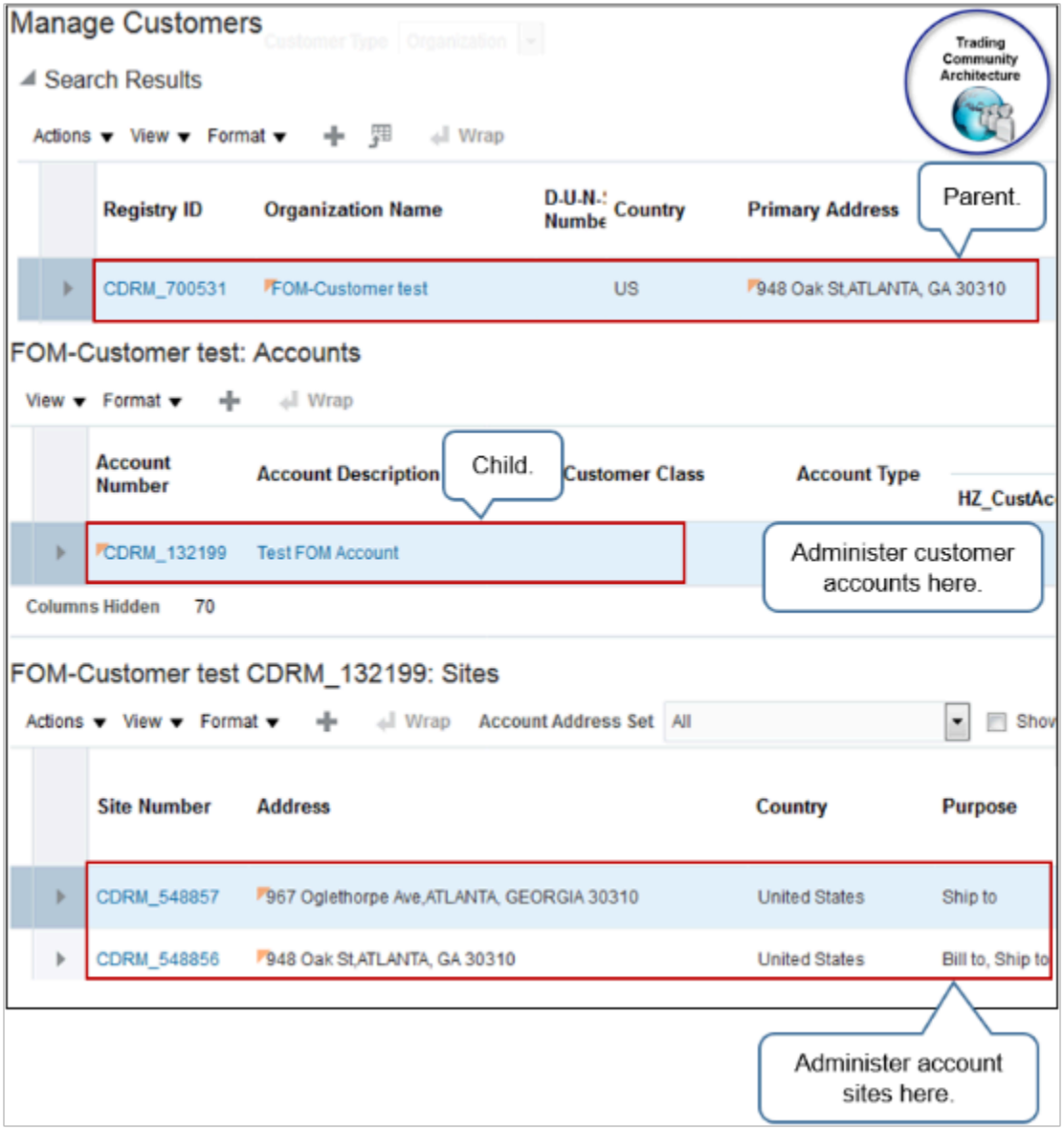
You must collect data any time you modify party data. For details, see *Collect Planning Data for Order Management*.

Display Account Details on Sales Orders

Control how Order Management displays account details on a sales order.

1. Get the license you need to use Oracle Financials.
2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Customers
    - Task: Manage Customers
3. On the Manage Customers page, search for a customer or create a new one.

    If you create a new account site, then the application also creates a party site that includes objects from the account site. This behavior helps to maintain the relationship between the party and the account.



Note

- Don't delete a contact or address or set the end date for a contact or address. If you delete a contact or address, then sales orders that already exist might not display the correct data.
- Each account is a child of a party. If you add a contact, account site, or account site usage in the account, then Manage Customers adds objects in the party that correspond to the contact, account site, or account site usage that you added.
- If you delete or end date an object in a child contact, account site, or account site usage, then Manage Customers doesn't update the corresponding object in the parent party.

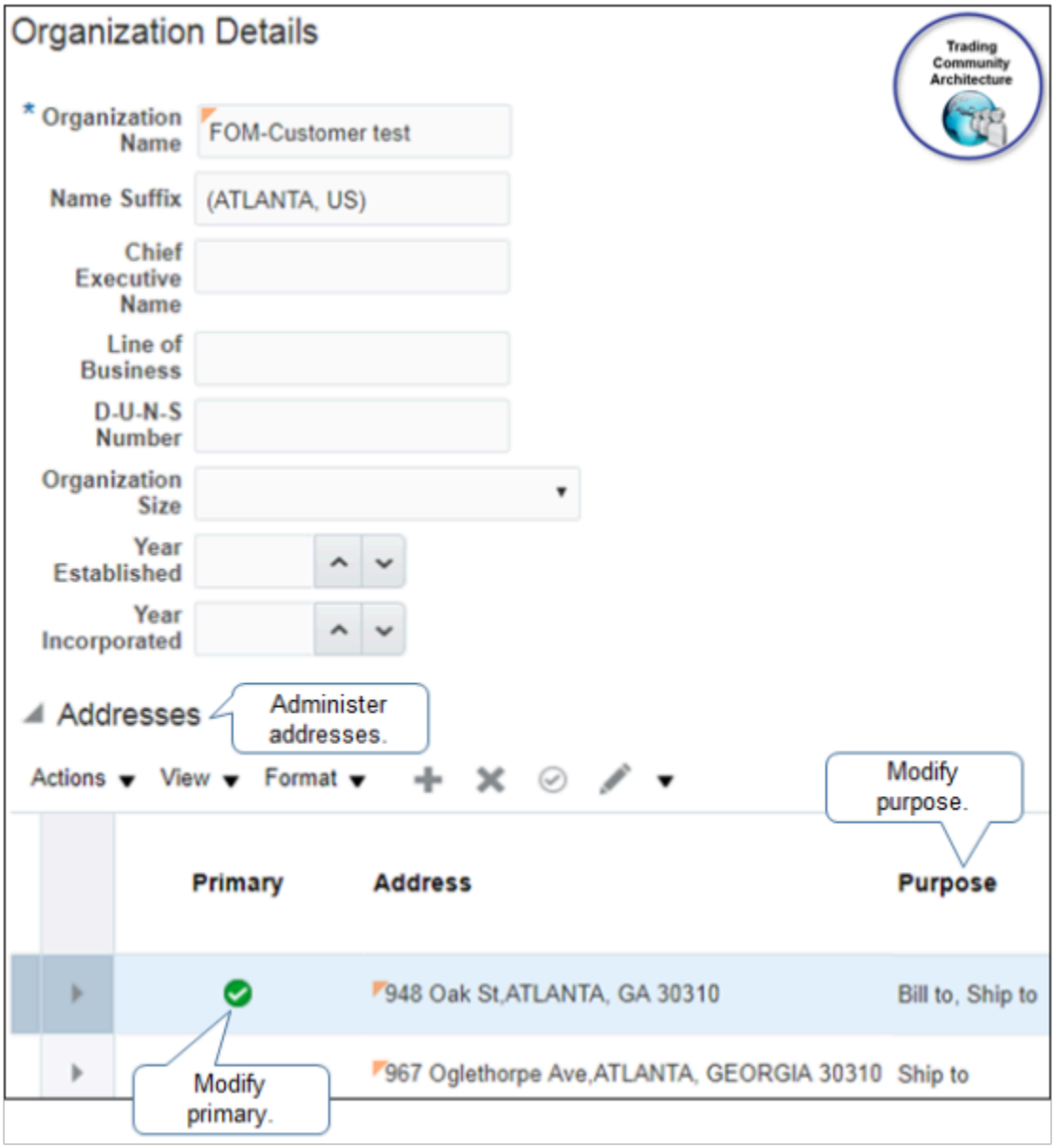
  For example, if you delete the ship-to address of a contact, then Manage Customers doesn't delete the corresponding ship-to address of the party. Manage Customers uses this functionality to maintain the relationship between party and customer account, which is one to many. Some other account might reference the ship-to address of the party.

Display Party Details on Sales Orders

Set up the party details that display on a sales order in Order Management.

You can't use the Manage Customers page to manage the parent party. Instead, do these steps.

1. Get the license that you need to use Oracle CX Sales.
2. Make sure you have the privileges that you need to administer customer data, such as TC_DATA_STEWARD.
3. Go to the Organizations work area.
4. On the Organizations page, search for your party according to organization name.
5. Use the Organization Details page to manage the party profile, party address, and party usage.



Note

- Modify the value in the Purpose attribute to control site usage. For example, Order Management uses Ship-to purpose to filter the list of values that it displays for ship-to address.

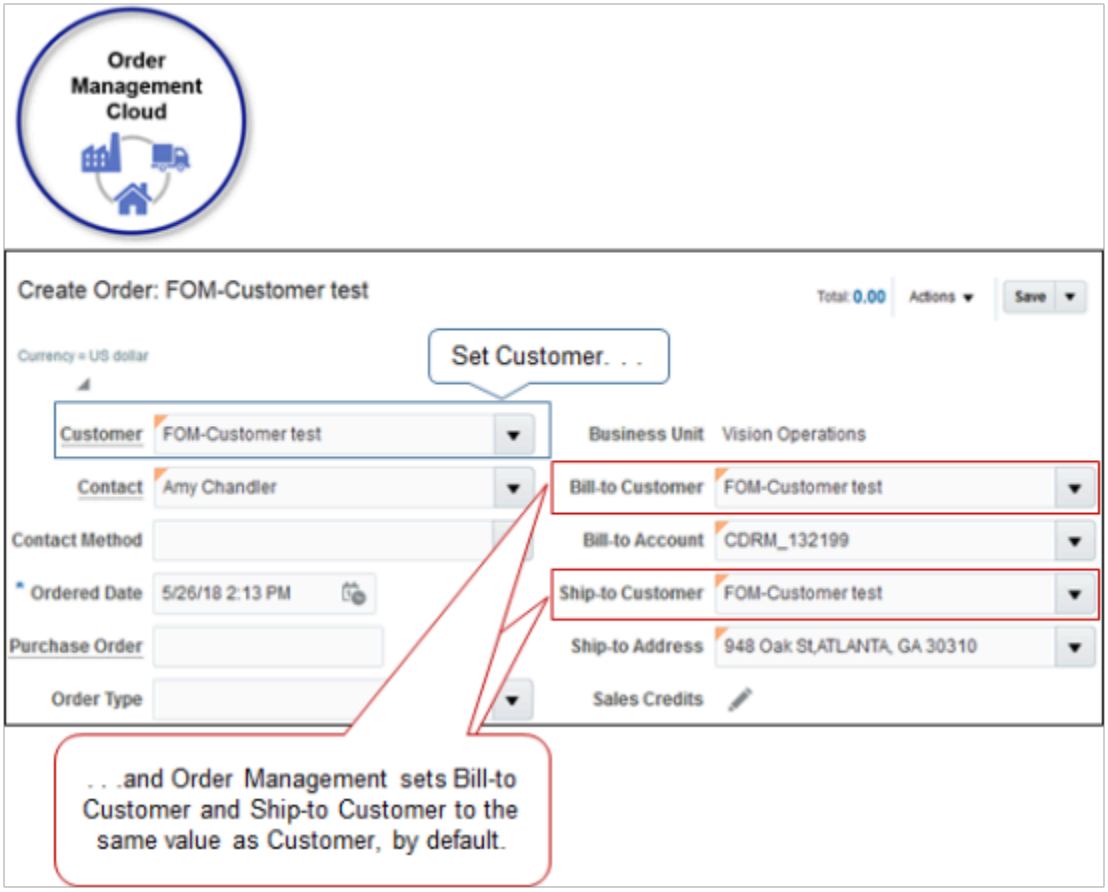
- Add a check mark to the Primary attribute to specify the address that Order Management displays as the default value.
- You can't use the Manage Organization page to manage accounts.

Learn how to manage an organization. For details, see *Implementing Customer Data Management* and *Using Customer Data Management*.

Display Default Values for Customer Attributes

Control how Order Management displays default values for customer attributes on sales orders.

If the Order Entry Specialist sets the Customer attribute on the order header, then Order Management automatically sets Bill-to Customer and Ship-to Customer to the same value that Customer contains, by default. You can modify this behavior.

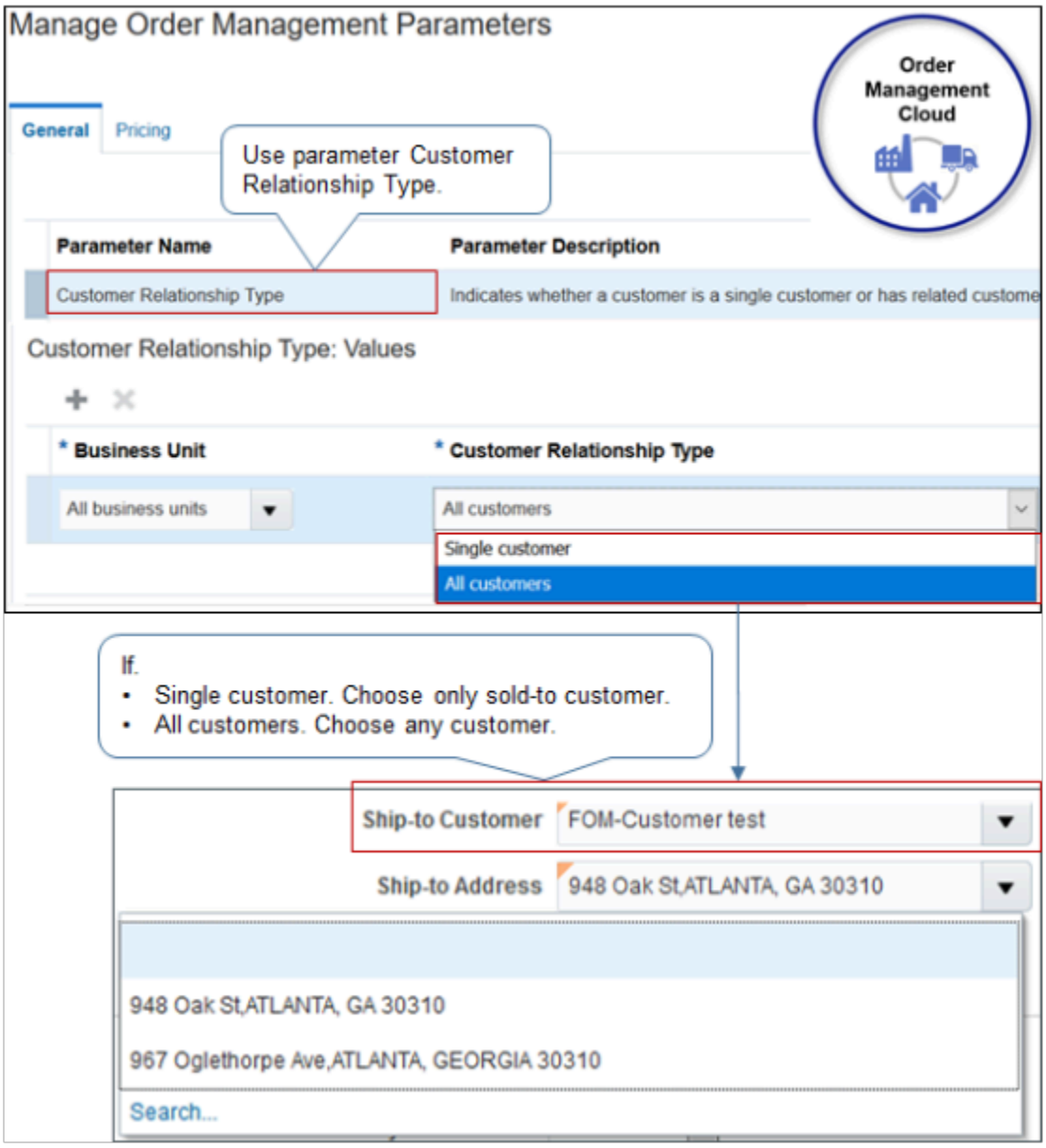


To control the values that the user can select, set the Customer Relationship Type parameter.

Value	Description
All Customers	Allow the user to select any customer for Bill-to Customer and Ship-to Customer.
Single Customer	Allow the user to select only the same value that the Customer attribute references for Bill-to Customer and Ship-to Customer.

Value	Description
Business Unit	Specify the business unit where this behavior applies. Use All Business Units to apply behavior to all sales orders.

For details, see *Manage Order Management Parameters*.



Note

- Order Management applies the behavior you specify to sales orders in the Order Management work area and to sales orders that you import.
- Order Management doesn't use relationships that you set up in Trading Community Architecture. If you must create a relationship, then set Customer Relationship Type to All Customers, and write an order management

extension that enforces ship-to or bill-to customer for each party that Trading Community Architecture defines. For details, see *Overview of Creating Order Management Extensions*.

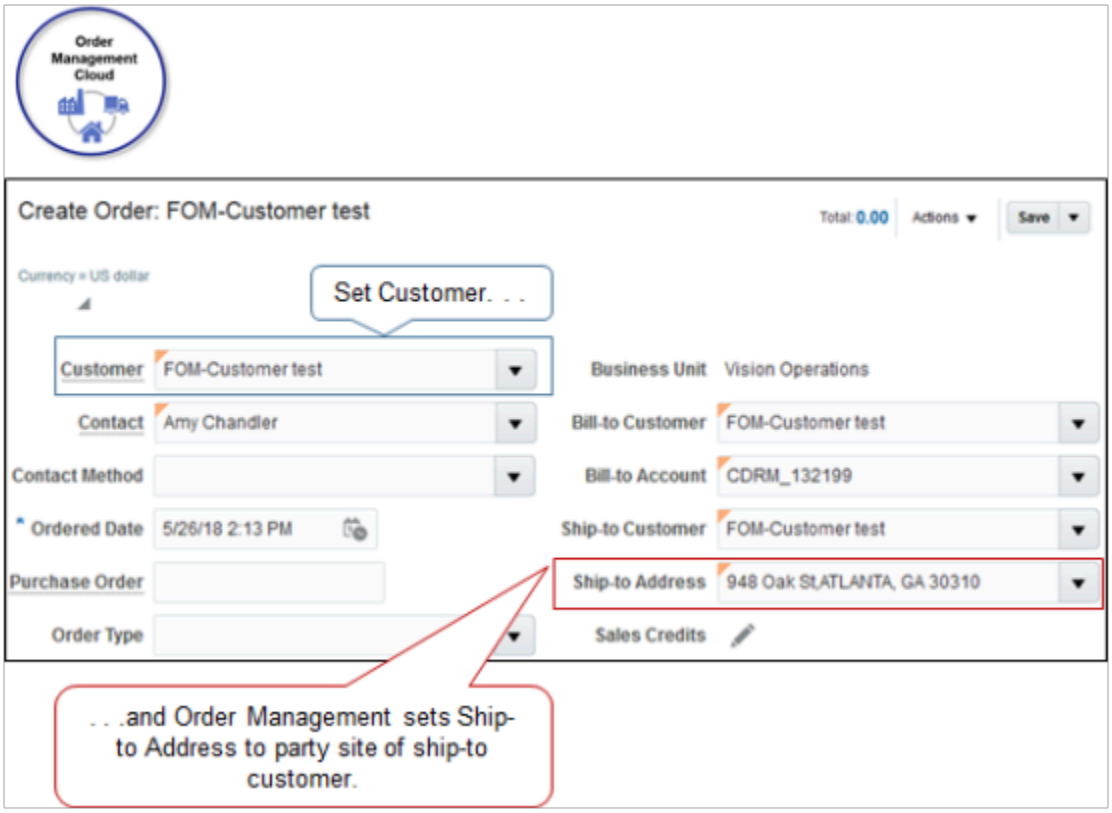
- For other relevant details, see *Manage Order Management Parameters*.

Display Ship-to Addresses On Sales Orders

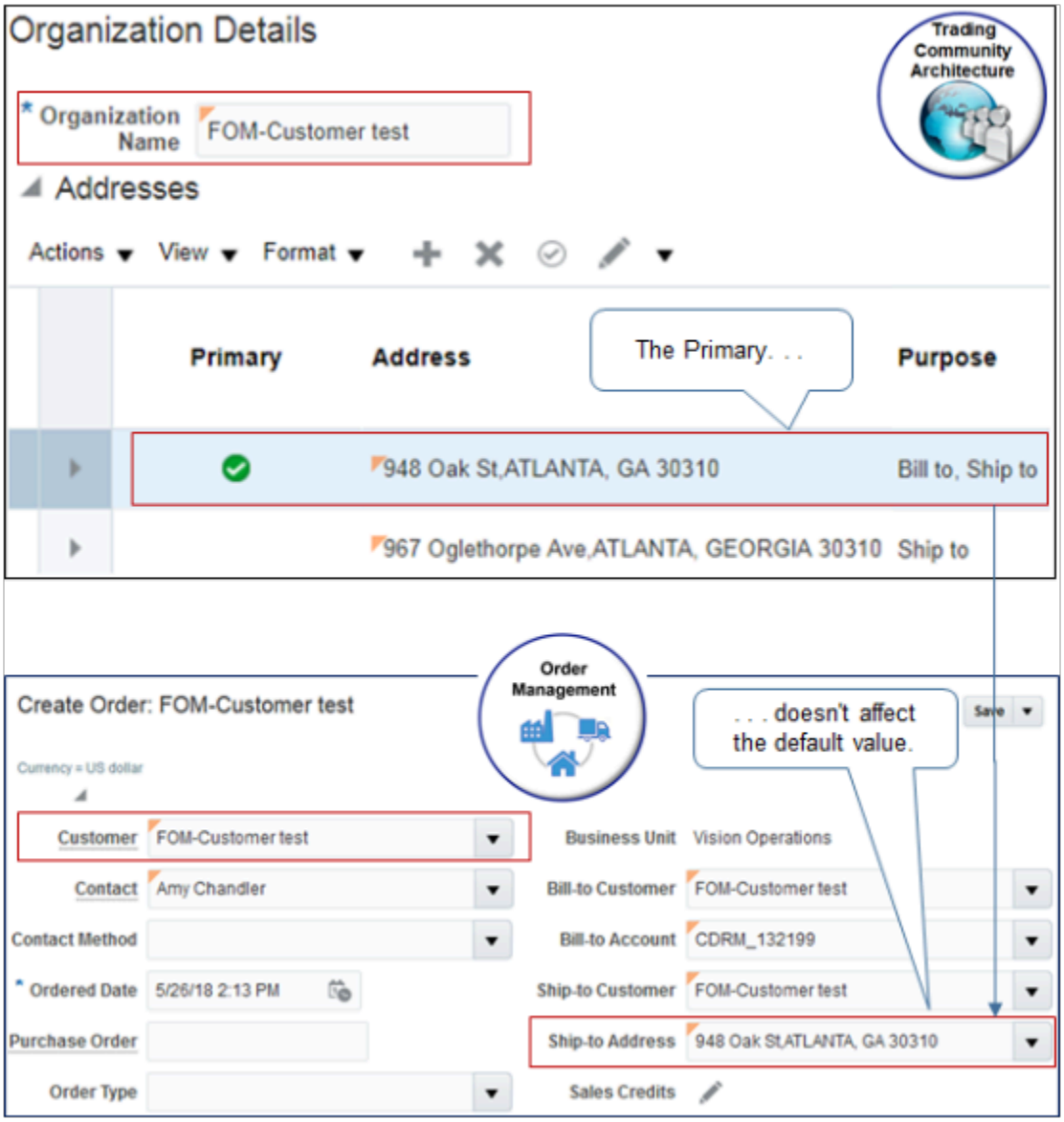
Control how Order Management displays the ship-to address on a sales order.

Set Default Value for Ship-to Address

If you set the value for the Customer attribute on the order header, then Order Management sets the value for the Ship-to Address attribute to the first address that you add on the Organization Details page, by default.



Setting the Primary attribute on the Organization Details page doesn't affect the Ship-to Address on the sales order.



Instead, you can use the OrganizationService web service to specify the default value to use for the ship-to address:

1. Make sure you have the HZ_ENTER_TRADING_COMMUNITY_ORGANIZATION_INFORMATION_PRIV privilege.
2. Use the findOrganization operation to get the values that you will use to identify party details in the mergeOrganization operation, such as PartyId, PartySiteId, and PartySiteUseId. For example:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/
applicationModule/types/">
 <ns1:findOrganization>
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>-1</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
```

```
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute>PartyName</ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>Computer Service and Rentals</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute>false</ns2:excludeAttribute>
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns3="http://xmlns.oracle.com/adf/svc/types/">
 <ns3:retrieveAllTranslations>false</ns3:retrieveAllTranslations>
 </ns1:findControl>
 </ns1:findOrganization>
 </soap:Body>
</soap:Envelope>
```

For more, see *Get Organizations with PartySiteNumber*.

3. Set the default value for the Ship-to Address attribute to the primary ship-to site. Use the details from the response that you received in step 2. Assume you need to set it to party site 300100178657747:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/applicationModule/
types/"
xmlns:org="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/"
xmlns:par="http://xmlns.oracle.com/apps/cdm/foundation/parties/partyService/"
xmlns:sour="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/sourceSystemRef/"
xmlns:con="http://xmlns.oracle.com/apps/cdm/foundation/parties/contactPointService/"
xmlns:con1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/contactPoint/"
xmlns:org1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/organization/"
xmlns:par1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/partySite/"
xmlns:rel="http://xmlns.oracle.com/apps/cdm/foundation/parties/relationshipService/"
xmlns:org2="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/orgContact/"
xmlns:rel1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/relationship/">
 <soapenv:Header/>
 <soapenv:Body>
 <typ:mergeOrganization>
 <typ:organizationParty>
 <org:PartyId>300100178657728</org:PartyId>
 <org:PartySite>
<par:PartySiteId>300100178657747</par:PartySiteId>
 <par:PartySiteUse>
<par:PartySiteUseId>300100178657752</par:PartySiteUseId>
<par:PrimaryPerType>Y</par:PrimaryPerType>
 </par:PartySiteUse>
 </org:PartySite>
 </typ:organizationParty>
 </typ:mergeOrganization>
 </soapenv:Body>
</soapenv:Envelope>
```

Use Web Service to Set Ship-to Address and Ship-to Contact

If you set the value for the Customer attribute on the order header:

- Order Management sets the value for the Ship-to Address attribute on the order header to the first address that you add and that has a value of Ship-To in the Purpose attribute on the Organization Details page, by default. Setting the Primary attribute on the Organization Details page doesn't affect the Ship-to Address on the sales order.

- Order Management sets the value of the Ship-to Contact attribute on the order header to the first contact that you add on the Account Site page. Setting the Primary attribute for a contact on the account site has no effect on the value that Order Management uses to set the default value for the Ship-to Contact on the sales order.

Assume you go to the Edit Organization page and add the 500 Oracle Parkway address. Next, you add the 600 Oracle Parkway address to the same organization, but you also enable the Primary attribute for 600 Oracle Parkway. You set the Purpose attribute for both addresses to Ship-To. At run time, Order Management will set the Ship-to Address attribute on the order header to 500 Oracle Parkway because it's the first address that you added.

Assume you add 500 Oracle Parkway as an account site to your organization, then add Yu Li as a contact on the site. Next, you add June Tsai as a contact on the same site, but you also enable the Primary attribute for June. At run time, Order Management will set the Ship-to Contact attribute on the order header to Yu Li because it's the first contact that you added.

You can't use the Organizations work area to modify this behavior. Instead, you can use a web service:

1. Use the findOrganization operation of the organizationService web service to get the values that you need to identify your party details.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/
applicationModule/types/">
 <ns1:findOrganization>
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>-1</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute>PartyName</ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>Computer Service and Rentals</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute>false</ns2:excludeAttribute>
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns3="http://xmlns.oracle.com/adf/svc/types/">
 <ns3:retrieveAllTranslations>false</ns3:retrieveAllTranslations>
 </ns1:findControl>
 </ns1:findOrganization>
 </soap:Body>
</soap:Envelope>
```

   where
   - PartyName identifies your customer, such as Computer Service and Rentals.
2. Examine the response from step 1 to get the values of the attributes that contain the person you want to use for the ship-to address and the ship-to contact. Assume the response contains these values:

Attribute	Value
PartyId	1006

Attribute	Value
RelationshipRecId	999990007700611
RelationshipId	5041

3. Use the mergeOrganization operation of the organizationService web service.

```
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/applicationModule/types/"
 xmlns:org="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/" xmlns:par="http://
xmlns.oracle.com/apps/cdm/foundation/parties/partyService/" xmlns:sour="http://xmlns.oracle.com/apps/
cdm/foundation/parties/flex/sourceSystemRef/" xmlns:con="http://xmlns.oracle.com/apps/cdm/foundation/
parties/contactPointService/" xmlns:con1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/
contactPoint/" xmlns:org1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/organization/"
 xmlns:par1="http://xmlns.oracle.com/apps/cdm/foundation/parties/flex/partySite/" xmlns:rel="http://
xmlns.oracle.com/apps/cdm/foundation/parties/relationshipService/" xmlns:org2="http://xmlns.oracle.com/
apps/cdm/foundation/parties/flex/orgContact/" xmlns:rel1="http://xmlns.oracle.com/apps/cdm/foundation/
parties/flex/relationship/">
 <soapenv:Header/>
 <soapenv:Body>
 <typ:mergeOrganization>
 <typ:organizationParty>
 <org:PartyId>1006</org:PartyId>
 <org:Relationship>
 <rel:RelationshipRecId>999990007700611</rel:RelationshipRecId>
 <rel:RelationshipId>5041</rel:RelationshipId>
 <rel:PreferredContactFlag>true</rel:PreferredContactFlag>
 </org:Relationship>
 </typ:organizationParty>
 </typ:mergeOrganization>
 </soapenv:Body>
</soapenv:Envelope>
```

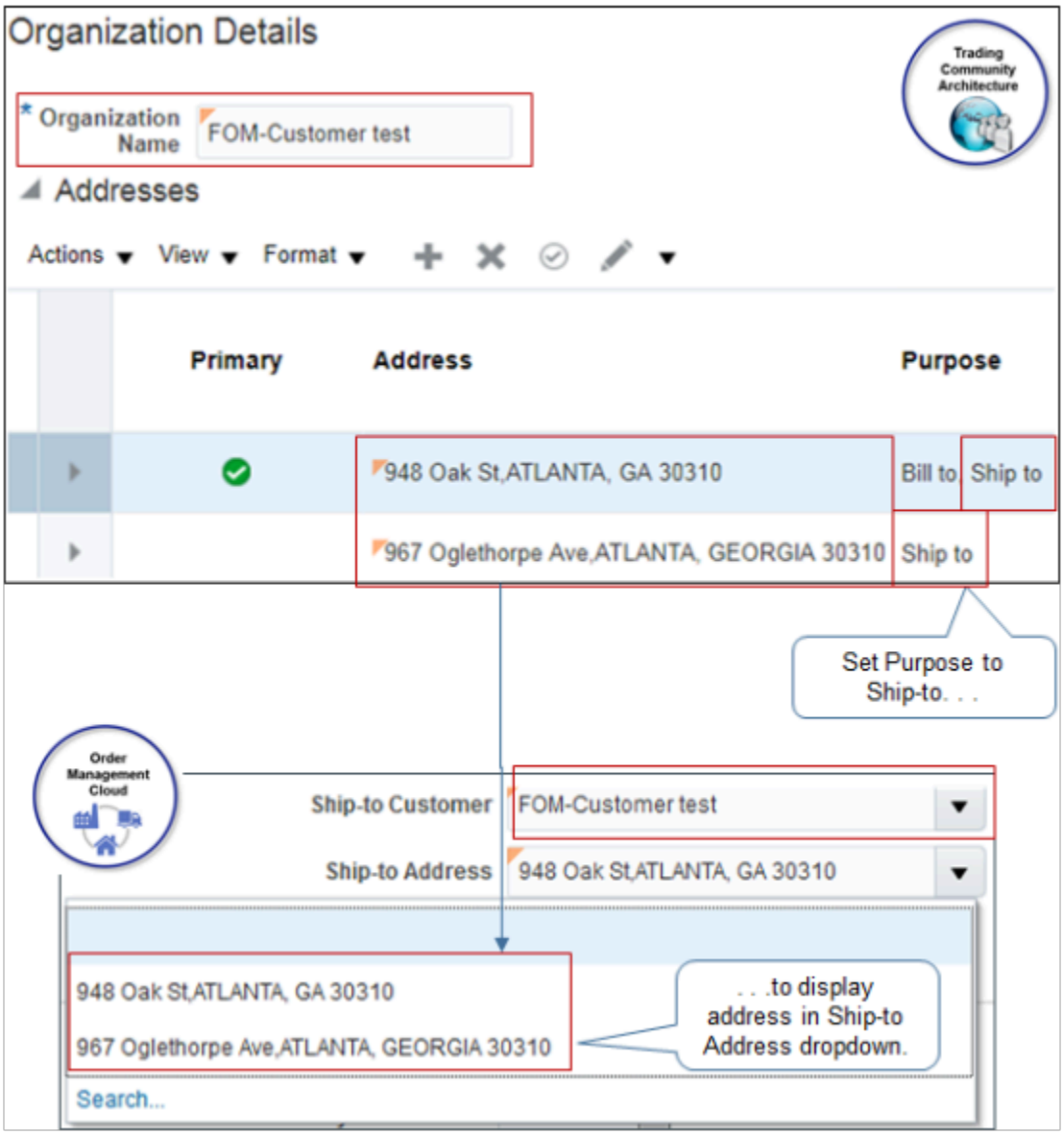
where

- PartyId identifies the party.
- RelationshipRecId identifies the contact relationship.
- RelationshipId identifies the contact. This is the contact that you want to use for the ship-to address and ship-to contact on each sales order.
- You replace the values for PartyId, RelationshipRecId, and RelationshipId with the values that you identified in step 2.
- Set the PreferredContactFlag attribute to true, and Order Management will set the ship-to address and ship-to contact on each sales order according to the contact that the PartyId, RelationshipRecId, and RelationshipId attributes identify.

In this example, you're using the address from the 5041 contact, in the 999990007700611 relationship, in the 1006 party to set the ship-to address and ship-to contact on each sales order.

Control Drop Down for Ship-to Address

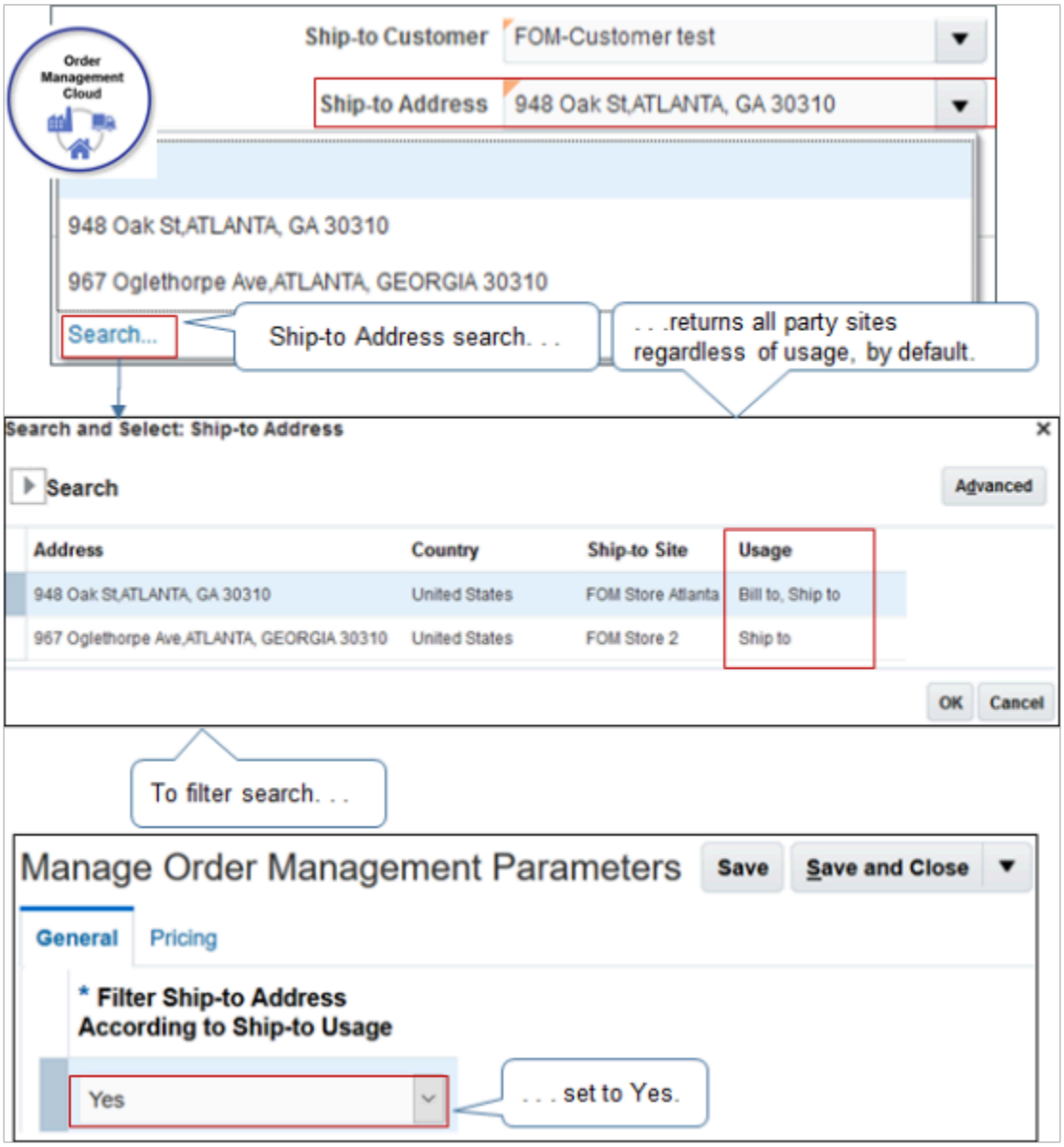
On the Organization Details page, set the Purpose attribute to Ship-to for each address that you must display in the Ship-to Address drop down in the Order Management work area.



Control Search for Ship-to Address

Order Management returns all party sites in the Ship-to Address attribute regardless of usage when you click Search, by default. For example, it returns ship-to usages and bill-to usages.

If you use Oracle Financials, then set the Filter Ship-to Address by Ship-to Usage parameter to Yes. This setting makes sure you can select only ship-to sites and avoid an error from occurring when the import automatically invoices the transaction.

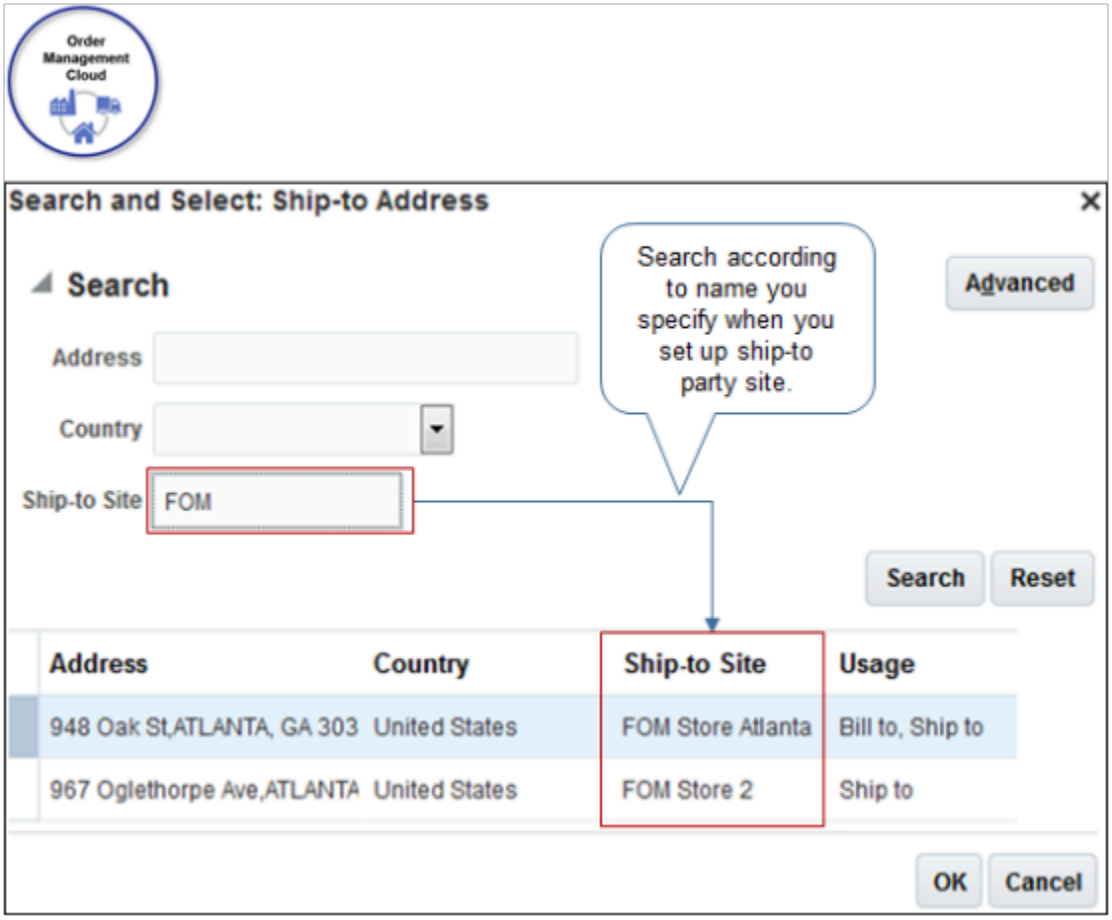


For details, see *Manage Order Management Parameters*.

Search Ship-to Address According to Party Site

You can use an advanced search in the Search and Select dialog to search for ship-to addresses according to the name of the ship-to party site. This dialog displays the name that you specify when you set up your ship-to party site. This feature is useful when your deployment uses a large set of addresses. It allows you to search when you don't know the site's mailing address.

Assume you support a retailer named Computer Service and Rentals who sells at 200 different physical locations. You can search on the text Computer Service to return all locations that begin with the text Computer Service.

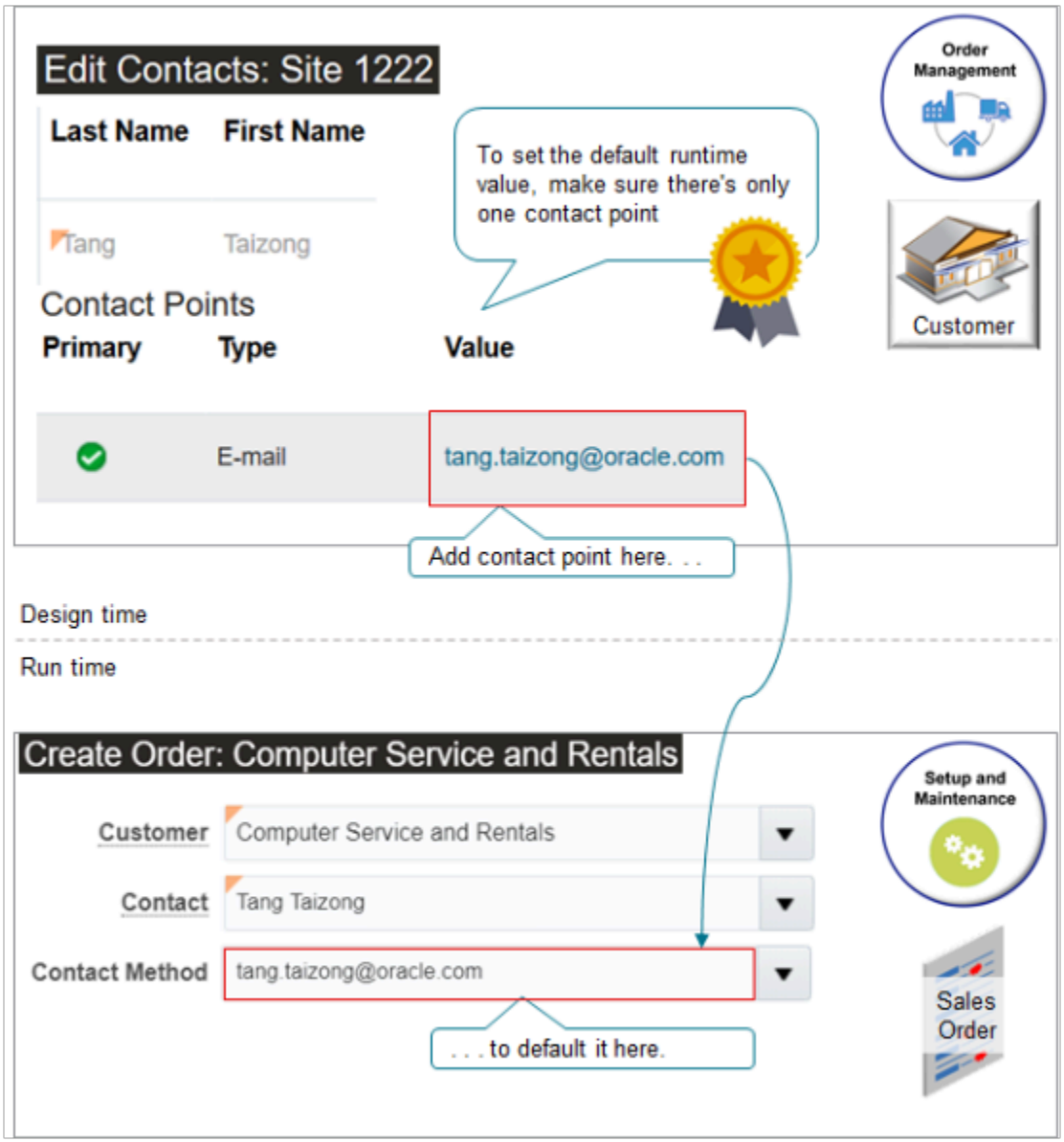


Set Default Value for Contact Method

The Contact Method attribute describes how to contact the person that you set in the Contact attribute on the sales order. Assume Tang Taizong is a contact for your Computer Service and Rentals customer. If Contact contains Tang Taizong, and Contact Method contains tang.taizong@oracle.com, then you can use the tang.taizong@oracle.com email address to contact Taizong.

Assume you need to set Taizong's site to 1800 Satellite Drive, Distribution Center, and you need to set the default value for the Contact Method to Taizong's email address, but you haven't yet specified a contact point for Taizong.

You edit Taizong's contact details in the Setup and Maintenance work area at design time to specify the contact point. Order Management then displays the contact point in the Contact Method on the sales order at run time.



To set the default value that Order Management displays at run time in the Contact Method attribute, make sure you have only one contact point on the Edit Contacts page. For example, if you have an email contact point and a phone contact point, then Order Management won't display any value in the Contact Method attribute, by default.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Manage Customers
    - Task: Manage Customers

2. On the Manage Customers page, search for the value.

Attribute	Value
Organization Name	Computer Service and Rentals

3. Scroll down to the Sites area, then click the **link** in the Site Number column in the row that contains the address that you're looking for.

Attribute	Value
Address	1800 Satellite Drive, Distribution Center,CHATTANOOGA, TN 37401

For this example, assume the Site Number is 1222.

4. On the Edit Site page, click **Communication**, then click **Edit Contact**.
5. On the Edit Contacts page, in the Contact Points area, click **Actions > Create**.
6. In the Create Contact Point dialog set the values, then click **OK**.

Attribute	Value
Contact Point Type	Email
Email Format	Plain Text Email
Email	tang.taizong@oracle.com

Make sure the Contact Points list contains only one row. If it has more than one row, then Order Management won't set a default value for the Contact Method attribute on the sales order.

7. Click **Save and Close**.
8. On the Manage Customers page, click **Done**.

Test Your Set Up

1. Go to the Order Management work area, then create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

2. Verify the attributes values.

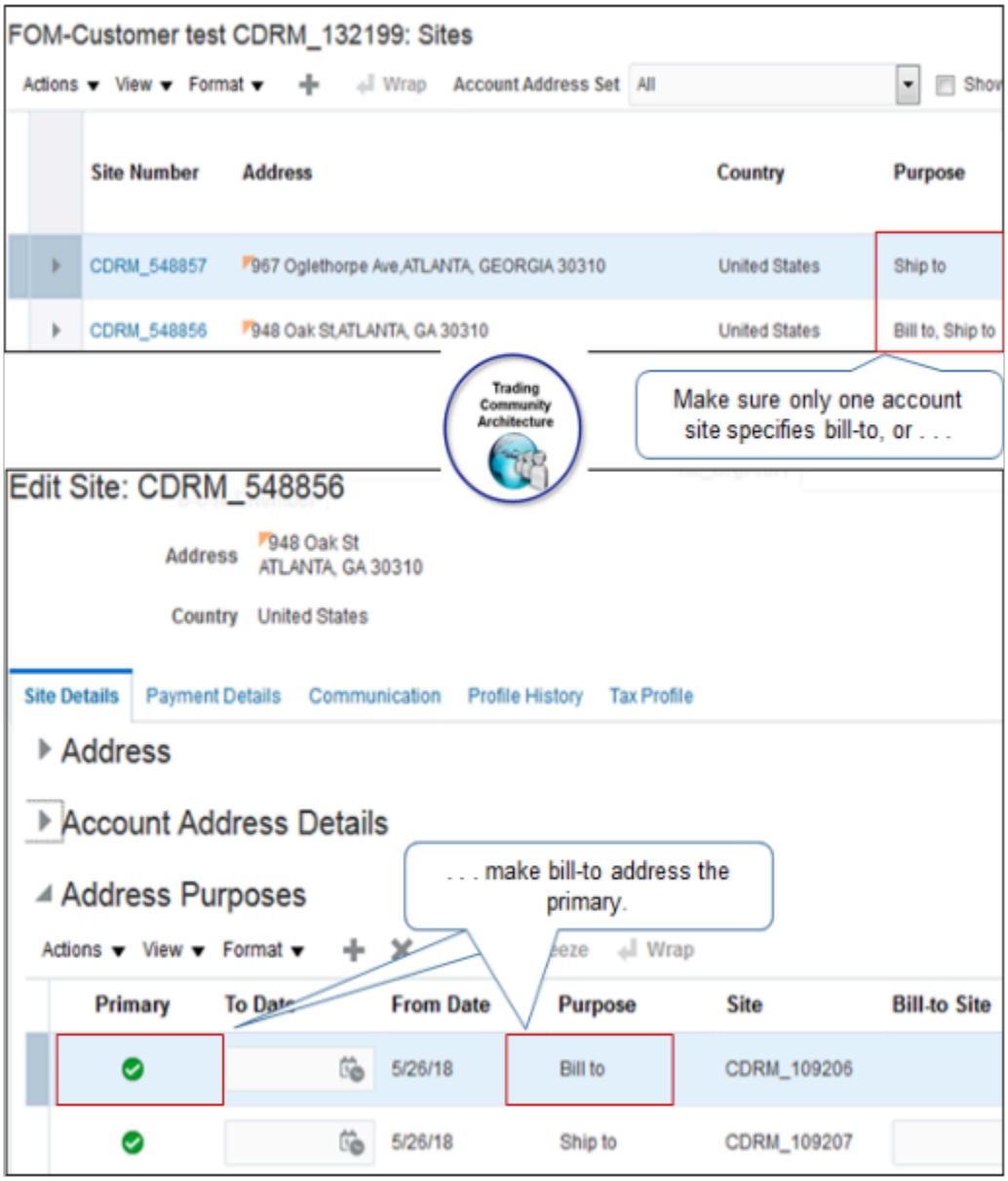
Attribute	Value
Contact	Tang Taizong
Contact Method	tang.taizong@oracle.com

Display Bill-to Addresses On Sales Orders

Control how Order Management displays the bill-to address on a sales order.

Set Default Value for Bill-to Address

Order Management can set the Bill-to Address to the account site that the bill-to account references. You can edit the site to control this behavior.

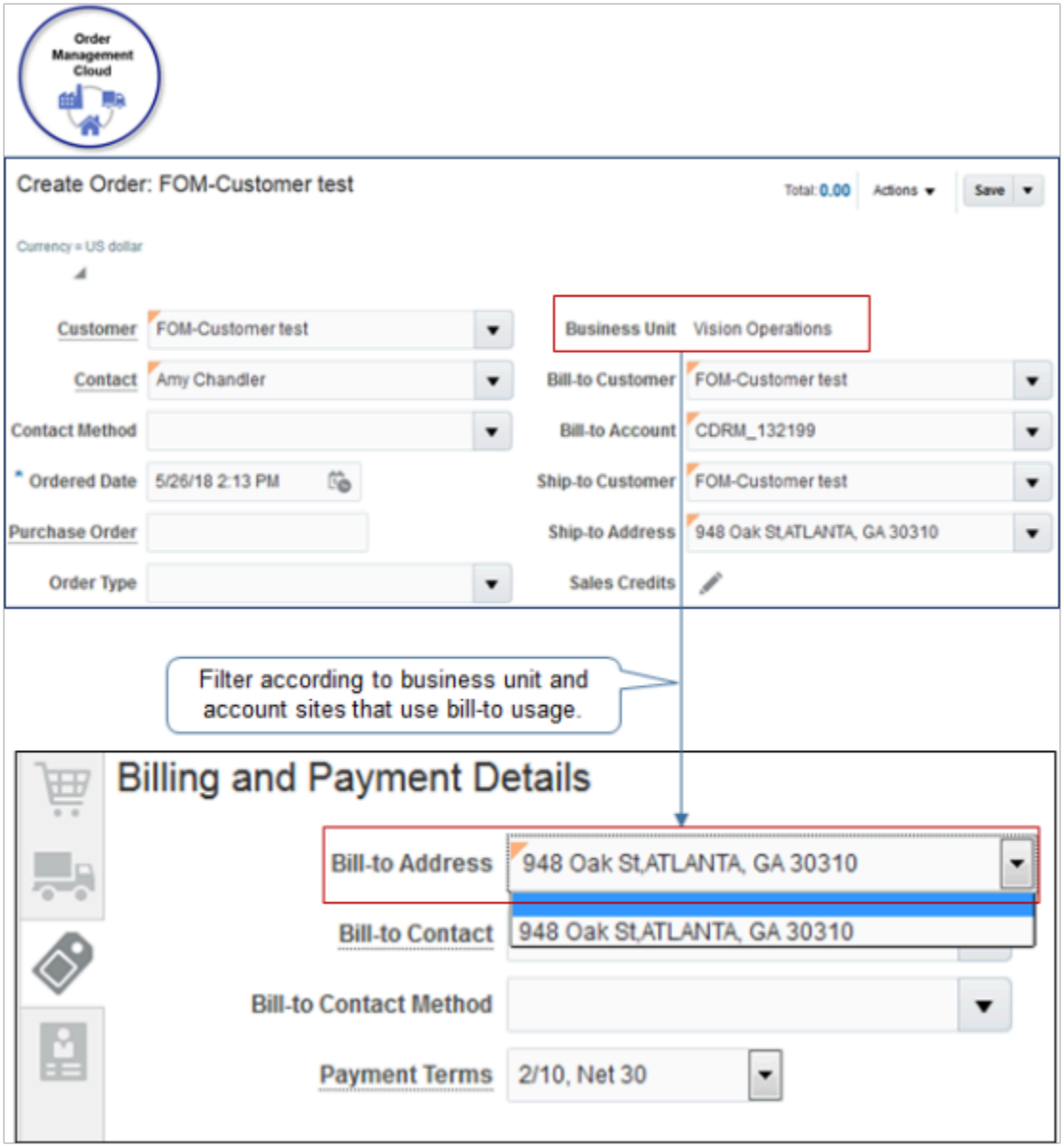


Navigate to the Manage Customers page, then edit the site.

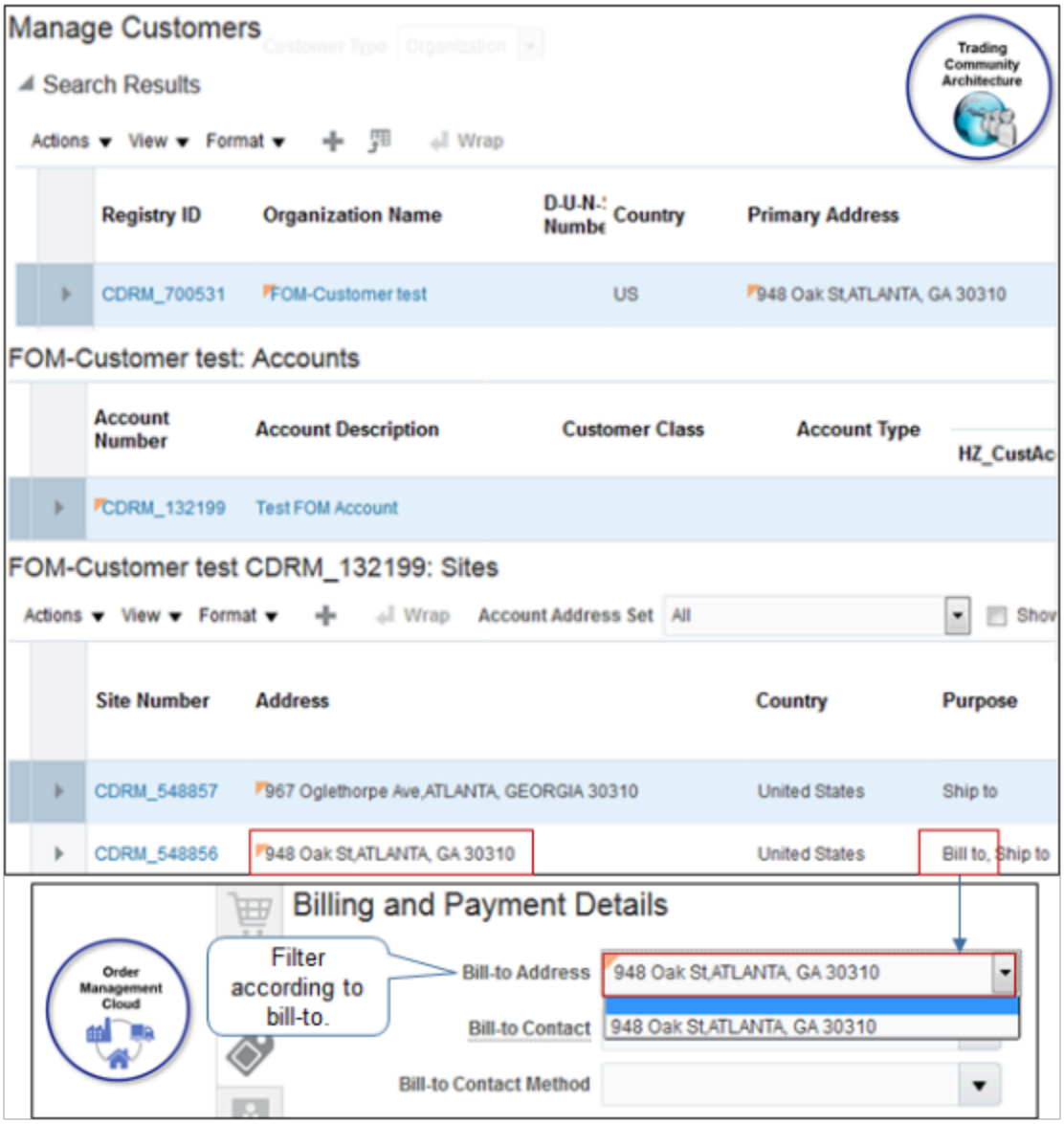
If	Then
Only one site specifies bill-to usage.	Order Management defaults the address to this one site.
More than one site specifies bill-to usage.	Use the Address Purposes area to set the Primary. Order Management will use the primary as the default value for Bill-to Address.

Display Only Bill-to Usages in Bill-to Address

Order Management includes each account site that uses bill-to usage in the drop down for Bill-to Address, by default. Bill-to Address filters account sites according to the business unit that the sales order references. The user can't search this attribute. The user can only select values from the list.



To control the addresses that display, set the Purpose on the account site to bill-to usage when you set up your account.

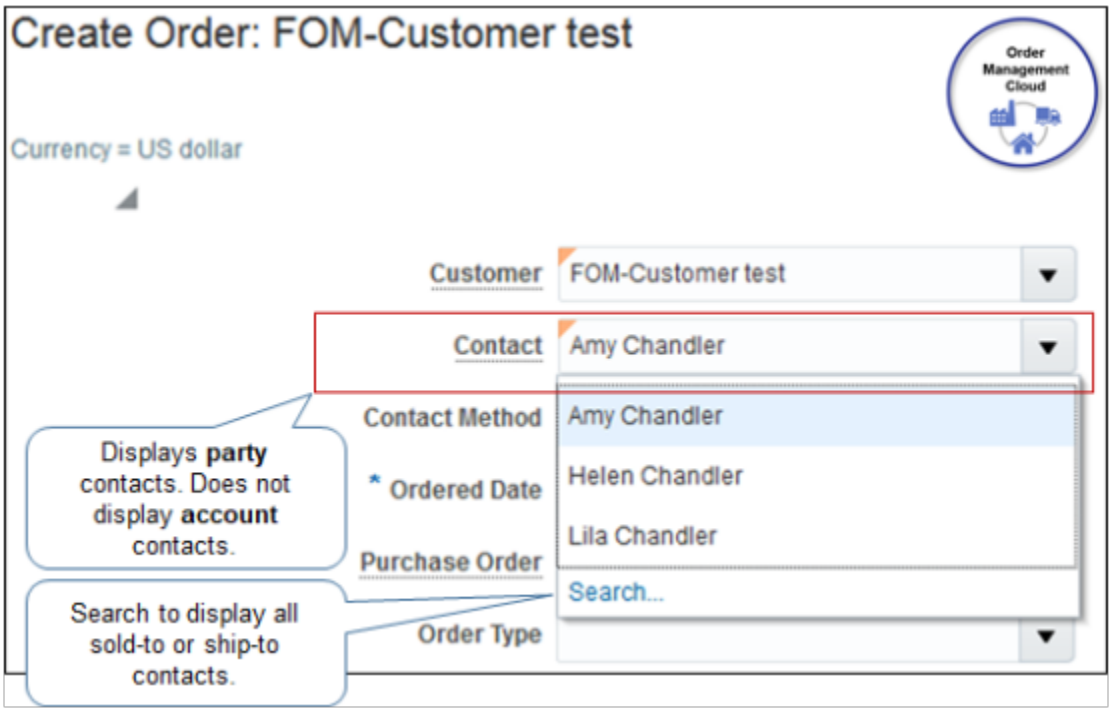


Display Contacts on Sales Orders

Control how Order Management displays contacts on sales orders.

Set Default Value for Contact on Order Header

The Order Entry Specialist can select a contact from the sold-to customer or the ship-to site in attribute Contact on the order header, by default.



Note

- If the Order Entry Specialist clicks Search, then the Contact attribute displays all contacts that reference a sold-to or ship-to contact.
- Note that Contact on the order header displays party contacts. It doesn't display account contacts.
- If you set the primary contact on the account site, then Order Management populates the Contact attribute on the order header to this primary when the user sets the Customer attribute on the order header.
- For details about how to set the default value for the ship-to contact, see *Display Ship-to Addresses On Sales Orders*.

Remove Duplicate Contacts

You might see the same contact more than one time in the list of values that displays when you click the down arrow in the Contact attribute on the order header in the Order Management work area. To remove the duplicate, set the end date for the duplicate contact at the party level. Don't delete it at the site level because this list of values doesn't get its values from the site.

Assume you create a sales order for Computer Service and Rentals, and you see the Diane Cho contact two times in the Contact list of values.

1. Go to the Organizations work area.

   The Organizations work area is part of the Customer Data Management offering. For details about the license and user you need, see *Display Party Details on Sales Orders*. If you don't have the license, then skip this part of the procedure and use REST API to set the end date for the contact.

   For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.

2. On the Organizations page, search for the value.

Attribute	Value
Name	Computer Service and Rentals

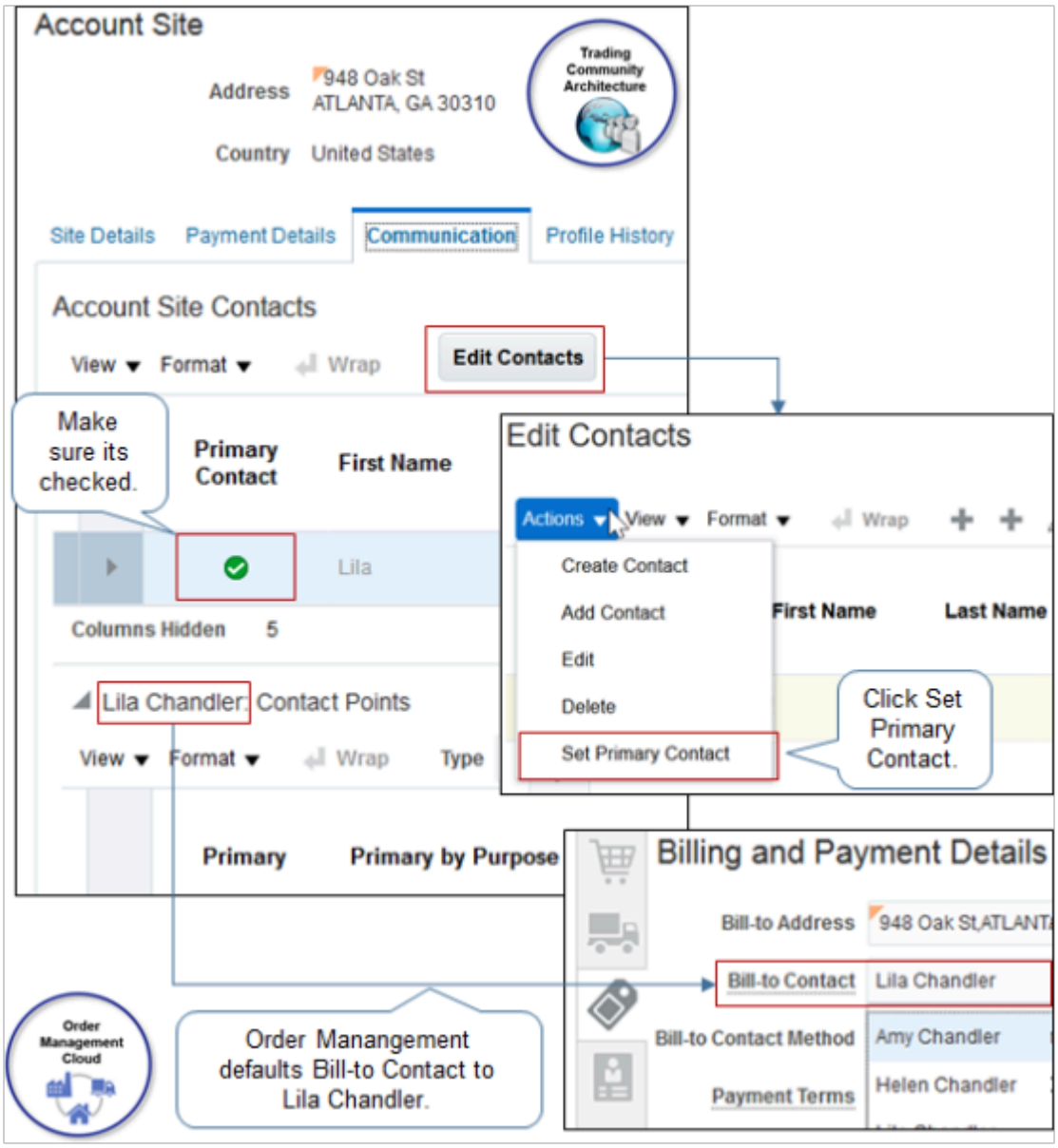
3. In the search results, in the Name column, click **Computer Service and Rentals**.
4. On the Edit Organization page, click **Relationships**.
5. In the Relationships area, click **View > Columns**, then add a check mark to To Date.
6. Look for Diane Cho in the Related Name column. There are probably two rows that have Diane Cho. Set the To Date in one of the rows to yesterday's date, then click **Save and Close**.
7. Collect data for the Organizations entity.

   For details, see *Collect Planning Data for Order Management*.

Set Primary Contact on Account Site

1. On the Manage Customers page, search for the party you must modify.
2. In the Sites area, click the **Site Number** for an account site.
3. On the Edit Site page, click **Communication**.
4. In the Account Site Contacts area, click **Edit Contacts**.

5. On the Edit Contacts page, click **Actions > Set Primary Contact**.



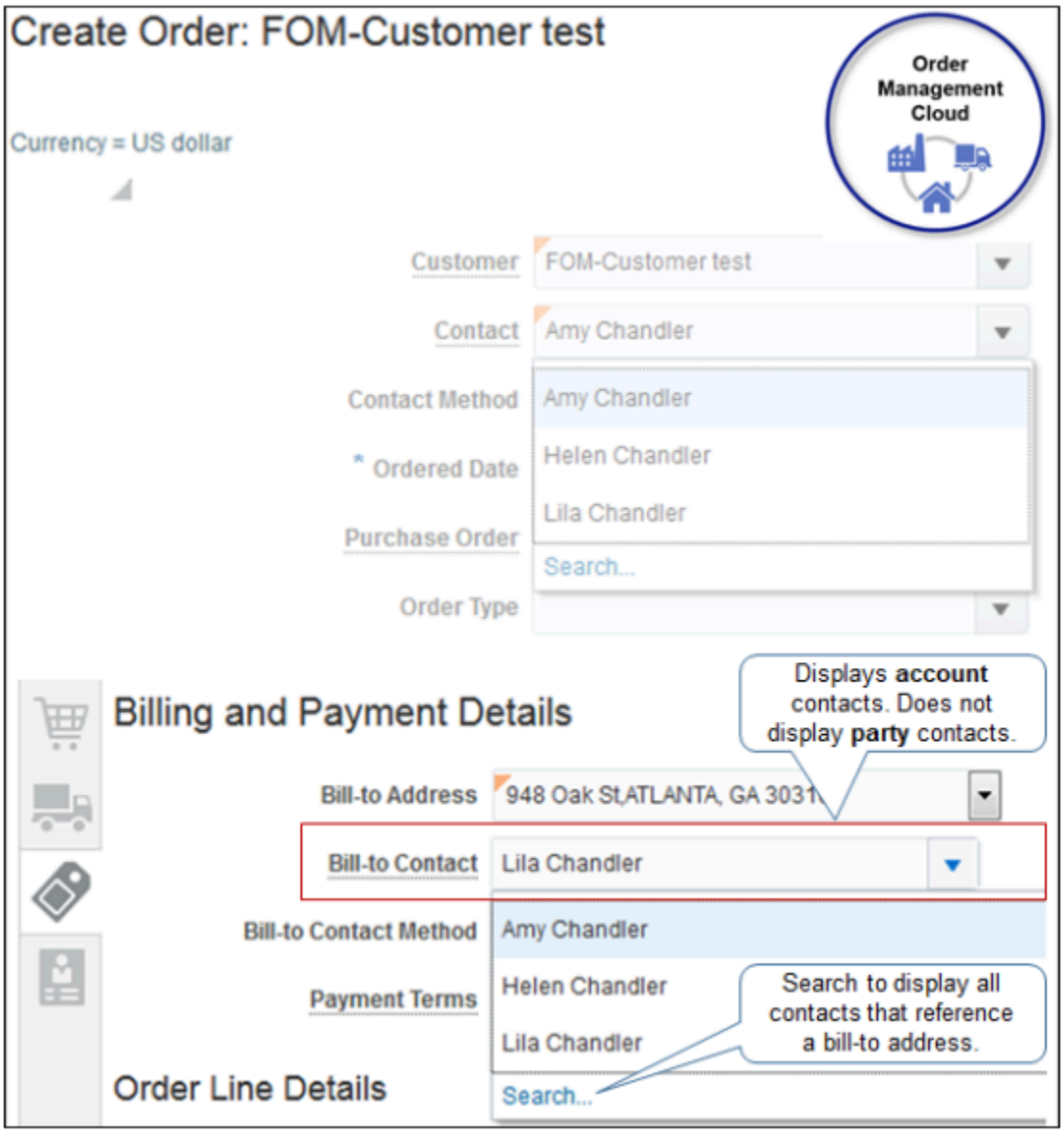
6. Click **Save and Close**.

Set Default Value for Contact in Billing and Payment Details

If you specify the primary account during set up, then Order Management populates the Contact attribute in the Billing and Payment Details area to the primary contact for this account when the user sets Customer on the order header.

If the user clicks Search, then the drop down for Contact displays each contact that references a bill-to address.

Note that Contact in the Billing and Payment Details area displays account contacts. It doesn't display party contacts.



Related Topics

- Overview of Collecting Promising Data for Order Management
- Display Party Details on Sales Orders

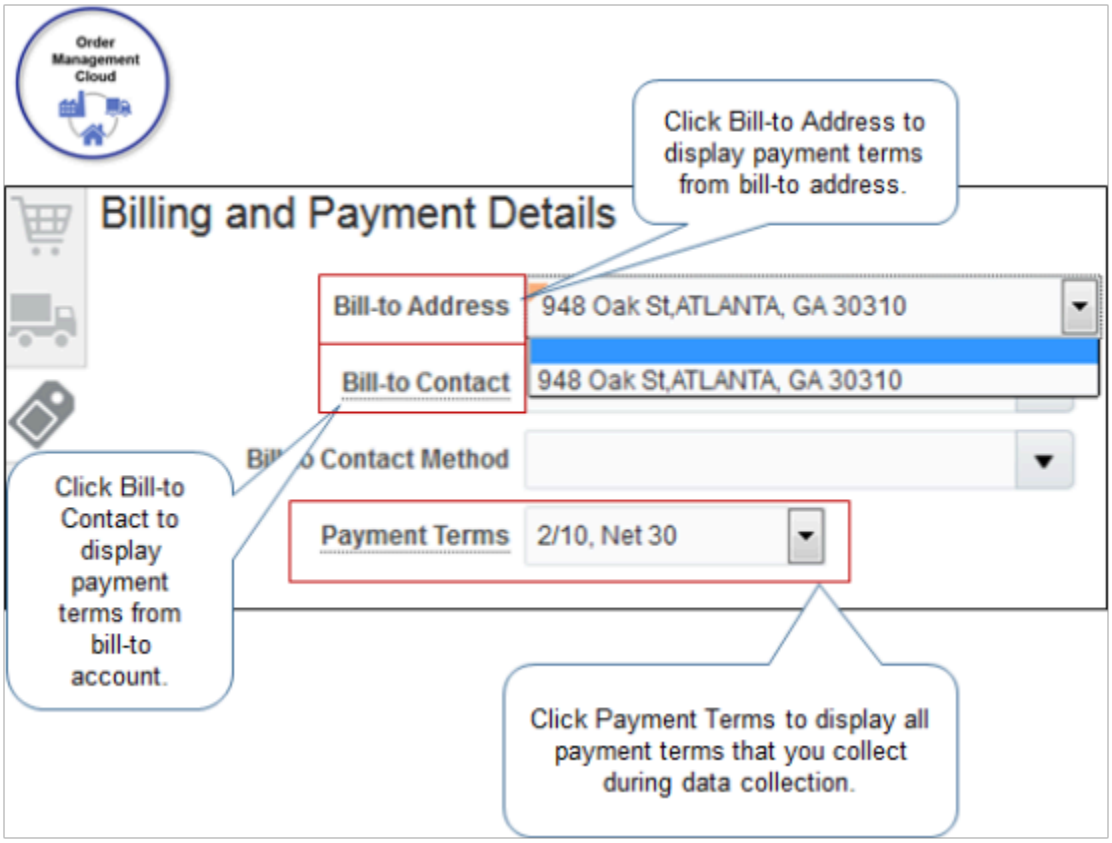
Display Payment Terms on Sales Orders

Control how Order Management displays payment terms on sales orders.

Set Default Value for Payment Term

The payment term is a financial attribute on the account and account site. Order Management populates the Payment Term attribute when the Order Entry Specialist sets one of these attributes in the Billing and Payment Details area.

Attribute	Description
From Bill-to Address	Get and display the payment term that you define on the bill-to address for the bill-to site during set up.
From Bill-to Contact	Get and display the payment term that you define on the bill-to account during set up.



Note

- If you don't specify the payment term on the bill-to address or the bill-to account in Trading Community Architecture, then Order Management doesn't set any default value for the payment term, and the Order Entry Specialist must manually set it.
- The Order Entry Specialist can click Payment Terms to view all values that you collect during data collection, including values from Oracle Applications, Oracle Advanced Supply Chain Planning, your channel systems, and so on. For details about data collection, see *How Order-to-Cash Works with Order Capture Systems*.

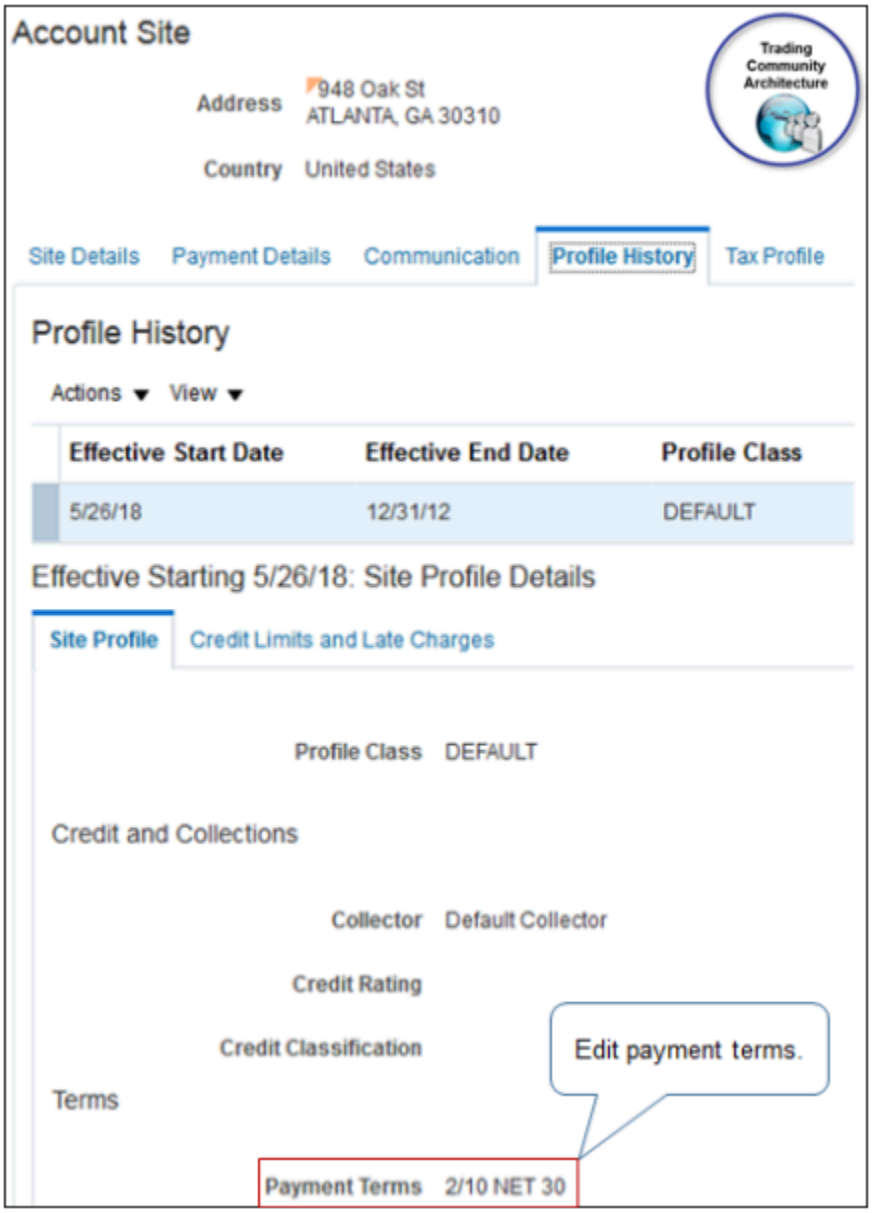
Set Default Value to Payment Term from Account Site

Specify to use the payment term from the account site as the default value.

1. On the Manage Customers page, search for the party you must modify.
2. In the Sites area, click the **Site Number** for an account site.
3. On the Edit Site page, click **Profile History**.
4. In the Profile History area, click **Actions > Correct Record**.
5. On the Site Profile tab, in the Terms area, select a value for the Payment Terms attribute.

6. Click **Save and Close**.

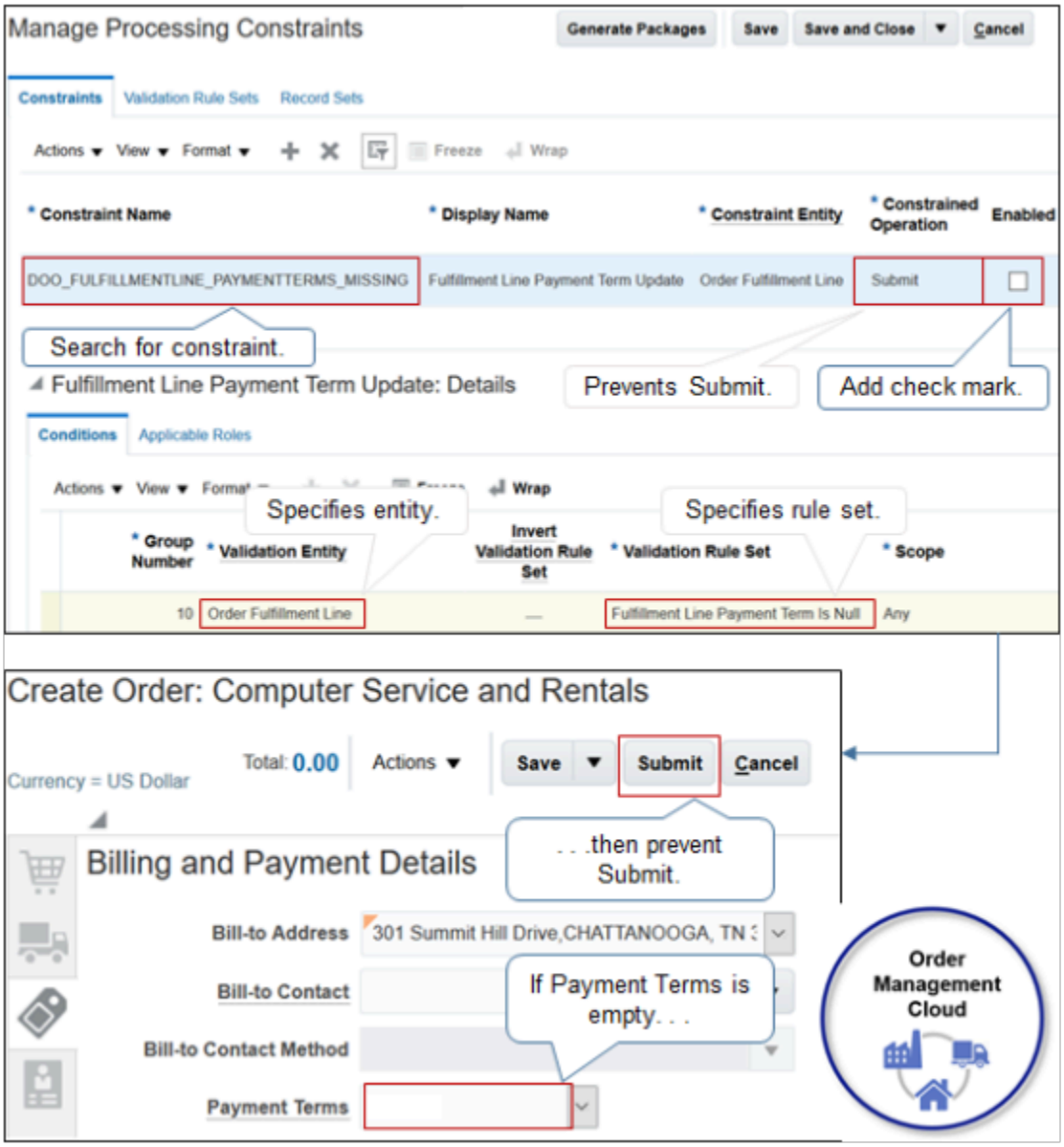
For example:



Make Sure User Sets Payment Term

If you use Oracle Financials, then Payment Term must contain a value. To meet this requirement, you can enable the Fulfillment Line Payment Term Update predefined constraint. If you enable it, and if Payment Term is empty, then the constraint prevents the Submit action.

This requirement helps to avoid problems when Financials invoices the fulfillment line. This constraint comes predefined as disabled. You can enable it.



Set the values on the Managing Processing Constraints page. For details, see *Manage Processing Constraints*.

Attribute	Description
Constraint Name	Search for DOO_FULFILLMENTLINE_PAYMENTTERMS_MISSING.
Enabled	Add a check mark.

Here's the logic that the constraint.

Attribute	Description
Constraint Name	DOO_FULFILLMENTLINE_PAYMENTTERMS_MISSING.
Constraint Entity	Prevent Order Management from performing the operation that Constrained Operation specifies on Order Fulfillment Line.
Constrained Operation	Submit.
Validation Entity	Specifies to perform the validation on Order Fulfillment Line.
Validation Rule Set	Specifies to use the Fulfillment Line Payment Term Is Null rule set to determine whether the Payment Term attribute on the fulfillment line contains a value.

Set Default Values for Other Attributes on Sales Orders

Here are some techniques you can consider when you set the default value for various attributes on sales orders.

- Set up a pretransformation rule. For example, set the default value for the customer contact according to business unit and customer.

  ```
  If business unit is y, and if customer is x, then set contact to z.
  ```

- Order Management runs pretransformation rules each time the user modifies the Customer attribute. This configuration makes sure Order Management updates default values according to rules you define. For details, see *Overview of Using Business Rules With Order Management*.
- Set up an order management extension. For example, get data from a customer entity on the sales order, and then use this data as the default value.

  ```
  If customer class is x, then set order type to z.
  If descriptive flexfield on customer contains a, then set attribute b to c.
  ```

Import Customer Data for Your Sales Orders

Import customer data, then display it on your sales order.

Set Default Values During Order Import

Order import doesn't set the default value for an address, contact, or payment term. Your import payload or the order import template must specify them.

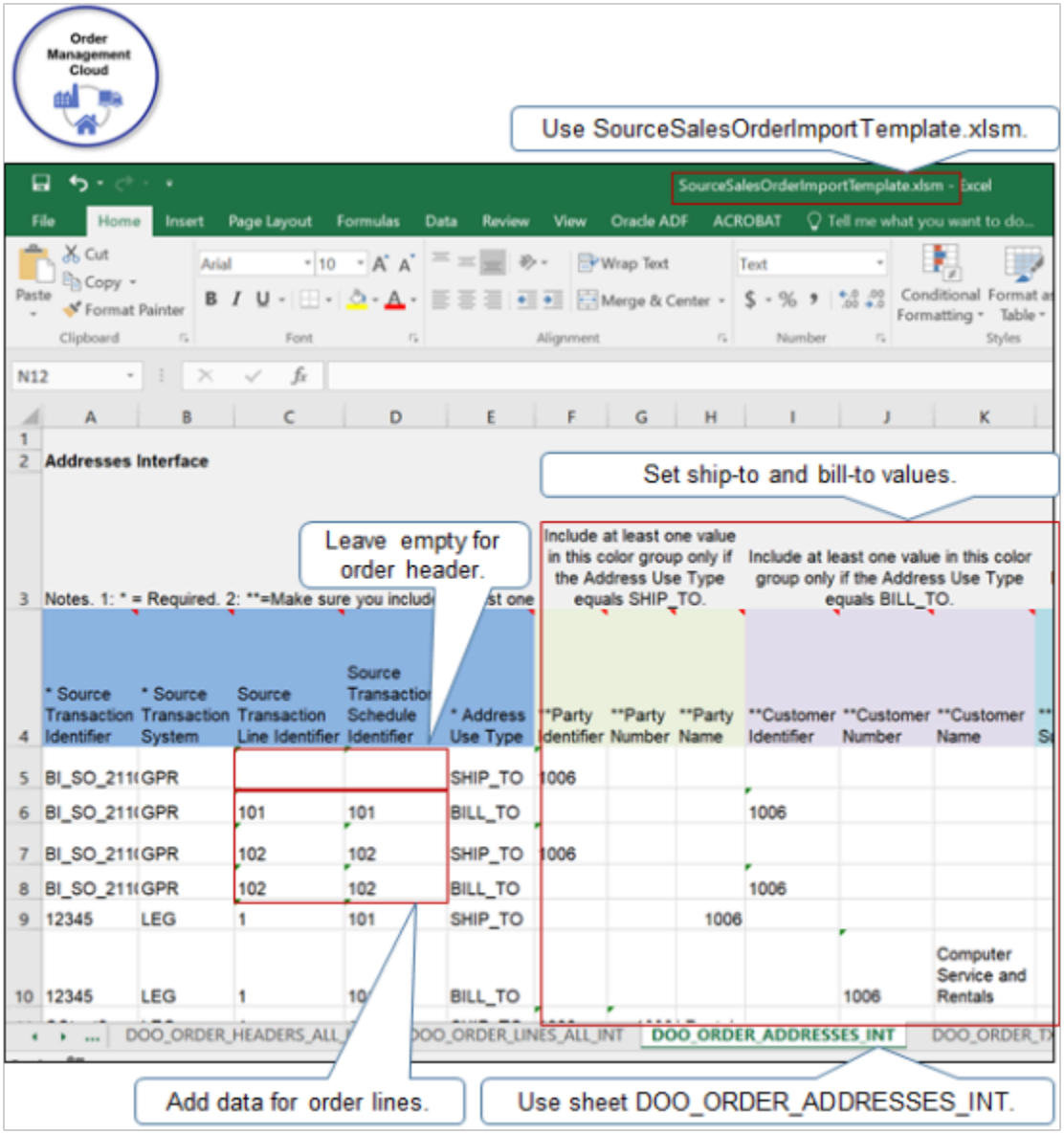
If you use order import template SourceSalesOrderImportTemplate.xlsm, then do these tasks.

- Use the DOO_ORDER_ADDRESSES_INT worksheet to specify default values for the order header and order lines.

- Set the values depending on whether your import creates or updates.

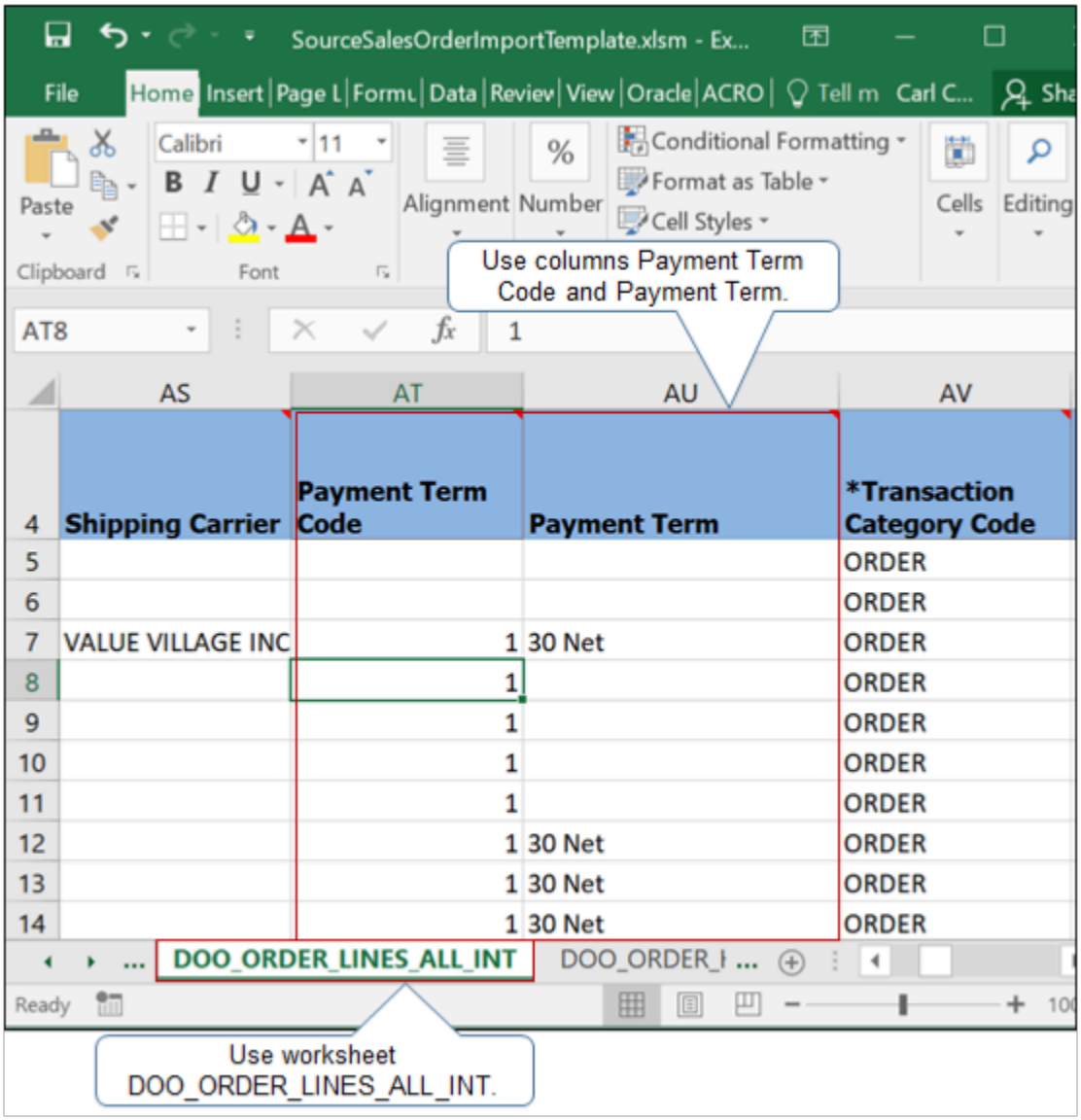
Create or Update	Description
Create a new sales order	Send addresses on the order header, and the import will cascade order header data to the new order line.
Update a sales order	To set the order header, leave these columns empty. o Source Transaction Line Identifier o Source Transaction Schedule Identifier To set order lines, make sure these columns contain values. An update must contain data for each order line.

- Use the instructions in the worksheet to set ship-to and bill-to values.



For details, see *Import Orders Into Order Management*.

Import Payment Terms



Note

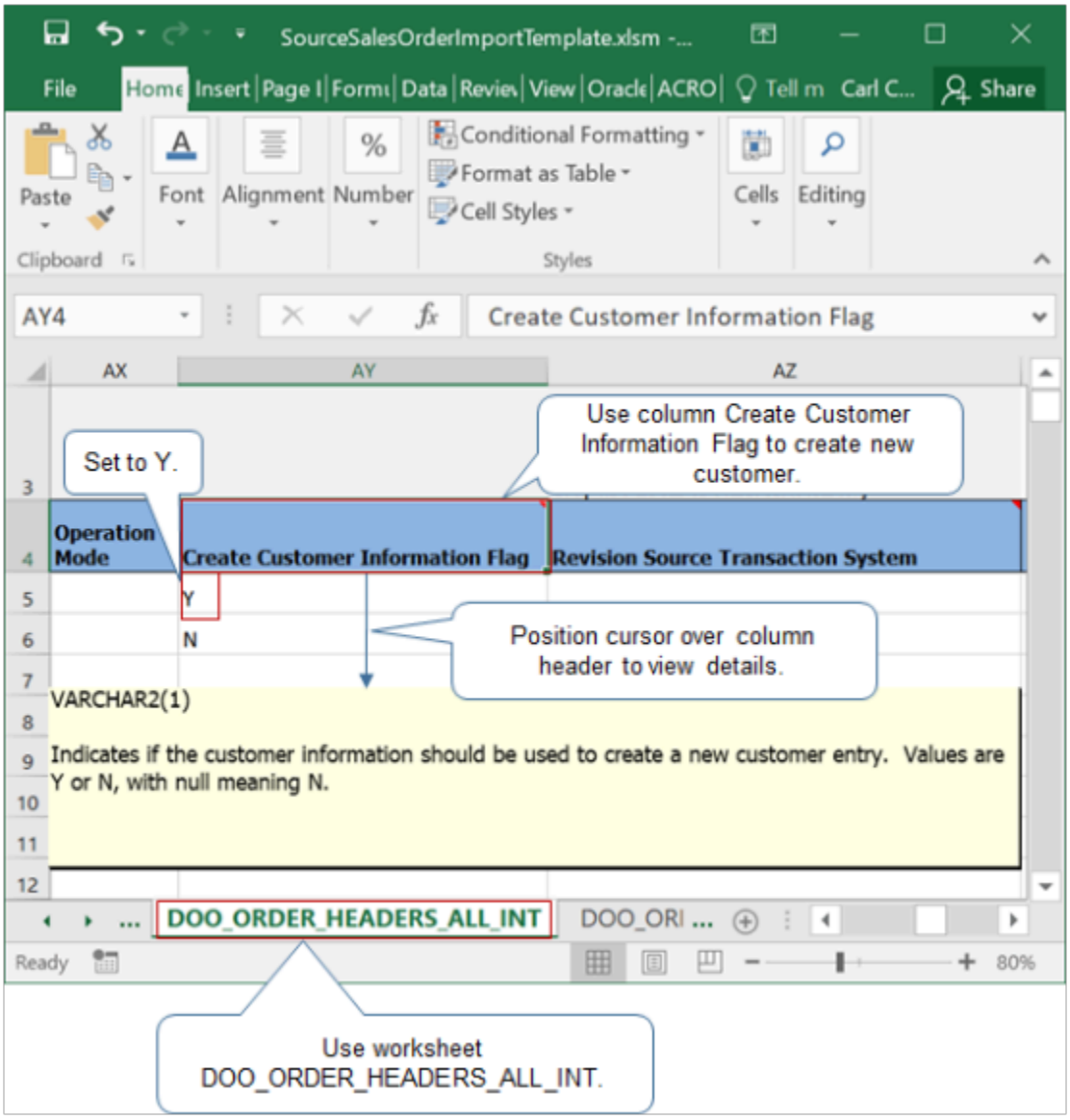
- Use the DOO_ORDER_LINES_ALL_INT worksheet of the order import template to import payment terms.
- Use the Payment Term Code or the Payment Term column.
- If you use Oracle Financials, then make sure you send payment terms for each order line in each imported order that you must invoice.
- Order import doesn't support payment terms on the order header so you can't cascade payment terms from the order header to order lines.

Create New Customers

Order Management doesn't provide an administrative interface you can use to create a new customer, but you can use the order import template instead.

1. Open the DOO_ORDER_HEADERS_ALL_INT worksheet.
2. Set the `Create Customer Information Flag` column to Y.

3. Position the cursor over the column header to view details.
4. Fill in all customer data on these worksheets.
    - DOO_ORDER_HEADERS_ALL_INT
    - DOO_ORDER_ADDRESSES_INT



Other Ways to Create Customer Data

Use the customer import services in Trading Community Architecture to bring customer data into Oracle from your channel system or legacy system.

Copy Setups

Copy Setups Between Instances of Order Management

Copy your setup from one instance of Order Management into another instance during the lifecycle of your Order Management deployment. For example, migrate your setup from a test environment to a production environment.

- Use the Manage Configuration Packages page in the Setup and Maintenance work area to export and import a configuration package.
- Learn about the list of business rules that the copy migrates. For details, see *Migrate Business Rules in Order Management*.
- Learn about the approval rules you can migrate. For details, see *Migrate Approval Rules Between Instances of Order Management*.
- You can use WebLogic Scripting Tool (`WLST`) commands to move order orchestration rules. Order Management stores order orchestration rules in the Metadata Services (MDS) Repository. You can also use data collection and interface tables to import your setup and your transaction data. For details, go to *Implementing Common Features for Oracle SCM*, then see the chapter that describes import and export.
- The migration adds hold codes that you set up in the source instance to hold codes that exist in the target instance. If the same hold codes exist in the source and target, then the holds in the source replace the holds in the target.
- The migration doesn't migrate holds that apply a credit check hold or release a credit check hold.
- The migration migrates only your set ups. It doesn't migrate transactional data, such as sales orders.
- Don't modify the orchestration process name, task name in an orchestration process step, or the status rule set name in either environment. Modifying the name might prevent Order Management from updating references to other data in the orchestration process. For details, see:
  - *Guidelines for Setting Up Orchestration Processes*
  - *Guidelines for Setting Up Orchestration Process Steps*
  - *Fulfillment Line Status*
- If you created an order management extension, then prepare it before you migrate. For details, see *Overview of Creating Order Management Extensions*.
- If you migrate from a production environment to a test environment, and if your migration includes an orchestration process that's currently in progress, then the sales order that references the process will become stuck. It isn't possible to recover the stuck order.
- The migration migrates only your set ups. It doesn't migrate transactional data, such as sales orders.

Summary of the Steps

1. Export your setup.
2. Import your setup.
3. Deploy flexfields.

Export Your Setup

Export your setup from the source instance of Order Management.

1. Make sure the source instance and the target instance are at the same release level.
2. Sign into the source instance of Order Management with administrative privileges.

3. Go to the Setup and Maintenance work area.
4. On the Setup page, click **Tasks > Manage Configuration Packages**.
5. On the Manage Configuration Packages page, click **Actions > Create**.
6. On the Enter Basic Information page, set the values, then click **Next**.

Attribute	Value
Name	Select the name of the implementation project that defines the source instance.
Export	Setup task list and setup data

7. On the Select Objects for Export page, accept default values, then click **Next**.

   The Select Objects for Export page comes predefined to select the objects that the export needs to support most instances of Order Management. For details, see *Copy Setups Between Instances of Order Management*.
8. On the Schedule and Notifications page, accept the default value, click **Submit**, then, in the Warning dialog, click **Yes**.

Attribute	Value
Run	As soon as possible

9. On the Manage Configuration Packages page, examine results in the elements, then sign out.

Element	Description
Export Setup Data button	Export a configuration package after you create it. The export identifies setup data according to the export definition and adds it to the configuration package. Export a configuration package more than one time. Each export creates a different configuration package version that you can manage individually.
Download Latest Version	Download the most recent version of the configuration page.
Export and Import Processes area	Get details of the export or import for each configuration package.
Status	Examine the process status for each implementation project. Click the status to get details about each step of the export or import process.
Download	Download a version so you can use it during an export or import.

Element	Description
Setup Data Report	View or download a report that contains the setup data exported to the configuration package, including the business objects processed and details about errors that occurred.

Import Your Setup

Import your setup into the target instance of Order Management.

1. Sign into the target instance of Order Management with administrative privileges.
2. Go to the Setup and Maintenance work area.
3. On the Setup page, click **Tasks > Manage Configuration Packages**.
4. On the Manage Configuration Packages page, in the search results, click the **row** that contains the configuration package you must update.
5. In the Export and Import Processes area, click **Import Setup Data**.
6. On the Enter Basic Information page, accept default values, then click **Next**.
7. On the Select Pauses for External Import page, click **Submit**.

   The import process.

   - Adds setup data that doesn't already exist in the target configuration package. It adds setup data from the source configuration package into the target configuration package.
   - Updates setup data that already exists in the target package with modifications from the source package.
   - Doesn't delete existing setup.
   - Doesn't modify setup that exists in the target instance but not in the source instance.
8. Restart the server that hosts the target instance.

Deploy Flexfields

You must deploy each flexfield that you import.

Get details about:

- How to handle objects that aren't flexfields after the import, go to *Implementing Common Features for Oracle SCM*, then see the chapter that describes import and export.
- Flexfields, see *Set Up Extensible Flexfields in Order Management*.

Deploy flexfields.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Extensible Flexfields
2. On the Manage Order Extensible Flexfields page, identify each flexfield where the deployment status isn't Deployed, then deploy it.
3. Repeat steps 1 and 2 on each of these pages, as necessary.
   - Manage Item Revision Descriptive Flexfields

- Manage Units of Measure Descriptive Flexfields
- Manage Item Descriptive Flexfields
- Manage Item Revision Descriptive Flexfields
- Manage Item Relationship Descriptive Flexfields
- Manage Trading Partner Item Descriptive Flexfields
- Manage Catalog Descriptive Flexfields
- Manage Category Descriptive Flexfields
- Manage Source System Descriptive Flexfields
- Manage Order Descriptive Flexfields
- Manage Pricing Descriptive Flexfields

4. Sign into Oracle Enterprise Manager.

   For details about Oracle Enterprise Manager, see https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.
5. In the navigation tree, select **SOA Infrastructure**, then click **UpdateSOAMDS**.
6. On the UpdateSOAMDS page, set the values.

Attribute	Description
Operation	Set to updateDuring.
Value This Value attribute resides in a row in the **`Input Arguments`** area, in Tree View. The value for Name in the row is **`*payload`**, and the Type is duration.	Specify the time frame that has elapsed since the last time you deployed the flexfields. Use this format. **`PXMYD`** where - **X**. Number of months that have elapsed. - **Y**. Number of days that have elapsed. For example, if one month and one day have elapsed since the last time you deployed flexfields, then use this format. **`P1M1D`** You can specify only the month, only the days, or months and days.

7. Run the UpdateSOAMDS composite to synchronize SOA (Service Oriented Architecture) with flexfields in Order Management.
8. Go to the Scheduled Processes work area.
9. On the Scheduled Processes page, click **Schedule New Process**, then run the *Publish Extensible Flexfield Attributes* scheduled process.

Related Topics

- More Set Up Details for Extensible Flexfields
- Reference for Copying Setup Data Between Instances of Order Management
- Export Setup Data Using Implementation Project
- Import Setup Data Using Implementation Project

Reference for Copying Setup Data Between Instances of Order Management

Get the details you need when you copy setup data between instances of Order Management.

Collect Reference Data and Transaction Data for Order Orchestration and Order Promising

The Order Orchestration and Planning Data Repository includes the data that Order Management needs to orchestrate sales order fulfillment and that Global Order Promising needs to promise sales orders. Use data from your source system or from an Oracle Application source system to populate the repository.

Source Systems

Specify the source system and maintain data collection parameters for the source system. Collect data from applications.

Application	Description
Order Management	Collect data. • Oracle Applications data, including data that's not specific to an item or customer. Collect reference entities so you can validate and cross-reference them. • Order capture codes. • Accounts receivable codes. • Accounting terms and currencies. • Miscellaneous data.
Global Order Promising	Collect data. • Existing supply, including on-hand, purchase orders, and work orders • Capacity, including supplier capacity and resource capacity • Related demand, including work order demand and resource requirements for work orders • Planned supply, including buy and make orders that you plan • Reference data, including calendars, transit items, and routing

You must refresh Global Order Promising after you collect data to make sure it has the data that you most recently collected.

Setup and Maintenance

Use pages in the Setup and Maintenance work area to manage source systems and collect data.

Page	Description
Manage Orchestration Source Systems	Manage your source system and the parameters that you use to collect data.
Manage Orchestration Data Collection Processes	Manage reference data from other source systems. Specify whether to enable cross-referencing for various entities, such as Currencies, Units of Measure, and so on.
Review Collected Order Reference Data	Examine the data that you collected from other source systems for order orchestration.
Manage Planning Source Systems	Manage planning source systems and collection parameters so you can use it to collect data.
Manage Planning Data Collection Processes	Manage planning data from a source system.
Review Planning Collected Data	Examine the data that you collected from other source systems.
Monitor Planning Data Collection Process	Monitor the collection for planning data that's currently running or that collections already finished.

For details, see:

- *Implementing Order Management*
- *Implementing Supply Chain Planning*
- *Using Order Promising*

Copy Other Setup Data

Copy your setup data from a source instance of Order Management to a target instance of Order Management.

Setup	Description
Item Setup	• Export and import a configuration package to copy setup data for your item, such as organization, class, catalog, category, lifecycle phase, and so on. • Use interface tables so you can copy items, item structures, and item relationships. • Use the Import Items page and the Monitor Item Imports page to manage imports for your item.
Party Setup	• Export, then import a configuration package so you can copy setup data for your parties, such as relationship type, classification, geography lookup, source system for your trading community, and so on. • Use interface tables to copy party data. • Use the Import Persons and Organizations page to manage your imports that include party data.
Security Setup	Use commands in the Lightweight Directory Access Protocol (LDAP) to copy job roles. For details, see *Implementing Security*.

Get More Documentation

See these books to get details about copying your setup data.

- *Implementing Common Features for Oracle SCM*
- *Configuring and Extending Applications*
- *Using Order Promising*

Get details about using Product Information Management.

- *Using Product Master Data Management*
- *Importing Data into PIM Hub, Generic Examples of Steps (Doc ID 1279983.1)*
- *Item Import, Example SQL Code Scripts (Doc ID 1393229.1)*

Get other details.

- *Example SQL To Import Items into Product Information Management Using Open Interface Tables (Doc ID 1299158.1)*
- *Oracle Trading Community Bulk Import (Doc ID 1383922.1)*

Business Objects That You Can Export and Import

Specify business objects when you copy setup data for Order Management between configuration packages.

Supply Chain Management

Export and import business objects in Oracle Supply Chain Management.

Application	Page	Description	Business Object
Advanced Supply Chain Planning	Manage Enterprise Scheduler Jobs for Advanced Planning Applications	Manage jobs for Oracle Enterprise Scheduler and the sources that you use to provide values to a planning application, such as Global Order Promising.	Enterprise Scheduler Job List of Values Source
Advanced Supply Chain Planning	Manage Global Order Promising Profile Options	Manage profile options and values to control Global Order Promising, such as the default values to use on a page or timeout parameters.	Application Profile Value
Order Management	Generate Constraint Package	Create a dynamic package that activates new validation rule sets for processing constraints.	Orchestration Change Constraint
Order Management	Manage Constraint Entities	Manage the entities to use when you apply a constraint.	Orchestration Change Constraints Object
Order Management	Manage Enterprise Scheduler Jobs for Order Orchestration	Manage jobs for Oracle Enterprise Scheduler and the sources that these jobs use for lists of values in Order Management.	Enterprise Scheduler Job List of Values Source

Application	Page	Description	Business Object
Order Management	Manage Hold Codes	Manage the abbreviations that Order Management uses to hold processing for a sales order or order line.	Orchestration Hold Code
Order Management	Manage Jeopardy Priorities	Manage score codes that indicate the severity of the delay of a task.	Orchestration Jeopardy Priority
Order Management	Manage Jeopardy Thresholds	Manage jeopardy threshold definitions that determine the degree of action to take when orchestration highlights potential or actual fulfillment issues because of delays.	Orchestration Jeopardy Threshold
Order Management	Manage Orchestration Attachment Categories	Manage attachment categories that group messages together.	Application Attachment Category
Order Management	Manage Orchestration Descriptive Flexfields	Manage validation and display properties of descriptive flexfields. Use descriptive flexfields to add modified attributes to entities.	Application Descriptive Flexfield
Order Management	Manage Orchestration Extensible Flexfields	Manage properties of extensible flexfields to extend the attributes of a transactional entity. Use extensible flexfields to collect more than one context in the same flexfield.	Application Extensible Flexfield
Order Management	Manage Order Lookups	Manage lookup values for attributes, such as the return reason code or activity type.	Application Standard Lookup
Order Management	Manage Orchestration Process Definitions	Manage the definitions that determine how Order Management orchestrates fulfillment.	Orchestration Process
Order Management	Manage Orchestration Profiles	Manage profile definitions to specify how orchestration processes data.	Application Profile Value
Order Management	Manage Order Attributes That Identify Change	Manage the attributes that are necessary to identify changes in each sales order.	Orchestration Change Attribute
Order Management	Manage Processing Constraints	Manage the rules that control attempted changes to order orchestration.	Orchestration Change Constraint
Order Management	Manage Status Values	Manage status values for tasks.	Orchestration Status Code

Application	Page	Description	Business Object
Order Management	Manage Task Status Conditions	Manage the process status conditions that indicate when the process uses a status. Specify these conditions when you set up the orchestration process.	Orchestration Status
Order Management	Manage Task Types	Manage the task types that group tasks and services for status management, jeopardy, orchestration processes, and run time behavior.	Orchestration Task
Global Order Promising	Manage Order Promising Server Profile Options	Manage profile options and values to control the Order Promising server, such as specifying the assignment set or enabling a web service so it can do available to promise.	Application Profile Value
Inventory Management	Manage Enterprise Scheduler Jobs for Logistics Common Applications	Manage Oracle Enterprise Scheduler jobs and the sources that they use to provide lists of values for common components in Logistics.	Enterprise Scheduler Job List of Values Source
Inventory Management	Manage Units of Measure	Manage the units of measure that your organization uses to identify the quantity for an item.	Unit of Measure Unit of Measure Class Unit of Measure Interclass Conversion Unit of Measure Intraclass Conversion
Product Information Management	Create Catalog	Create an item catalog, and add attachments and images to it.	Item Catalog, Item Category
Product Information Management	Manage Enterprise Scheduler Jobs for Product Management Common Applications	Manage Oracle Enterprise Scheduler jobs and the sources that they use to provide lists of values for common components in Product Information Management.	Enterprise Scheduler Job List of Values Source
Product Information Management	Manage Default Catalogs	Manage catalog assignments for each functional area.	Functional Area Item Catalog
Product Information Management	Manage Default Item Class	Manage the root item class.	Item Class
Product Information Management	Manage Item Organizations	Manage the structure that you use for each item in Product Information Management.	Item Organization

Application	Page	Description	Business Object
		Each structure contains details about the parent item, components, attachments, and descriptive elements.	
Product Information Management	Manage Key Flexfield for Catalogs	Manage flexfield segments for each catalog and the validation to use for the catalog classification key. You must set up the flexfield that you use for the catalog key to make sure Product Information Management works as expected.	Application Key Flexfield
Product Information Management	Manage Lifecycle Phases	Manage the phases of each item lifecycle.	Item Lifecycle Phase
Supply Chain Management Common Components	Manage Data Security Policies	Manage grants of entitlement for each user or job role on an object or attribute group according to a condition.	Application Data Security Policy

Common

Export and import business objects in Oracle Middleware Extensions for Applications in the Common family.

Page	Description	Business Object
Manage Administrator Profile Values	Manage settings and values for your profile options to control application behavior.	Application Profile Value
Manage Applications Core Administrator Profile Values	Manage settings and values for your profile options to control Oracle Middleware Extensions for Applications behavior.	Application Profile Value
Manage Applications Core Attachment Categories	Manage attachment categories for Oracle Middleware Extensions for Applications.	Application Attachment Category
Manage Applications Core Attachment Entities	Manage attachment entities for Oracle Middleware Extensions for Applications.	Application Attachment Entity
Manage Applications Core Descriptive Flexfields	Manage descriptive flexfields for Oracle Middleware Extensions for Applications.	Application Descriptive Flexfield
Manage Applications Core Messages	Manage messages for Oracle Middleware Extensions for Applications.	Application Message
Manage Applications Core Profile Categories	Manage categories to group profile options in Oracle Middleware Extensions for Applications according to their functional area. Use	Application Profile Category

Page	Description	Business Object
	categories to search for related profiles and to set up data security rules.	
Manage Applications Core Profile Options	Manage profile options that affect Oracle Middleware Extensions for Applications behavior, and specify the levels that you can use to set them.	Application Profile Value
Manage Applications Core Standard Lookups	Manage lookup values for Oracle Middleware Extensions for Applications.	Application Standard Lookup
Manage Applications Core Value Sets	Manage value sets for Oracle Middleware Extensions for Applications.	Application Flexfield Value Set
Manage Attachment Categories	Manage categories for attachments for security purposes.	Application Attachment Category
Manage Attachment Entities	Manage the default repository folders to use when storing attachments for each application entity.	Application Attachment Entity
Manage Common Lookups	Manage lookups that are common across applications and that you use in common lookup views.	Application Common Lookup
Manage Currencies	Manage ISO standard currencies.	Application Reference Currency
Manage Data Security Policies	Manage entitlements for each user or job role on an object or group of attributes according to a condition.	Application Data Security Policy
Manage Descriptive Flexfields	Manage segments and other parts of the descriptive flexfields that you use to store details that Order Management doesn't typically store.	Application Descriptive Flexfield
Manage Document Sequence Categories	Manage categories that group documents for sequencing purposes.	Application Document Sequence Category
Manage Document Sequences	Manage document sequences to create an audit trail that identifies the application that created the transaction.	Application Document Sequence
Manage Extensible Flexfields	Manage segments and other parts of the extensible flexfields that you use to store details that Order Management doesn't typically store.	Application Extensible Flexfield
Manage Industries	Manage ISO industries.	Application Reference Industry

Page	Description	Business Object
Manage ISO Languages	Manage ISO languages.	Application Reference ISO Language
Manage Key Flexfields	Manage flexfield segments and validation. You must set up most key flexfields so Order Management works as you expect it to.	Application Key Flexfield
Manage Languages	Manage installed languages.	Application Reference Language
Manage Messages	Manage messages that you use in each application.	Application Message
Manage Natural Languages	Manage natural, spoken languages.	Application Reference Natural Language
Manage Profile Categories	Manage categories that group profile options according to functional area. Use categories to search for profiles and to set up your data security rules.	Application Profile Category
Manage Profile Options	Manage profile options that affect application behavior, and specify the levels that you can use to set them.	Application Profile Value
Manage Reference Data Sets	Manage sets that you can use to separate and share reference data across organizations.	Application Reference Data Set
Manage Set Assignments for Set Determinant Type	Manage assignments for your reference data sets.	Application Reference Data Set Assignment
Manage Set Enabled Lookups	Use codes to manage lookups that vary depending on the value of the reference data set.	Application Set-Enabled Lookup
Manage Standard Lookups	Manage lookups that are common across applications and that you use in lookups.	Application Standard Lookup
Manage Taxonomy Hierarchy	Manage the hierarchy of functional units that make up Oracle Applications, from product families and applications to functional components.	Application Taxonomy
Manage Territories	Manage ISO territories.	Application Reference Territory
Manage Time Zones	Manage time zones.	Application Reference Time Zone
Manage Tree Labels	Manage the labels to use for each tree element in a user interface.	Application Tree Label

Page	Description	Business Object
Manage Tree Structures	Manage tree structures so you can group business rules into a family of trees.	Application Tree Structure
Manage Trees and Tree Versions	Manage trees in a hierarchy so you can group and consolidate details that exist in your organization.	Application Tree
Manage Value Sets	Manage value sets to validate the content of a flexfield segment.	Application Flexfield Value Set
Register Descriptive Flexfields	Register your descriptive flexfields so you can use them.	Application Descriptive Flexfield
Set Activity Stream Options	Set options that determine the types of activities to display in the Activity Stream area in Oracle Applications.	Application Activity Stream Configuration

Business Intelligence

Export and import business objects in the Transactional Business Intelligence application in the Business Intelligence family.

Page	Description	Business Object
Configure Descriptive Flexfields for Transactional Business Intelligence	Specify how to validate and display a descriptive flexfield in Transactional Business Intelligence. Use descriptive flexfields to add attributes to entities.	Application Descriptive Flexfield
Configure Key Flexfields for Transactional Business Intelligence	Specify the segments and validation to do for Transactional Business Intelligence. You must specify these flexfields so Transactional Business Intelligence works as you expect it to.	Application Key Flexfield

Customer Relationship Management

Export and import business objects in the Customer Relationship Management family.

Application	Page	Description	Business Object
Marketing	Manage File Import Mappings	Manage mappings between columns in a source file in a source system and columns in a staging table in Order Management. Use these mappings when you import business objects, such as sales leads, customers, contacts, or sales catalogs.	File Import Mapping

Application	Page	Description	Business Object
Marketing	Manage File Import Objects	Manage business objects, such as sales leads and opportunities, that you import from a file.	File Import Object
Trading Community Model	Enable Click to Dial	Enable automated dialing when clicking a phone number.	Application Profile Value
Trading Community Model	Manage Geography Lookups	Manage the lookup values that provide choices for a geography, such as different ways to validate an address.	Application Standard Lookup
Trading Community Model	Manage Import Lookups	Manage the lookup values that provide choices about how to process data import, such as batch status, batch identifier, batch configuration, or process status.	Application Standard Lookup
Trading Community Model	Manage Source System Descriptive Flexfields	Manage fields that your users can use to enter details. Validate descriptive flexfields according to values that the user enters in other areas of the page.	Application Descriptive Flexfield
Trading Community Model	Manage Source System Entities	Manage entities for source systems. For example, use one source system to import customer data, and use another to import customers and contacts.	Trading Community Original System Mapping Trading Community Source System
Trading Community Model	Manage Source System Lookups	Manage the lookup values that provide choices for your source system, such as original system type.	Application Standard Lookup
Trading Community Model	Manage Trading Community Source Systems	Manage the source system for a trading community.	Examine and define the types of information imported for each source system

Human Capital Management

Export and import business objects in the Human Capital Management family.

Application	Page	Description	Business Object
Global Human Resources	Manage Enterprise HCM Information	Manage details about an enterprise, such as the default work day.	Enterprise
Global Human Resources	Manage Locations	Manage locations for your enterprise. For example, create the	Location

Application	Page	Description	Business Object
		locations where people work or the locations of your organizations.	
HCM Configuration Workbench	Review Enterprise Configuration	Examine a functional summary or a detailed technical inventory of all objects.	Enterprise Configuration

Financials

Export and import business objects in the Financials Common Module application in the Financials family.

Page	Description	Business Object
Manage Business Unit	Manage details about business units. Control and report on transactions, and share reference data sets across applications.	Business Unit, Business Unit Detail
Manage Business Unit Set Assignment	Manage data for your business rules and policies. Assign this data to business units.	Business Unit Set Assignment

Application Toolkit

Export and import business objects in the Application Toolkit family.

Page	Description	Business Object
Manage Help Security Groups	Manage security groups to determine who can access your applications. Associate each group with a duty role. Edit lookup codes, database resource conditions, and data security policies.	Application Data Security Policy Application Standard Lookup
Map Reports to Work Areas	Select the reports that an Oracle Application work area displays in the Reports and Analytics area.	Reports and Analytics Mapping
Set Help Options	Set the options for your help features, such as collaboration, access to a web site, settings for modified help content, or access to an Oracle User Productivity Kit library.	Help Configuration
Set Watchlist Options	Specify the Watchlist categories and items that are available at your site.	Watchlist User and Site Preference

Governance Risk and Compliance

Export and import business objects in GRC Application Access Controls Governor application of the Governance Risk and Compliance family.

Page	Description	Business Object
Manage Application Access Controls	Manage rules and resolve problems that happen when attempting to access an application.	Governance Risk and Compliance Setup Configuration
Manage Application Configuration Controls	Manage rules and resolve problems that happen that are related to configurations in an application.	Governance Risk and Compliance Setup Configuration
Manage Application Preventive Controls	Manage rules regarding how users interact in an application.	Governance Risk and Compliance Setup Configuration
Manage Application Transaction Controls	Manage rules and resolve problems that happen for transactions in an application.	Governance Risk and Compliance Setup Configuration

Related Topics

- Copy Setups Between Instances of Order Management

3 Integrate

Introduction

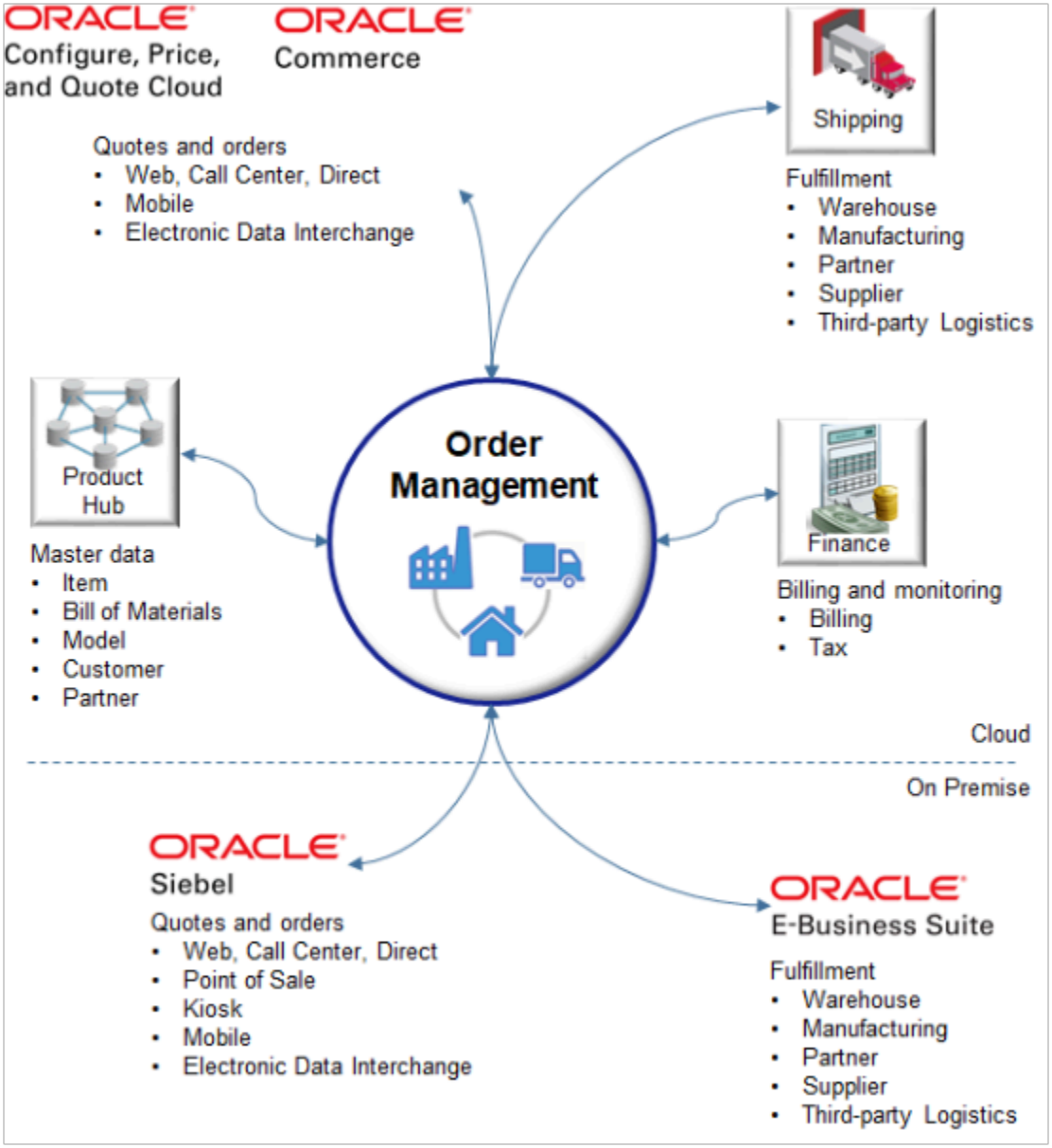
Overview of Integrating Order Management

Use web services to integrate Order Management with other applications and systems.

Order Management comes predefined with integrations to other Oracle services so you can use fulfillment processes that require minimal set up to get them up and running in your environment.

For example, Order Management and Oracle Inventory Management work together to fulfill each sales order that an Order Entry Specialist enters in the Order Management work area or that you import from a source system.

Use a web service to integrate with some other Oracle application, a third-party cloud application, or an on-premise application that your supply chain uses to complete the order-to-cash process. Here are some ideas.



Note

Integration	Description
Integrate with cloud processes.	Integrate cloud processes, including order-to-cash, drop-ship, back-to-back, configure-to-order, or internal orders. The integration includes predefined processes and simplified set up.
Integrate with other applications.	Oracle Order Management comes predefined with integrations to other Oracle Applications. • Oracle Inventory • Oracle Cost Management

Integration	Description
	• Oracle Manufacturing • Oracle Procurement • Oracle Financials • Oracle Configure, Price, and Quote.
Integrate through web services.	Use a web service. • Import each order right after you create it. • Create orders, then import them together as a group. • Send a request to your fulfillment system. • Receive a status from your fulfillment system. • Integrate to some other cloud or on-premise system. Inventory, shipping, receiving, finance, and Order Management are each an example of a fulfillment system.

Here are some examples of the functionality you can implement through integration.

- Bill shipping charges at the time of shipment.
- Send order status updates from Order Management to your fulfillment system when the update happens.
- Send order status updates from a warehouse management system, such as Oracle Warehouse Management, to Oracle Order Management when the update happens.
- Integrate Order Management with your current implementation of Oracle E-Business Suite (`EBS`).

Integrate Order Management Features

Get details about some of the features in Order Management that require integration.

- *More Guidelines for Setting Up Approval*
- *Overview of Setting Up Credit Check*
- *Overview of Setting Up Credit Cards*
- *Overview of Setting Up Projects in Order Management*
- *Overview of Setting Up Sales Agreements in Order Management*
- *Overview of Setting Up Trade Compliance*

Related Topics

- Integrate Order Management with Source Systems
- Use Integration Cloud Service with Order Management
- Manage Connector Details Between Order Management and Your Fulfillment System
- Fix Connection Problems with Source Systems

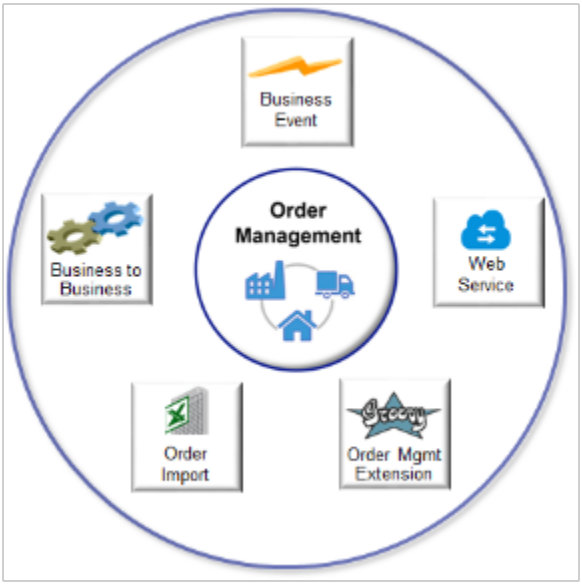
Guidelines

Guidelines for Integrating Order Management

Use a variety of technologies to integrate Order Management with a fulfillment system that resides outside of Oracle Applications.

Consider the Technologies You Use to Integrate

Use these technologies.



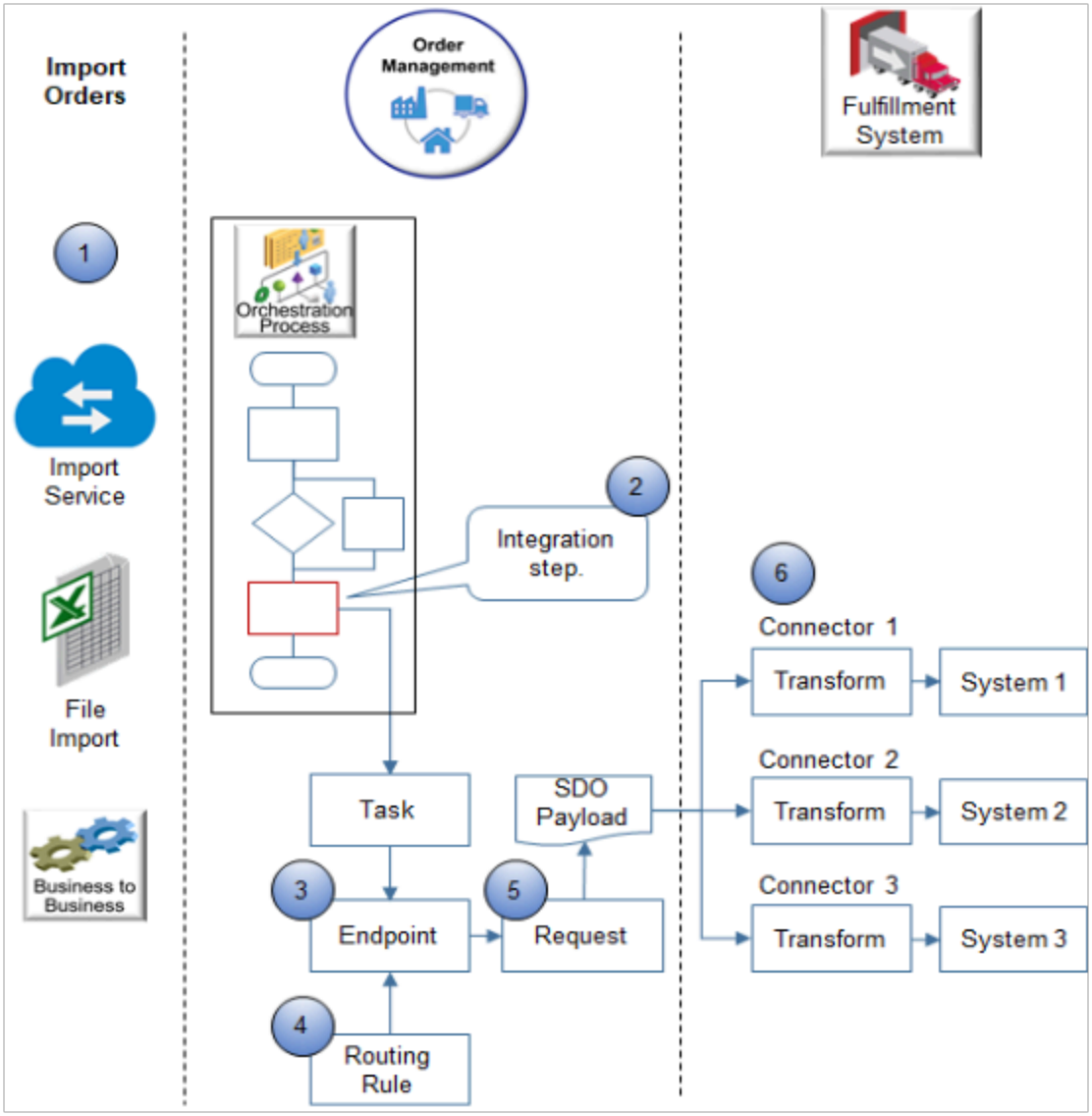
Get details.

Technology	Description	Get Details
Business Event	Integrate your business processes to operate across applications in the cloud or on premise.	*How Integration Cloud Service Integrates Order Management*
Web Service	Integrate with some other Oracle application, a third-party cloud application, or an on-premise application that your supply chain uses to finish the order-to-cash process.	*Guidelines for Using Web Services to Integrate Order Management*
Order Management Extension	Write your own Groovy script that modifies your Order Management deployment, implements your own functionality, and specifies the extension point that determines when to run this script.	*Overview of Creating Order Management Extensions*

Technology	Description	Get Details
File Based Data Import	Use an Excel file to simplify order import. This file contains a structure that the Oracle database requires for each database table.	*Overview of Importing Orders Into Order Management*
Business to Business	Automate message flow so Order Management can receive and process source orders from trading partners, then reply with an advance shipment notice after shipping finishes.	*Overview of Business-to-Business Messaging in Order Management*

Examine Your Flow

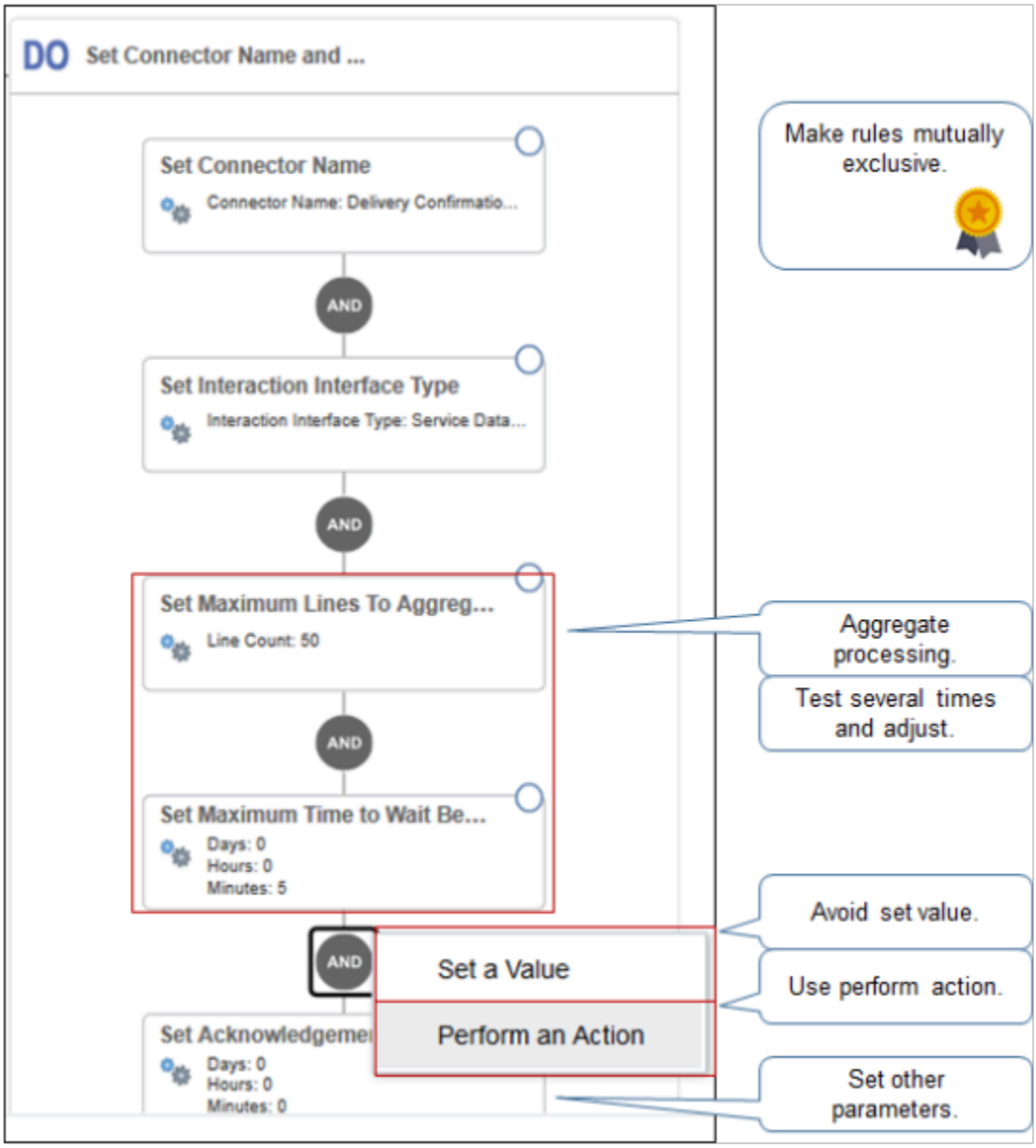
Most Order Management implementations use a set up that's similar to this flow.



Note

1. Use the import web service, file-based data import, or B2B messaging (business-to-business) to import source orders from your source system into Order Management.
2. A step in your orchestration process starts the integration. Order management assigns each source order to an orchestration process. The orchestration process orchestrates fulfillment for each fulfillment line. You add an integration step that integrates the orchestration process with your fulfillment system.
3. Web service endpoints identify each of your fulfillment systems that reside outside of Oracle Applications.
4. Routing rules specify the conditions to use when determining how to route each fulfillment line to each connector.
5. Order Management creates a service data object (SDO) and sends a request that includes the SDO payload. The SDO contains details about the endpoint, connector to use, fulfillment lines, and so on.
6. A connector web service transforms the SDO payload into a message payload that your fulfillment system can understand, and then calls the fulfillment system. You can create a separate connector for each of your fulfillment systems.

Use Business Rules



Note

- To avoid logic problems, make sure rules are mutually exclusive. Make sure no two rules can be true at the same time, or false at the same time. For details, see *Overview of Using Business Rules With Order Management*.
- Consider other attributes you can set.
    - To aggregate fulfillment lines before you send them to your fulfillment system, use the Maximum Lines to Aggregate and Send attribute and the Maximum Time to Wait Before Sending attribute. Run several tests and adjust these values after each test until you achieve the optimal balance between waiting and performance.

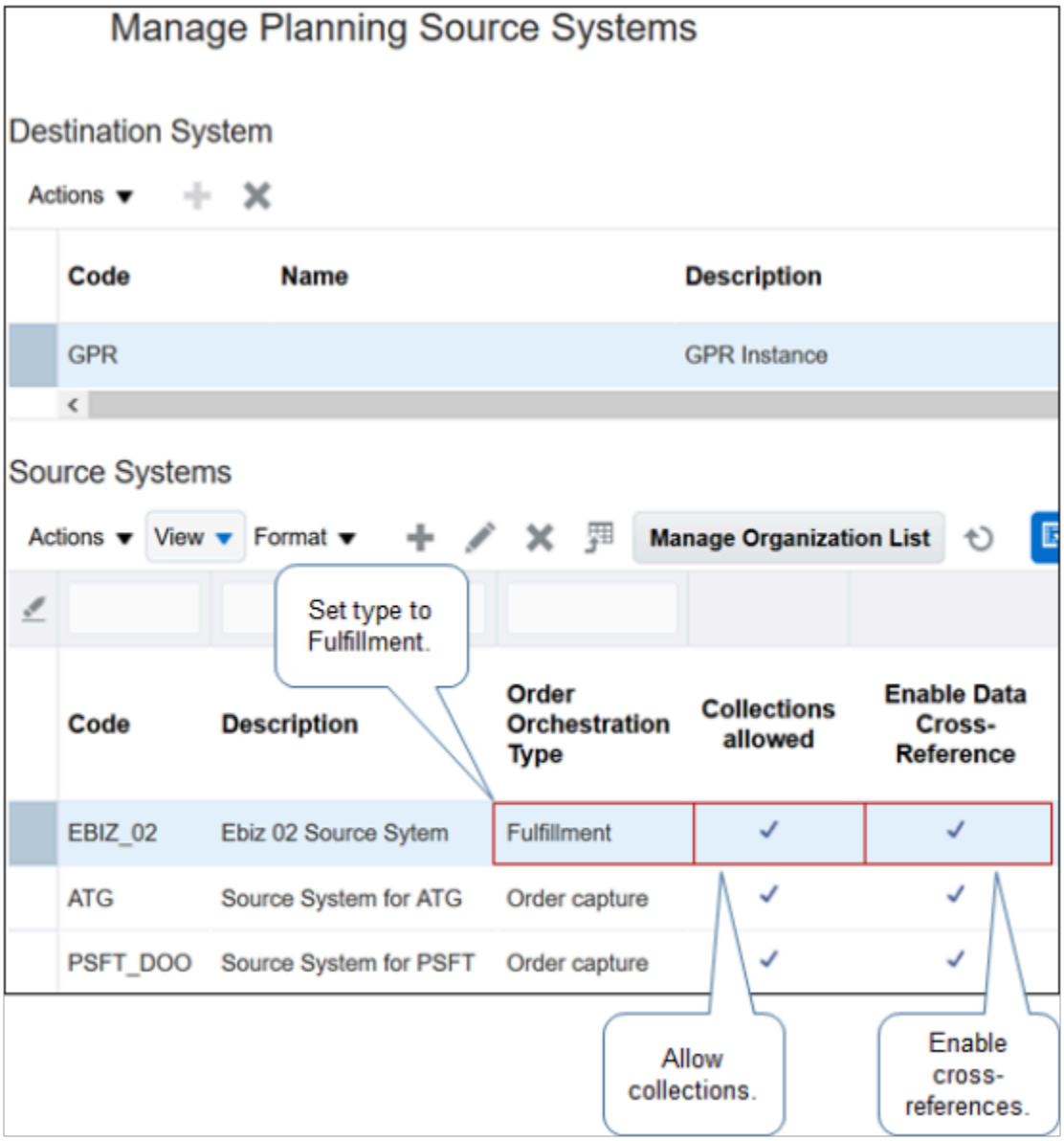
- Use other attributes to control processing, as necessary, such as Resolve Cross-Reference for Customer to determine whether to use a cross-reference, or Set `Acknowledgement` Timeout to determine how long to wait before exiting out of an implicit wait during an interaction with your fulfillment system.

For details, see *Manage Routing Rules*.

- Use Perform an Action. Avoid using Set a Value. Perform an Action automatically filters attributes and values for you to help make sure you specify values that the rule can understand.

Enable Cross-References

You must create and maintain cross-references that relate business data between your fulfillment system and Order Management. Use the Manage Planning Source Systems page as part of this set up.



For example, set these values.

Attribute	Value
Order Orchestration Type	Fulfillment
Collections Allowed	Contains a check mark.
Enable Data Cross-Reference	Contains a check mark.

For details about.

- Cross-references, see *Overview of Creating Cross-References in Order Management*.
- Managing planning source systems, see the section that describes how to collect data in *Quick Start for Setting Up Order-to-Cash*.

More

- Set up status behavior. Order Management typically sends statuses for each fulfillment line to your fulfillment system throughout the order fulfillment lifecycle. In return, your fulfillment system can send status updates to Order Management. You can set up this behavior. For details, see *Orchestration Process Status*.
- Determine whether you must add a wait step. For example, if a task happens almost instantly, as with credit check, then you typically don't need a wait step. However, a task that requires a long time to finish typically does require a wait step to allow the orchestration process to pause for the task to finish. For details, see *Overview of Pausing Orchestration Processes*.
- Set up failure logic. For example, if a task fails, then it goes into error recovery and you must retry or recover the task.

  Assume your analysis determines you must allow an Order Manager to override a delivery confirmation that fails and allow processing to continue. You can set up an extensible flexfield that allows the Order Manager to record the results of the transaction, then set up a pause task that uses the contents of the flexfield to allow the process to release the pause and continue to the next step.
- Use business events and web services with Integration Cloud Service. For details, see *How Integration Cloud Service Integrates Order Management*.
- Use order management extensions. For details, see the section about calling web services in *Guidelines for Using Extensions to Get Data from Oracle Applications*.
- Pay attention to case-sensitive usage. A number of the objects that you use during integration use case-sensitive text. Make sure you use the case-sensitive text that the predefined objects use and that you see in the documentation. For example:

  - Use TransactionInterfaceHeaderDff, don't use TransactionInterfaceHeaderDFF.
  - Use FLine, don't use Fline.

  Failing to use the correct case-sensitive text might not cause an obvious compiler error, but might result in runtime problems that are more difficult to troubleshoot, such as data not displaying on an invoice.

Related Topics

- Integrate Order Management with Source Systems
- Use Integration Cloud Service with Order Management
- Manage Connector Details Between Order Management and Your Fulfillment System
- Fix Connection Problems with Source Systems

Guidelines for Setting Up Your Integration

Use a proven sequence when you integrate Order Management.

Step 1: Get the Privileges That You Need

These privileges allow you to call web services that provide a response from your fulfillment system.

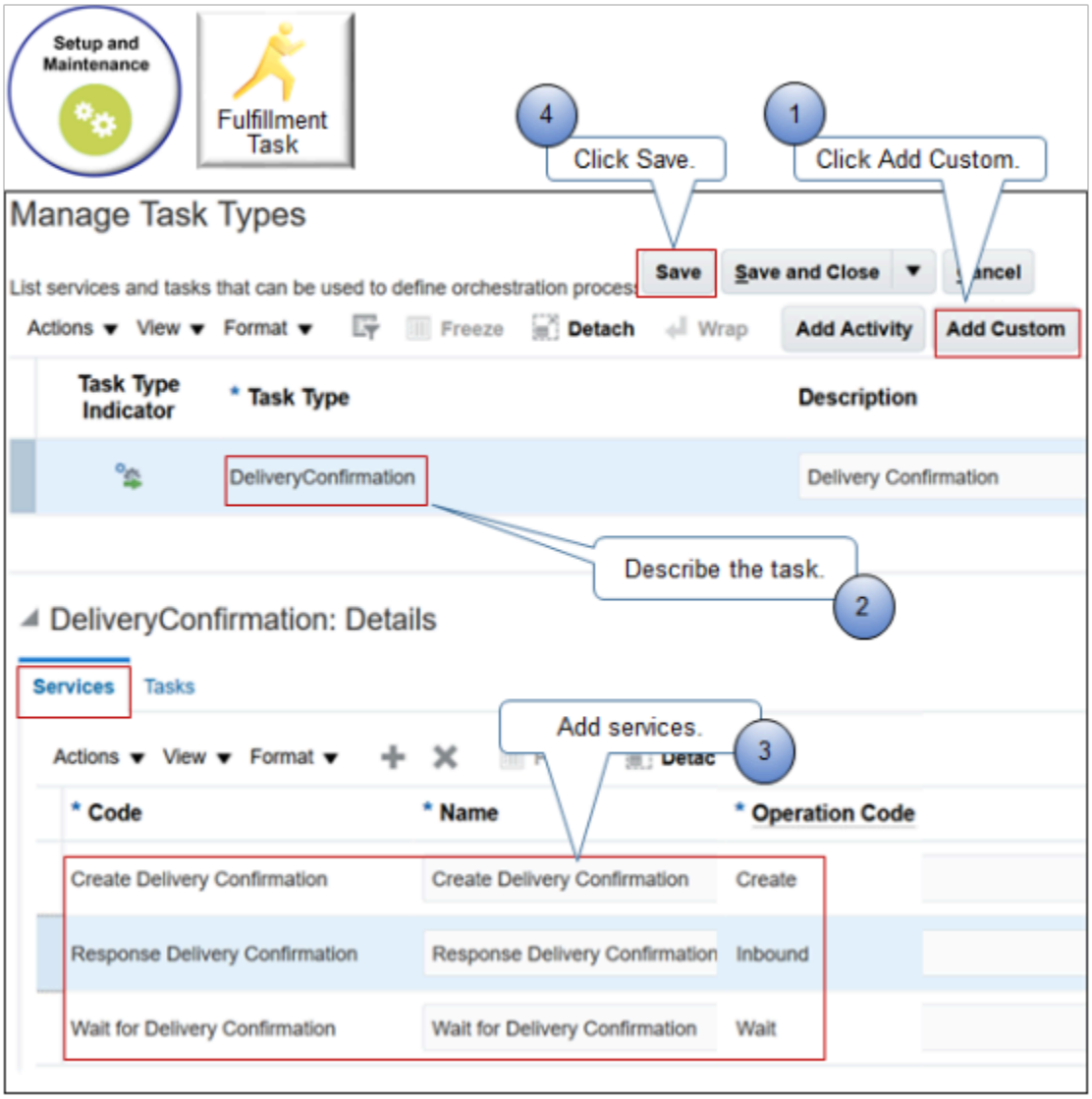
- Manage Orchestration Generic Web Service
- Manage Orchestration Order Activity Interface Web Service
- Manage Orchestration Order Fulfillment Interface Web Service
- Manage Orchestration Order Modification
- Manage Orchestration Order Purchasing Interface Web Service
- Manage Orchestration Order Receiving Interface Web Service
- Manage Orchestration Order Shipping Interface Web Service
- Manage Orchestration Order Template Interface Web Service
- Manage Web Service Interface to Transportation Data for Sales Order

Get the privilege that you need to call the web service. For example, if your integration manages shipping details, then get the Manage Orchestration Order Shipping Interface Web Service privilege.

For details about how to set up privileges, see *Security Reference for Order Management*.

Step 2: Set Up Task Type

Use the Manage Task Type page in the Setup and Maintenance work area to set up the task type.



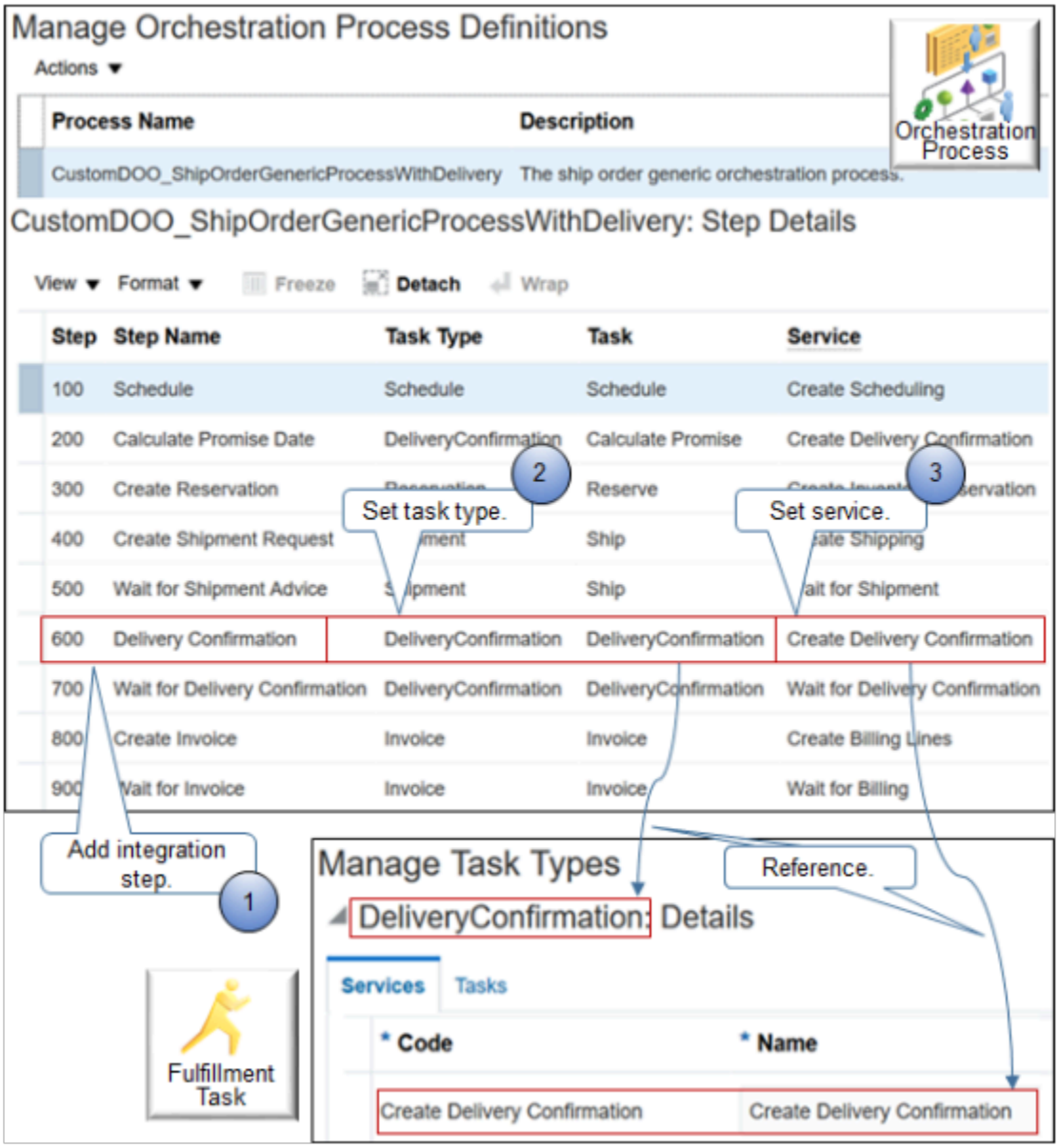
Note

1. Click **Add Custom**. Use this action you to specify services.
2. Use the Task Type attribute and the Description attribute to describe how your orchestration process uses the task, such as Delivery Confirmation.
3. Use the Services tab to add the service that does the task, such as Create Delivery Confirmation.
4. Click **Save** so you can reference the task type from your orchestration process.

The task type specifies the type of task that each orchestration process step does, such as to schedule a fulfillment line for shipment, to ship it, or to confirm delivery. For details, see *Create Your Own Task Type*.

Step 3: Add Integration Step

Add the step in your orchestration process that integrates with your fulfillment system.



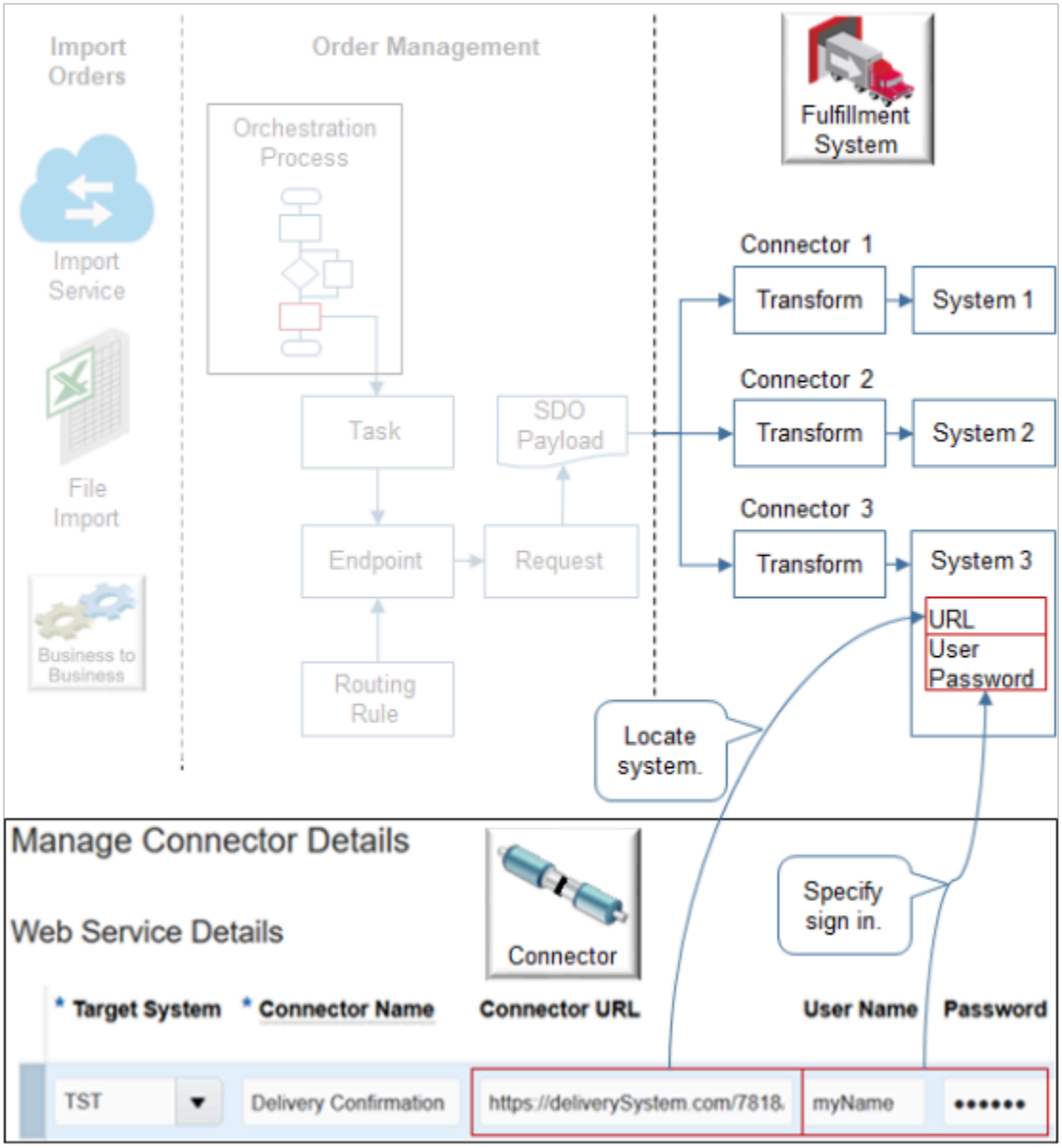
Note

1. Add the step at the point in your orchestration process where you must integrate. This example integrates after the orchestration process finishes the schedule, promise, reserve, and ship tasks, but before the invoice task.
2. On the integration step, set the Task Type attribute and the Task attribute to the task type you set up on the Manage Task Types page.
3. Set the Service attribute to the service you set up on the Services tab of the Manage Task Types page.

For details, see *Set Up Orchestration Processes*.

Step 4: Create Connector

Go to the Setup and Maintenance work area, search for Manage External Interface Web Service Details, then use the Manage Connector Details page to specify how to connect to your fulfillment system.



Note

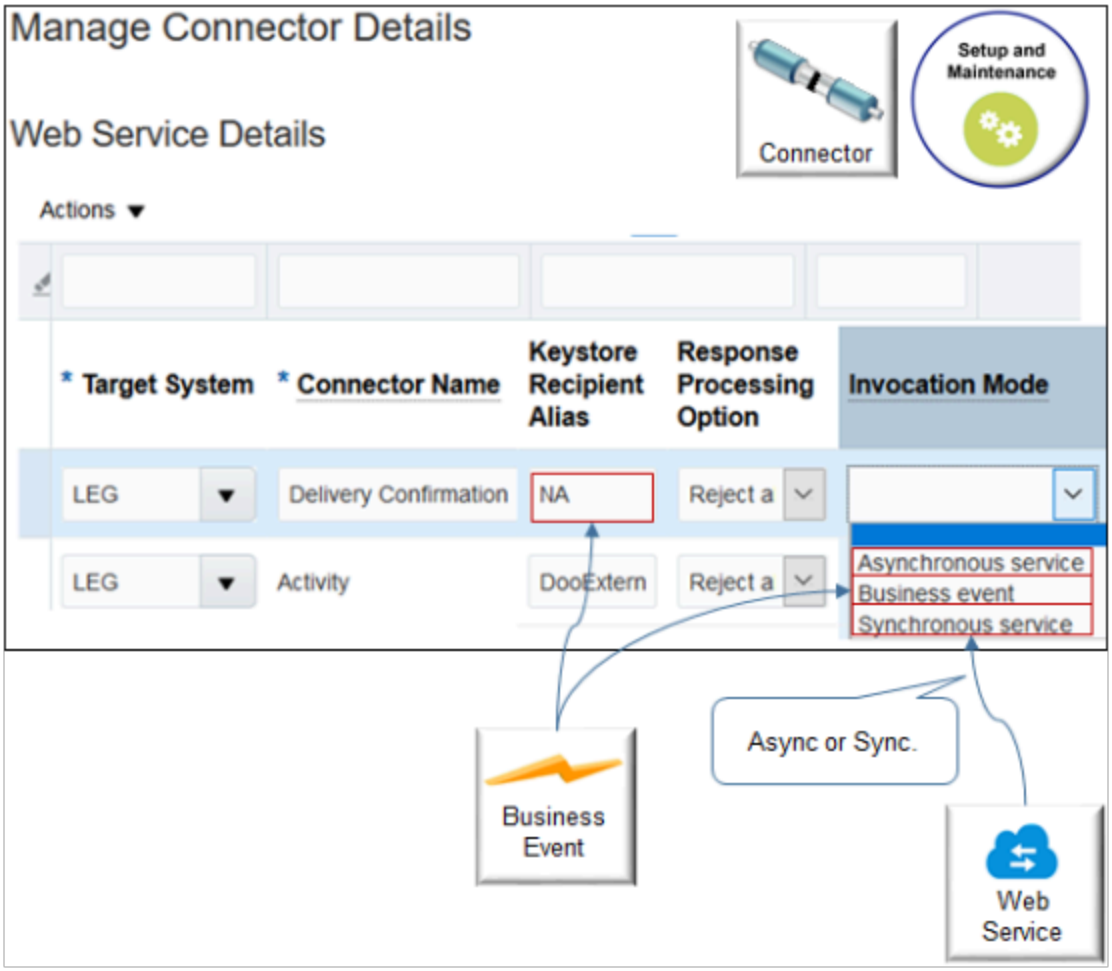
- Create a separate connector for each of your fulfillment systems.
- Use the Connector URL attribute to locate your fulfillment system. In this example, assume you defined a URL on the server that serves System 3.

  ```
  https://server:port/7818/soa-infra/services/default/ConfirmDelivery
  ```

  To make sure the connection is secure, use https in the URL. Don't use http.

- Use the User Name and Password attributes to specify the name and password you use when you sign into fulfillment system 3. In this example, assume you set up a user named myName on System 3.
    - Use a user name that works with an SSL security policy (secure sockets layers) that you set up on your fulfillment system.
    - Make sure the user name and password you specify are valid on your fulfillment system. Use them to verify that you can successfully sign into your fulfillment system. Note that this name isn't the name you use to sign into Oracle Applications.
    - Use CA signed certificates (certification authority) on your fulfillment system and on the connector.
- For details, see *Manage Connector Details Between Order Management and Your Fulfillment System*.

Set other attributes on the connector.

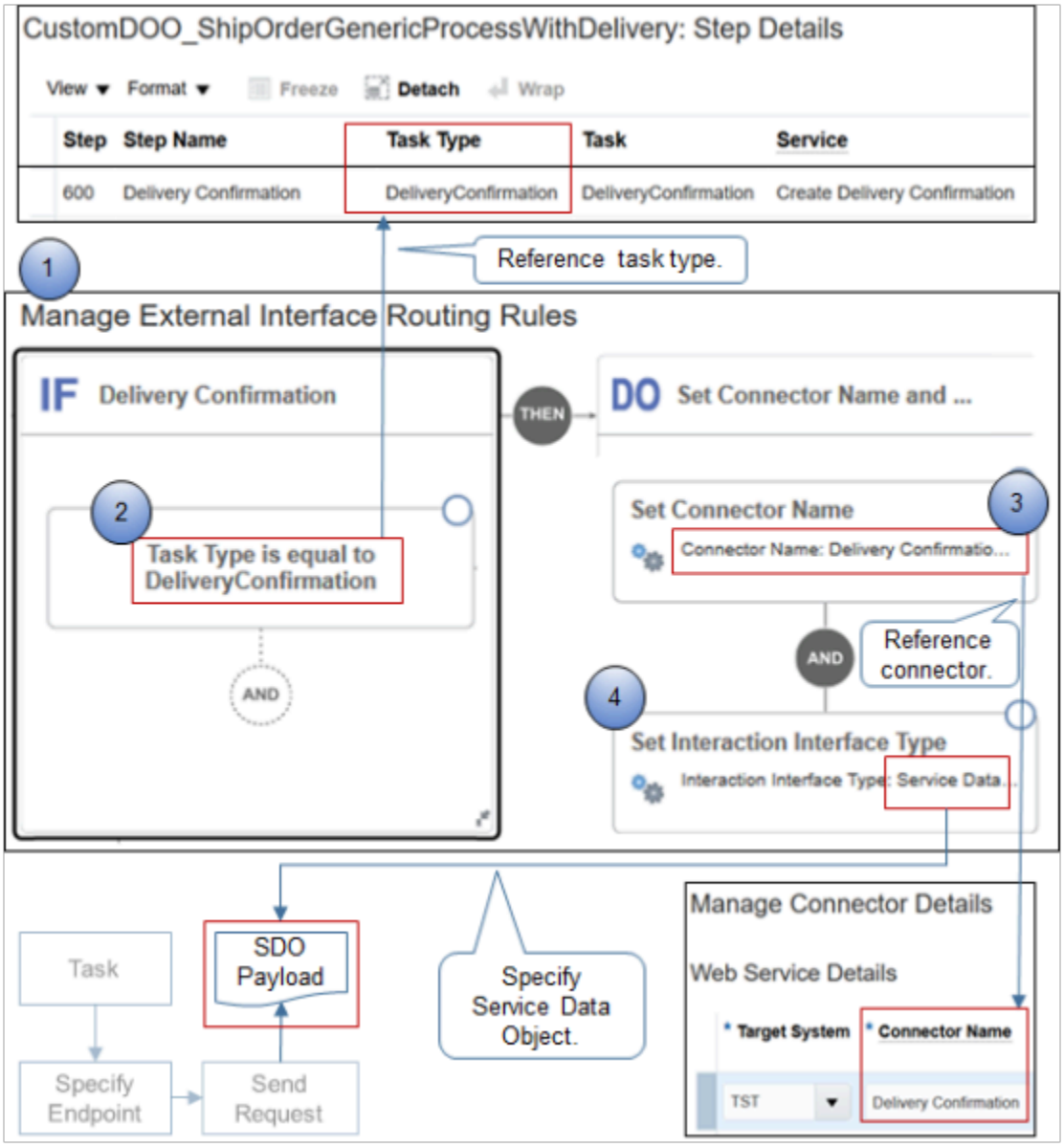


For example:

- If you use this connector to communicate a business event, then set the Keystore Recipient Alias attribute to NA, and set the Invocation Mode attribute to Business Event.
- If you use this connector with a web service, then set Invocation Mode to Asynchronous Service or Synchronous Service.

Step 5: Create Routing Rule

Create the rule that routes the orchestration process to the connector you just created.



Note

1. Use Visual Information Builder to create the rule. For details, see *Overview of Using Business Rules With Order Management*.
2. Add an If statement to determine whether the task type matches the type you specified in the Task Type attribute on the orchestration process step.

```
If Task Type is equal to DeliveryConfirmation
```

3. Add a Then statement to set the Connector Name attribute to the connector you created earlier that connects to your fulfillment system.

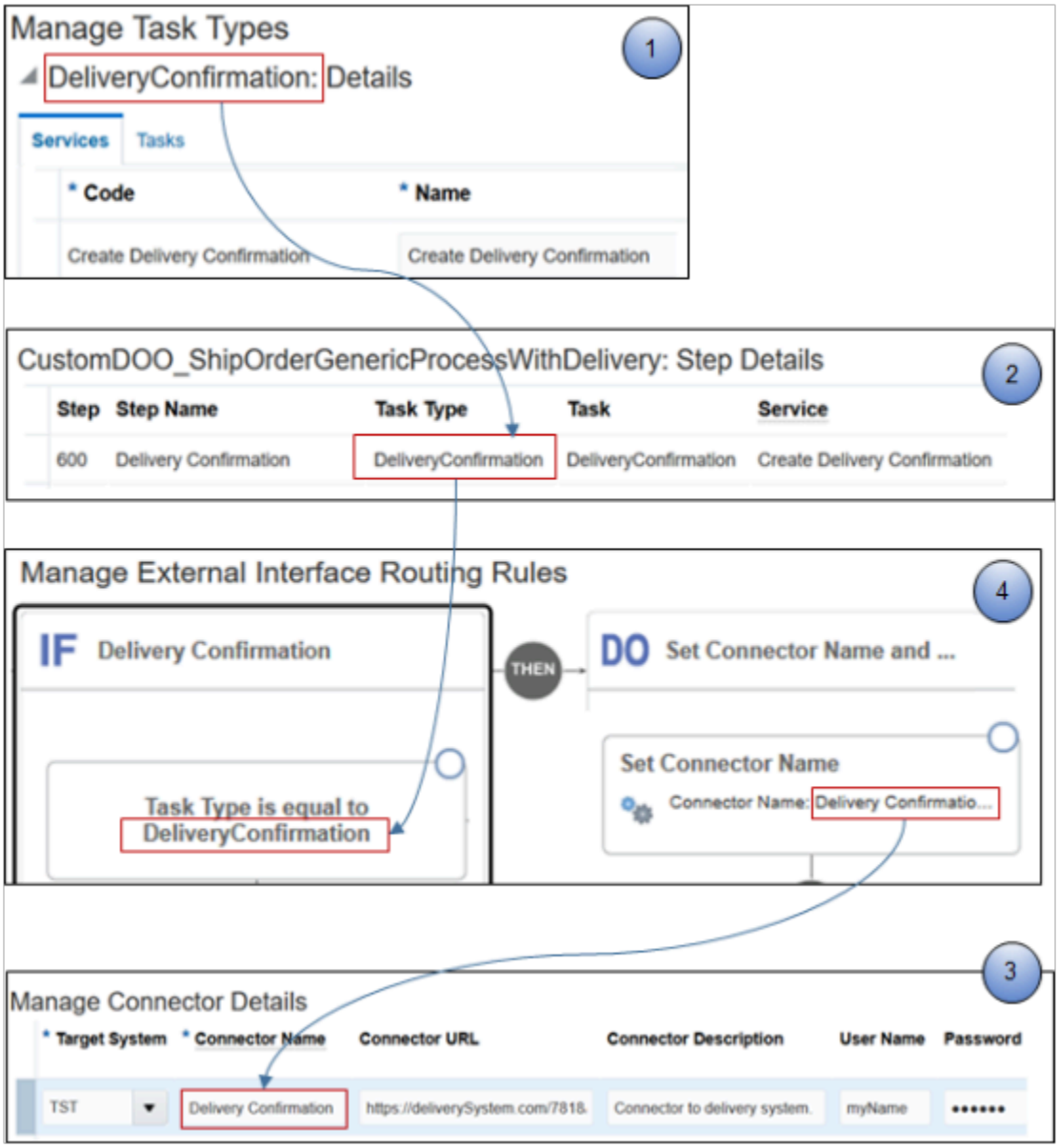
```
Set Connector Name: Delivery Confirmation
```

4. Add a Then statement to set the Interaction Interface Type attribute. Recall from the flow that order orchestration uses an SDO payload to communicate with your fulfillment system. You must use Service Data Object for any new implementation. Use other values only for backward compatibility to an earlier Oracle Applications update.

```
Set Interaction Interface Type: Service Data Object
```

Summary

Here's a summary of the set up you do.



Use these pages.

1. Manage Task Types to create the task type.
2. Manage Orchestration Process Definitions to add the integration step.
3. Manage Connector Details to create the connector.
4. Manage External Interface Routing rules to route the fulfillment line to your fulfillment system.

Related Topics

- Integrate Order Management with Source Systems
- Use Integration Cloud Service with Order Management
- Manage Connector Details Between Order Management and Your Fulfillment System
- Fix Connection Problems with Source Systems

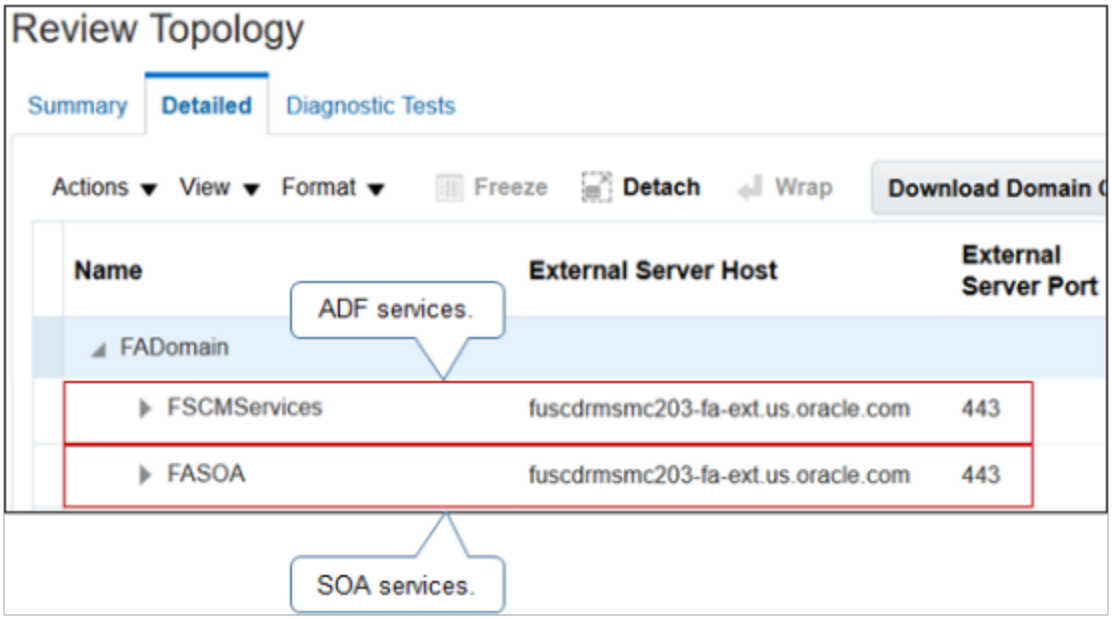
Guidelines for Sending the Response from Your Order Fulfillment System

The connector sends a request to your order fulfillment system, then the fulfillment system sends a response. You can use services in the response to communicate status updates from your fulfillment system to Order Management.

Service	Go Here for Details
FulfillmentResponse Service	*Web Services That You Can Use to Integrate Order Management*
Order Fulfillment Response Service	*Use Integration Cloud Service with Order Management*

Identify Hosts and Ports

Identify the hosts and ports that you will use to access your payloads. Go to the Setup and Maintenance work area, then click **Tasks > Review Topology**. For details, see *Identify Hosts and Ports for Order Management*.

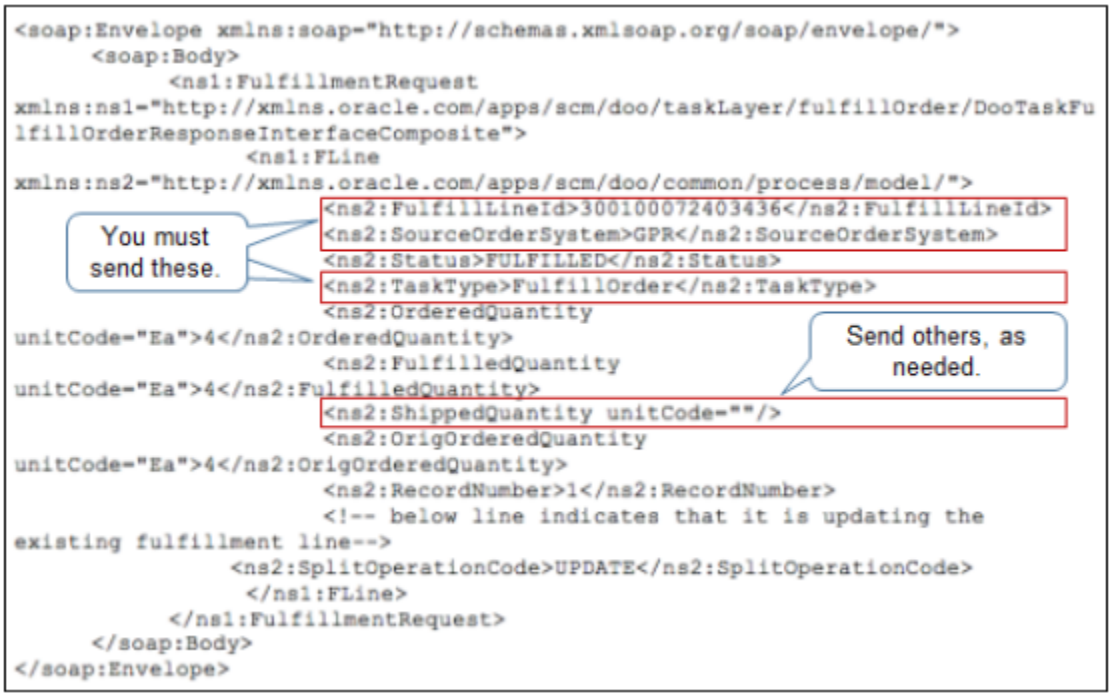


For example:

- If you use Oracle Application Development Framework (ADF), then use the host and port for FSCMServices.
- If you use Oracle Service-Oriented Architecture (SOA), then use the host and port for FASOA (Oracle Applications, Service-Oriented Architecture).

Set Up Payload for FulfillmentResponse Service

If you use Service-Oriented Architecture, then set up the payload for the FulfillmentResponse service.




```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Body>
            <ns1:FulfillmentRequest
xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/taskLayer/fulfillOrder/DooTaskFu
lfillOrderResponseInterfaceComposite">
                  <ns1:FLine
xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/common/process/model/">
                        <ns2:FulfillLineId>300100072403436</ns2:FulfillLineId>
                        <ns2:SourceOrderSystem>GPR</ns2:SourceOrderSystem>
                        <ns2:Status>FULFILLED</ns2:Status>
                        <ns2:TaskType>FulfillOrder</ns2:TaskType>
                        <ns2:OrderedQuantity
unitCode="Ea">4</ns2:OrderedQuantity>
                        <ns2:FulfilledQuantity
unitCode="Ea">4</ns2:FulfilledQuantity>
                        <ns2:ShippedQuantity unitCode=""/>
                        <ns2:OrigOrderedQuantity
unitCode="Ea">4</ns2:OrigOrderedQuantity>
                        <ns2:RecordNumber>1</ns2:RecordNumber>
                        <!-- below line indicates that it is updating the
existing fulfillment line-->
                  <ns2:SplitOperationCode>UPDATE</ns2:SplitOperationCode>
                   </ns1:FLine>
            </ns1:FulfillmentRequest>
      </soap:Body>
</soap:Envelope>
```

The payload that your fulfillment system sends must include these attributes.

```
<ns2:FulfillLineId>Id_number</ns2:FulfillLineId>
 <ns2:SourceOrderSystem>system_name</ns2:SourceOrderSystem>
```

For example:

```
<ns2:FulfillLineId>300100072403436</ns2:FulfillLineId>
<ns2:SourceOrderSystem>GPR</ns2:SourceOrderSystem>
```

where

- 300100072403436 is the fulfillment line Id.
- GPR is the source order system. Order Management typically uses the phrase `source order` to refer to an order that resides in an upstream order capture system. However, in this instance, SourceOrderSystem means any system that provides `input` details to FulfillmentResponse.

As an option, you can also send other attributes, such as ShippedQuantity, according to your business requirements.

Order Management needs SourceOrderSystem so it can get the user name, password, and Keystore Recipient Alias from the Manage Connector Details page and use them when it sends a reply to your order fulfillment system. It uses SourceOrderSystem to search the list of connectors on the Manage Connector Details page. It searches the list in alphanumeric order, and selects the first connector it finds where the value in the Target System attribute matches the

value in SourceOrderSystem. Make sure you add your connectors and name them so Order Management uses the one you need.

Assume SourceOrderSystem contains OrderCapture1, and the Manage Connector Details page contains three connectors.

Target System	Connector Name	User Name	Password	Keystore Recipient Alias
OrderCapture1	My Connector A	Administrator1	******	-
OrderCapture2	My Connector B	Administrator2	******	-
OrderCapture1	My Connector C	Administrator3	******	-

Order Management will use the Administrator1 user, password, and Keystore Recipient Alias from the row for My Connector A.

For details about how to use your payload, see *Connect Order Management to Your Fulfillment System*.

If you create your own task type, then you must use a slightly different payload. For details, see *Create Your Own Task Type*.

Set Up Payload for Order Fulfillment Response Service

If you use Application Development Framework, then set up the payload for the Order Fulfillment Response service.

The payload that your fulfillment system sends must include these attributes.

```
<com:IntegrationContextCode>context_code</com.IntegrationContextCode>
<com:FulfillmentSystem/>
<com:FulfillLineIdentifier>numeric_value</com:FulfillLineIdentifier>
 <com:TaskInstanceStatusCode>status_code</com:TaskInstanceStatusCode>
```

For example:

```
<com:IntegrationContextCode>DOO_TransportationPlanning</com.IntegrationContextCode>
<com:FulfillmentSystem/>
<com:FulfillLineIdentifier>3001000071295736</com:FulfillLineIdentifier>
<com:TaskInstanceStatusCode>DOO_TP_DELIVERED</com:TaskInstanceStatusCode>
```

where

- DOO_TransportationPlanning is the IntegrationContextCode.
- 3001000071295736 is the FulfillLineIdentifier.
- DOO_TP_DELIVERED is the TaskInstanceStatusCode.

Set Up Payload for Extensible Flexfield

You integrate an extensible flexfield with your fulfillment system the same way you integrate it during order import. The only difference is that instead of modifying the XSD file that you use during order import, you modify the XSD file that

you use when you integrate with your fulfillment system. For details, see the section that describes how to integrate with web services during order import in *Guidelines for Setting Up Extensible Flexfields in Order Management*.

Related Topics

- Integrate Order Management with Source Systems
- Manage Connector Details Between Order Management and Your Fulfillment System
- Fix Connection Problems with Source Systems

Oracle Applications

Overview

Overview of Integrating Order Management with Other Oracle Applications

Integrate Order Management to send details to some other Oracle application.

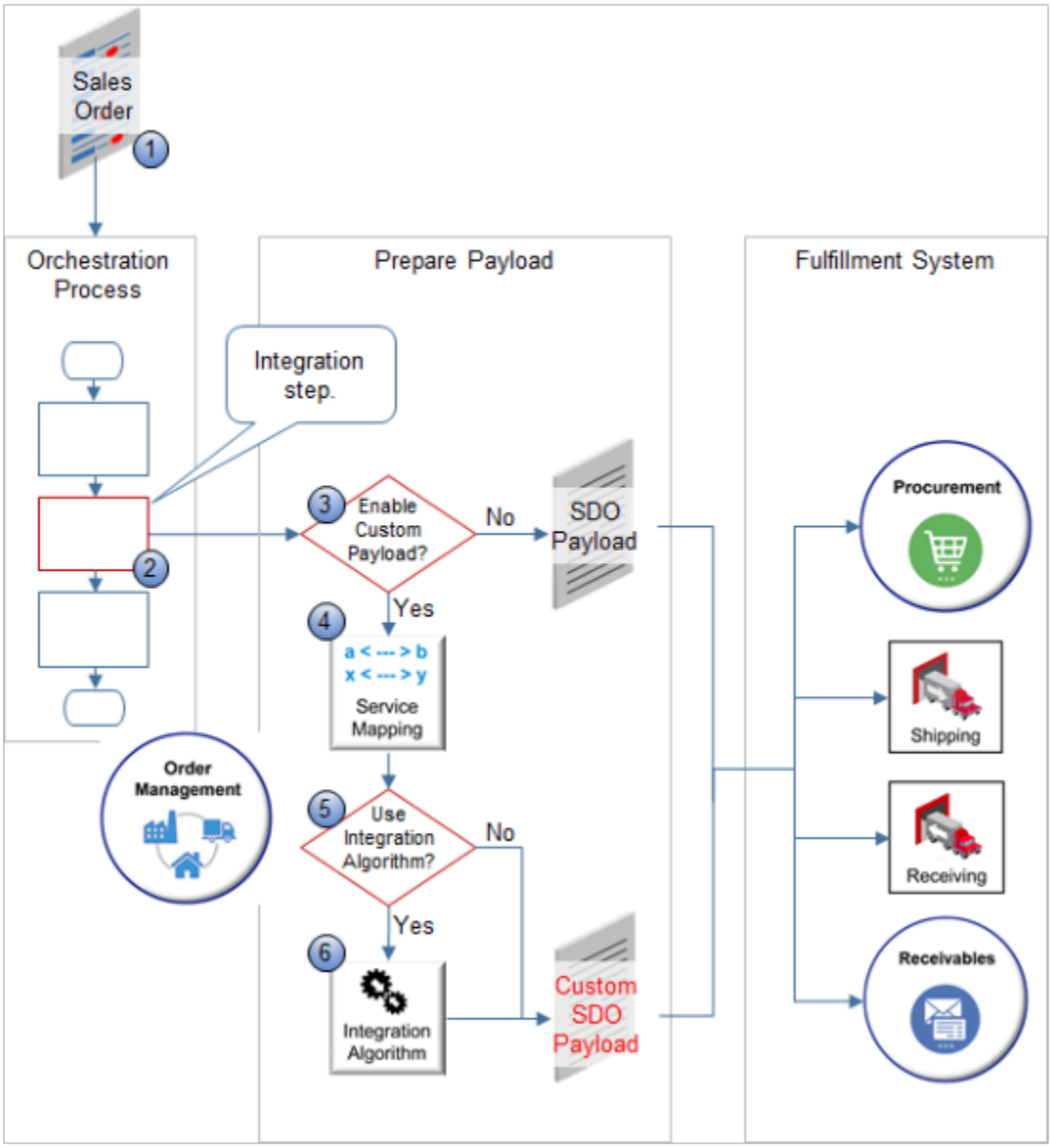
Use a predefined attribute to send details, or create an extensible flexfield in Order Management to store them, then map it to some other Oracle application.

For example:

Oracle Application	Description
Oracle Receivables	• Map the Purchase Order Line Number attribute on a fulfillment line in Order Management (CUSTOMER_PO_LINE_NUMBER) to the Invoice Line Level descriptive flexfieldin Oracle Receivables. Order Management doesn't come predefined to display this attribute. • Create an extensible flexfield on the order header or order line in Order Management. Map this flexfield to an interface column or descriptive flexfield in Oracle Receivables. • Send a concatenated value that represents the attributes of a covered item, such as Description, Start Date, or End Date, to the TranslatedDesription attribute in Oracle Receivables. Use an interface column in Oracle Receivables to send the data. • Send a covered item description, such as hardware, with coverage lines. Send freight charges that apply for a covered item as a separate line so you can invoice freight by itself.
Oracle Procurement	Create an extensible flexfield on a fulfillment line in Order Management to capture the price for the item. You can send the price that you negotiate with your supplier to the purchase request in Oracle Procurement. Order Management comes predefined to send a set of attributes to your fulfillment system for a purchase order in a drop ship flow. You can use a service mapping to send an attribute that isn't predefined. For example, send a price that you negotiate with your supplier during drop ship, or send a deliver-to address to your supplier during drop ship.
Oracle Receiving	Create extensible flexfields on the fulfillment line in Order Management to capture lot details and serial details. Send these details to the receiving request in Oracle Receiving.

Oracle Application	Description
Oracle Shipping	• Create an extensible flexfield on a fulfillment line in Order Management to capture shipment details. Send these details to a descriptive flexfield in Oracle Shipping. • Use a descriptive flexfield to send details to your receiving clerk for a sales order that includes a return material authorization.

How it Works



Note

1. Capture order details, such as in a sales order that you create in Order Management.
2. Use a predefined task type in the integration step of your orchestration process to send details to your fulfillment system.

3. Did you enable the Enable Custom Payloads for Downstream Integration feature?

Enabled	Description
Yes	Use the service mapping. You can set up the payload to integrate with your fulfillment systems.
No	Use the predefined SDO payload (service data object) and proceed to fulfillment. For an example that uses an SDO, see *Include Price, Discounts, and Shipping Charges in Your Payloads*.

4. Call the service mapping that you set up that maps attributes between Order Management and your fulfillment system.
5. Did you create an integration algorithm?

Created an Integration Algorithm	Value
Yes	Call it. Use the integration algorithm to do more complex logic, as necessary.
No	Use only the service mapping to create the payload.

6. Use the service mapping and the integration algorithm to create the payload.

Note

- You set up an integration algorithm and a service mapping when you create the integration.
- An integration algorithm uses logic that's similar to a pricing algorithm.
- A service mapping that you set up for an integration uses logic that's similar to the service mapping that you set up for pricing.
- You use the Pricing Administration work area to set up the integration algorithm and the service mapping that you use for integration. However, your service mapping and integration algorithm are completely separate from Pricing. You use Pricing Administration only to leverage some of the logic that Pricing uses for its service mappings and integration algorithms.
- An integration algorithm and the service mapping that you set up for an integration doesn't affect pricing.

Related Topics

- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Specify Transaction Types When You Integrate Order Management with Accounts Receivable
- Pricing Algorithms
- Service Mapping

Get Started with Integrating Order Management with Other Oracle Applications

Enable the features you need and determine whether you need a service mapping, integration algorithm, or both when you integrate Order Management with another Oracle application.

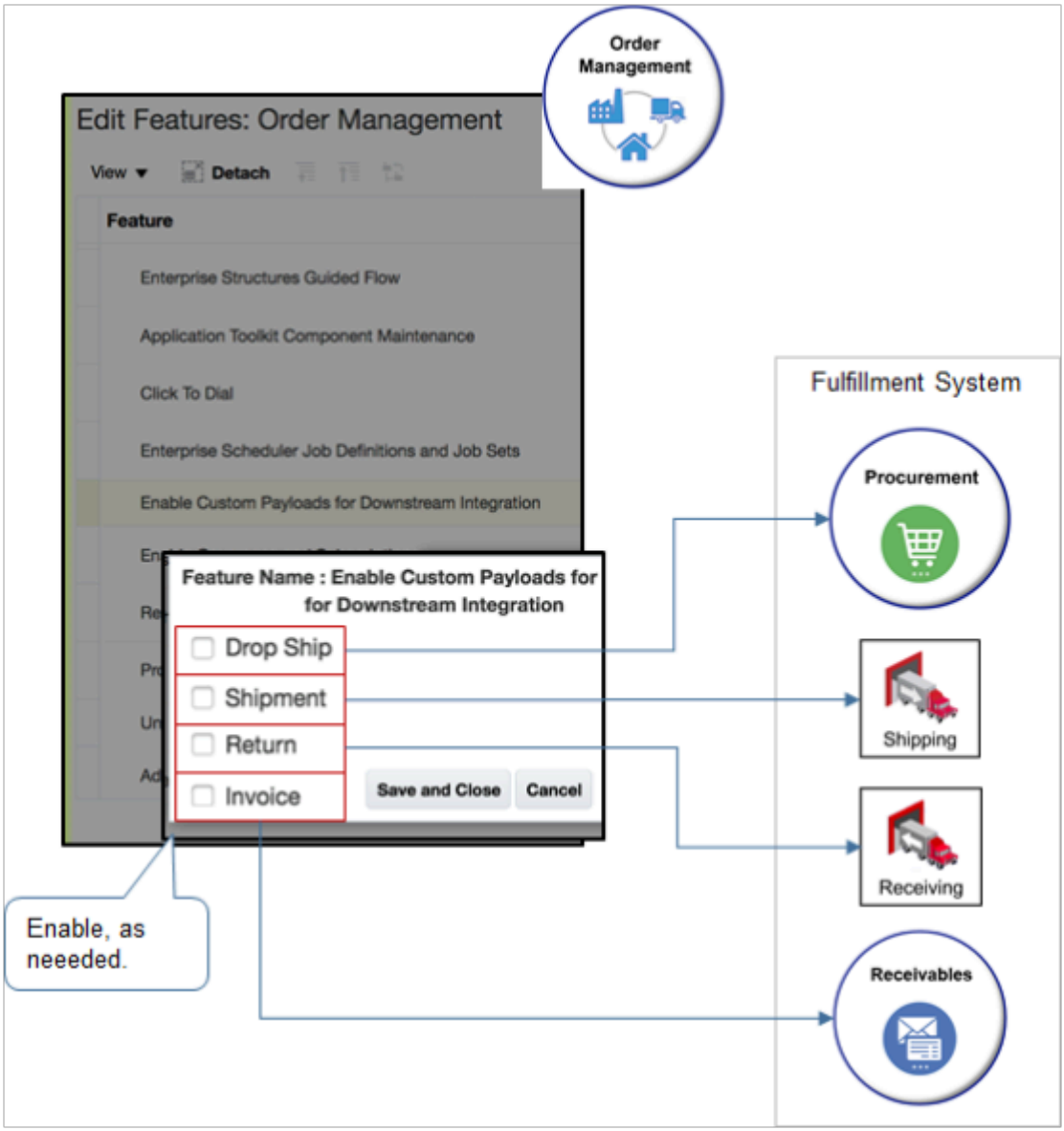
Enable Feature

Enable the feature you need.

1. Go to the Setup and Maintenance work area, then select the Order Management offering.
2. Click **Actions > Change Feature Selection**.
3. On the Edit Features page, in the Enable Custom Payloads for Downstream Integration row, click **Features**.
4. Add a **check mark** to the options you need.

Option	Enables Integration With
Drop Ship	Oracle Procurement
Invoice	Oracle Receivables
Return	Oracle Receiving
Shipment	Oracle Shipping

Option	Enables Integration With



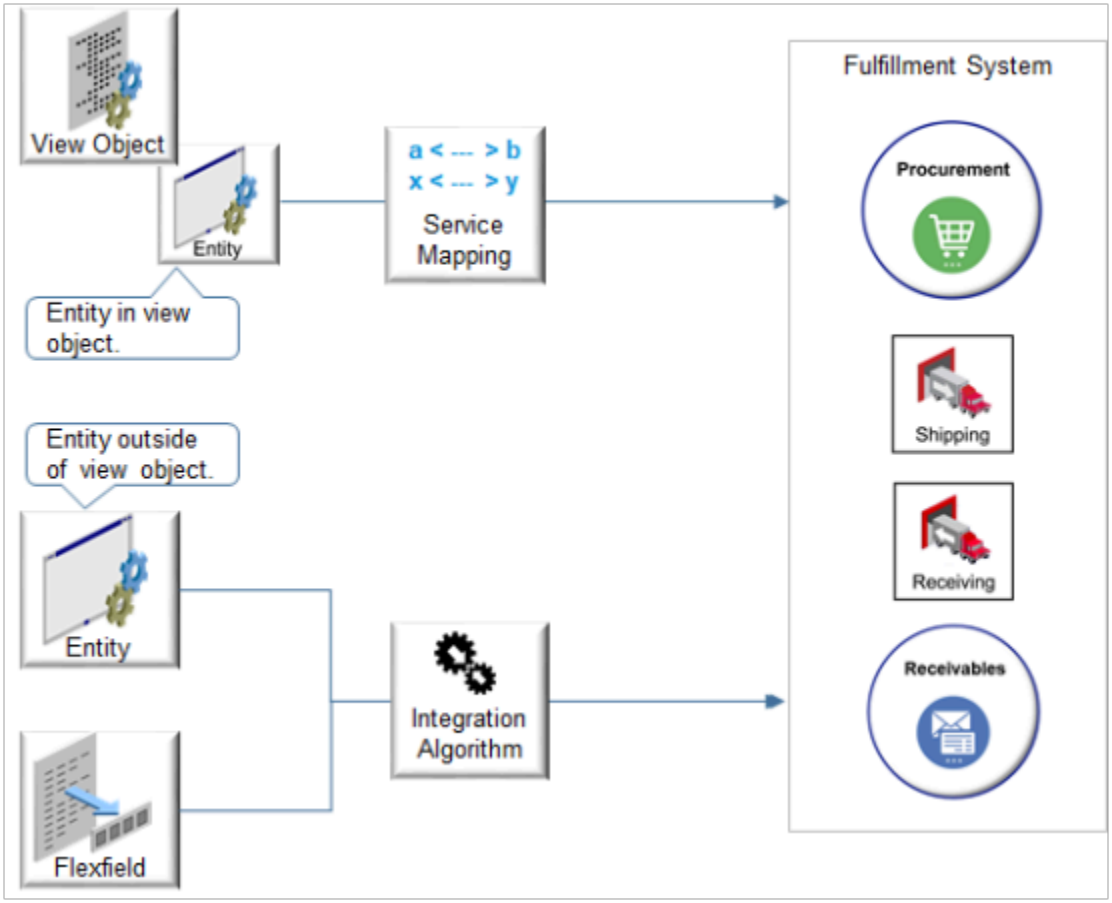
Note

- Each integration adds processing overhead. To improve performance, enable only the integrations you need.
- If you don't enable the feature, then you can still send a predefined SDO payload to your fulfillment system, but you can't send a payload that includes your service mapping or integration algorithm.

5. Click **Save and Close**, then click **Done**.

Service Mapping or Integration Algorithm?

Determine whether you need a service mapping, integration algorithm, or both.



Note

Object	Usage
Service mapping	Map an attribute from an entity in the view object that you're using as the source. Don't use an integration algorithm to do this mapping.
Integration algorithm	Map the attributes: • From an entity other than the view object that you're using as the source. For example, use an algorithm to map an attribute from the order header to an interface line. • With an extensible flexfield. • To a descriptive flexfield in Accounts Receivable.

The Service Mappings page and Algorithms page use values that are case sensitive. Enter exact names consistently, including _ (underscore), upper case, and lower case letters.

Apply guidelines when you set up your service mapping.

- Use a service mapping to map attributes from entities in the source view object. Don't use an integration algorithm to map them.

- If you add an attribute in the Sources tab, then make sure you also add it on the Services tab for each entity. If you forget to add it in Services, then the algorithm might run but not include your attribute in the payload.
- Use an expression or view object attribute to assign a value.
- Use the Expressions attribute to implement simple logic, such as If, If Then, If Then Else, or to concatenate.
- If you add an extensible flexfield.
    - Add it in the Entities tab, Sources tab, and Services tab. Add it in all three tabs. Don't add it in only one or two of these tabs.
    - Add an entity for your extensible flexfield one time for each context that you require.
    - Reuse an entity for your extensible flexfield in more than one service, as necessary.

Set up integration algorithms.

- Use an algorithm to map attributes from entities other than the view object or when you use a flexfield.
- Remember to enter the algorithm name on the Services tab.
- Remember to publish your algorithm.
- If you encounter a runtime error when a task runs, then correct the algorithm and recover the task.

Related Topics

- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Demonstration of Integrating Order Management with Other Oracle Applications

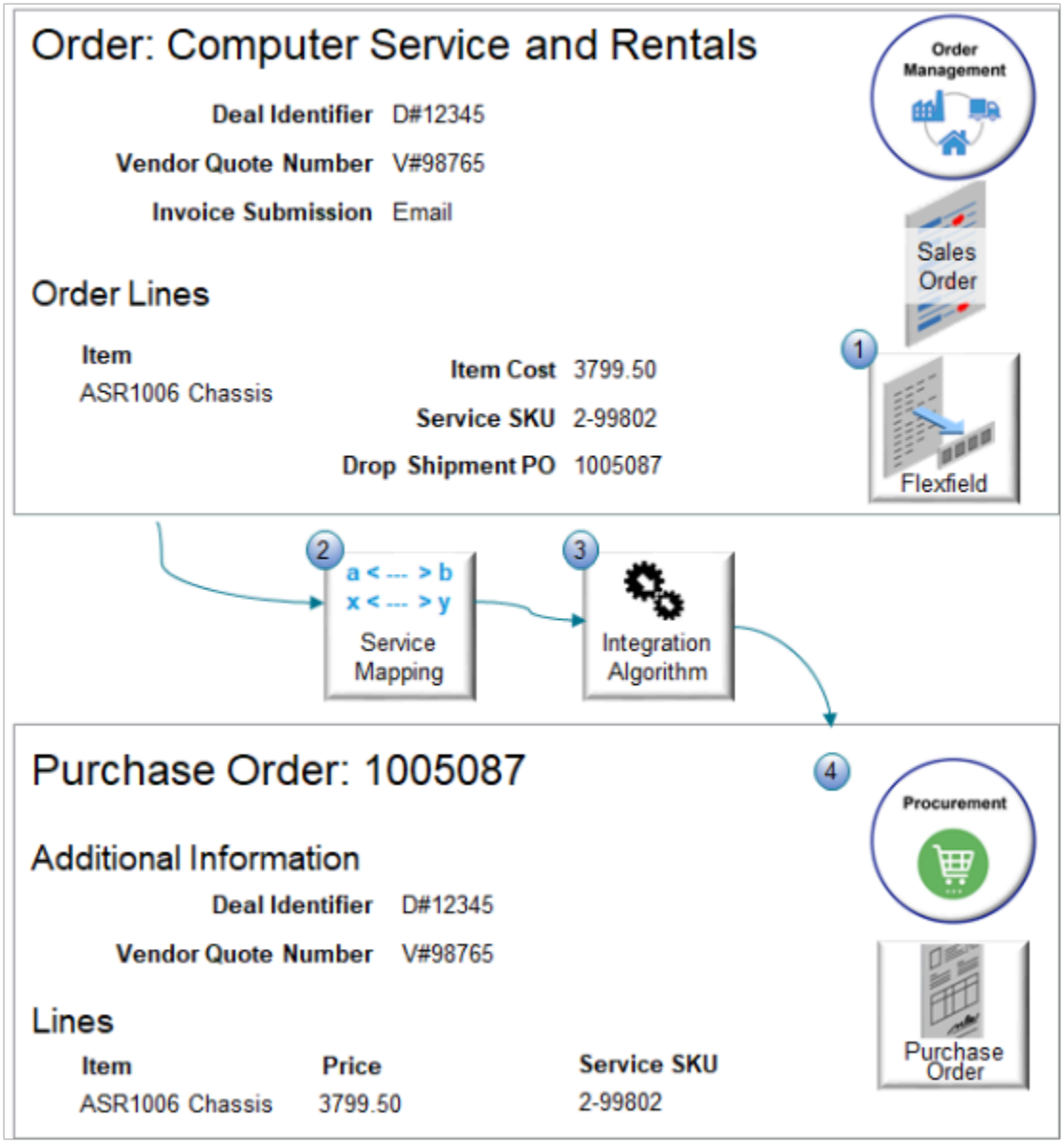
View and listen to a demonstration to learn how to integrate Order Management with another Oracle application.

Go to *Oracle Order Management Enhancements, Extensibility*, then see the Setup Demo section that starts at 10:47. This demonstration includes details about how to set up an integration that uses extensible flexfields, descriptive flexfields, a service mapping, and an integration algorithm.

Here's a summary of what the demo does.

Assume you have a special relationship with your customer where you use Oracle Procurement as part of a drop shipment. You need to capture details about each deal that you make with the customer on the order header, identify the quote for the vendor, and specify how you must submit the invoice on each sales order. And then on the order line, you:

- Override the price for the ASR1006 Chassis item and add your own price that you negotiate with your supplier. This is the price that Order Management will send on the purchase order for the drop ship to supplier. It isn't the price for your customer.
- Identify the stock keeping unit (SKU) for the item.
- Provide a purchase order number for the drop shipment.



What the Numbers Mean

1. You use extensible flexfields and descriptive flexfields to add attributes to the order header and the order line.
    - Deal Identifier
    - Vendor Quote Number
    - Invoice Submission
    - Item Cost
    - Service SKU
    - Drop Shipment PO
2. You use a service mapping to map these attributes between Order Management and Procurement.
3. You create an integration algorithm that specifies how to assign values to these attributes.

4. At run time, you use the Purchase Requisitions work area to verify that your integration accurately reflects data from Order Management.

As part of development, you create a mock up that illustrates the approach you will use to implement the requirement.

Demonstration That Integrates Order Management with Oracle Receivables

Here's another demonstration that you might find useful. Go to *Order Management Enhancements, Infrastructure*, then start the demonstration at 45:10.

Related Topics

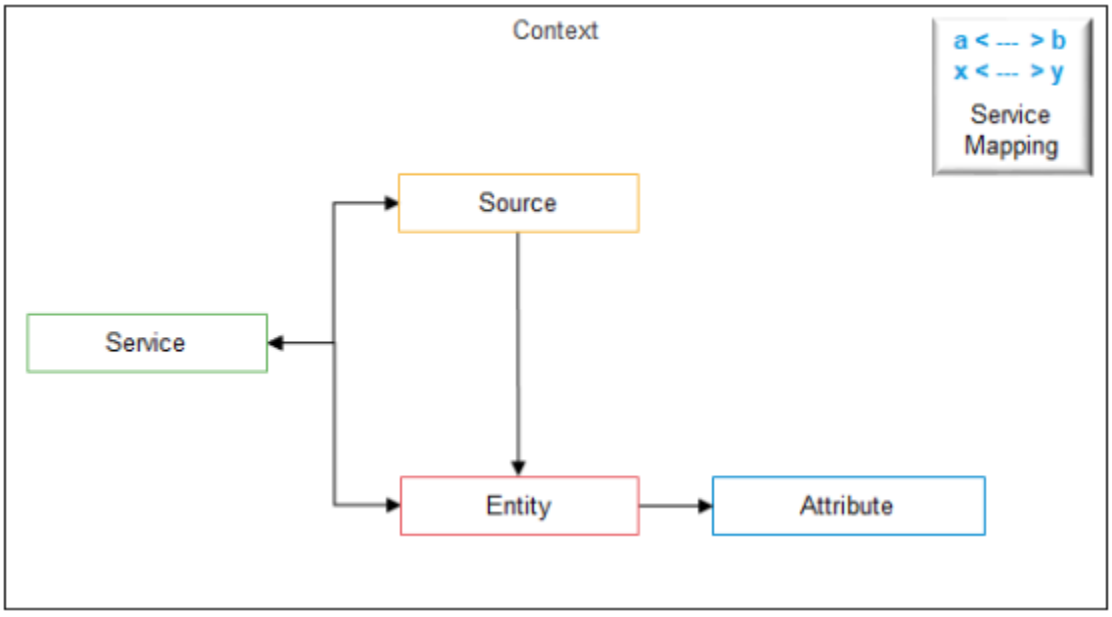
- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Specify Transaction Types When You Integrate Order Management with Accounts Receivable
- Pricing Algorithms
- Service Mapping

Guidelines

Create Your Service Mapping

Learn how to use a service mapping when you integrate Order Management with another Oracle application.

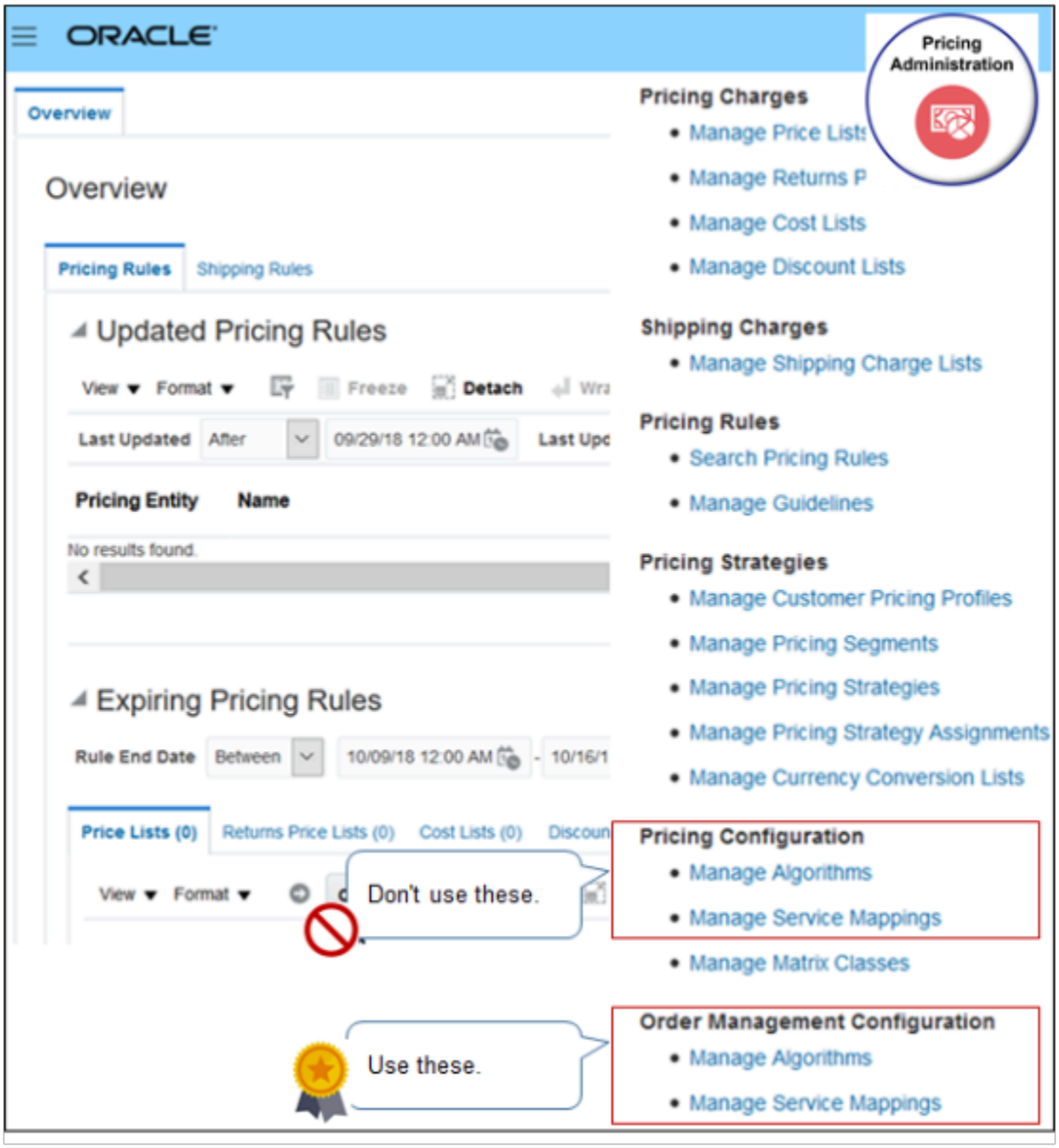
Understand Service Mapping



Note

Object	Description
Context	You specify the context for each service mapping. The context is the overall container for all objects that you create in your set up.
Entity	Specify the structure of each object. Examples of an entity include the order header, fulfillment line, purchase order request, and so on.
Source	Create what your mapping uses as the source for attribute values.
Service	Specify the attributes to send to your fulfillment system. You use it in the output that communicates with the fulfillment system. For example, in the structure of the SDO.
Algorithm	Optional. Instructions that implement complex logic.

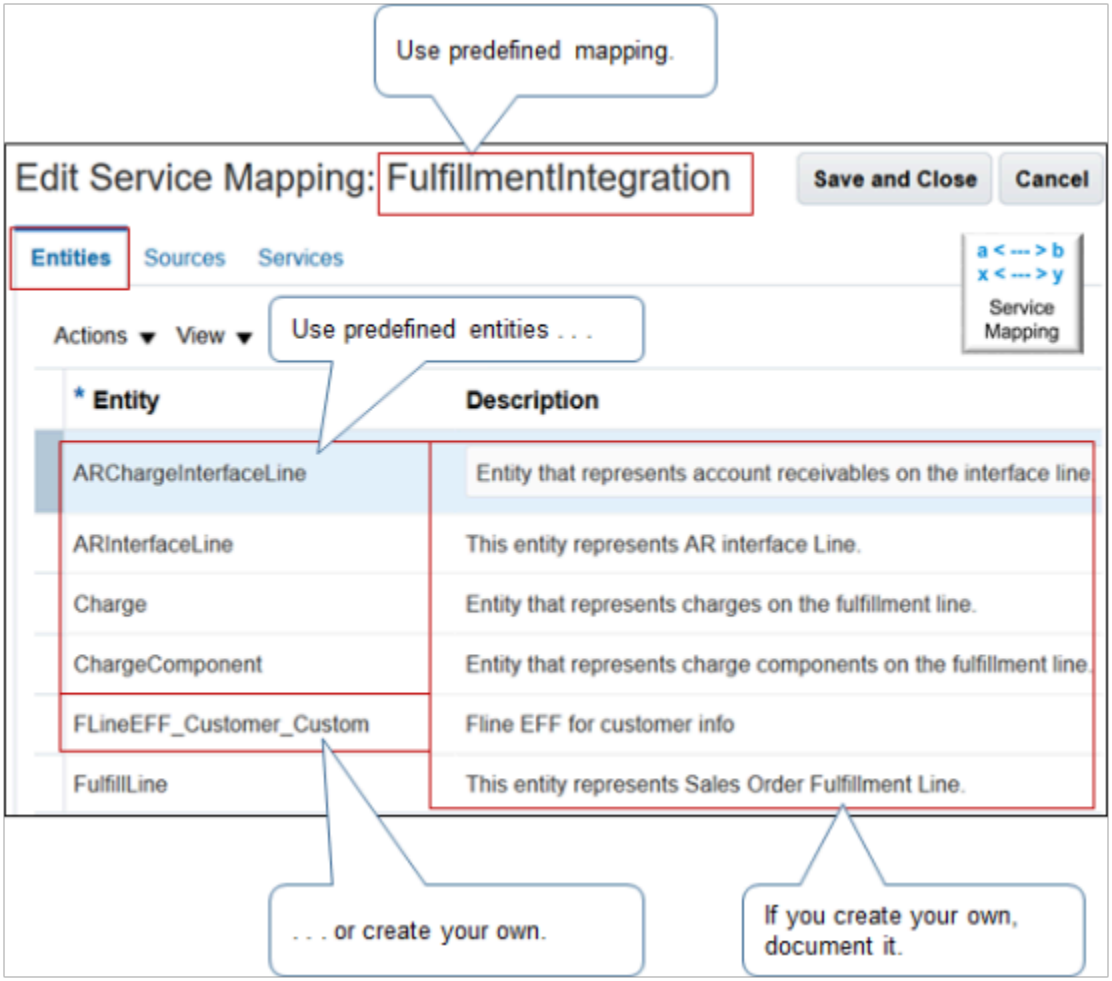
Use the Pricing Administration Work Area



Note

- You use the Pricing Administration work area to leverage the service mapping functionality that Pricing Administration provides, but you aren't setting up pricing.
- Make sure you have the privileges that you need to administer Order Management. Don't sign in pricing privileges.
- Go to the Pricing Administration work area.
- Click **Tasks**, then, under Order Management Configuration, click a **link**.
- Don't click links under Pricing Configuration.

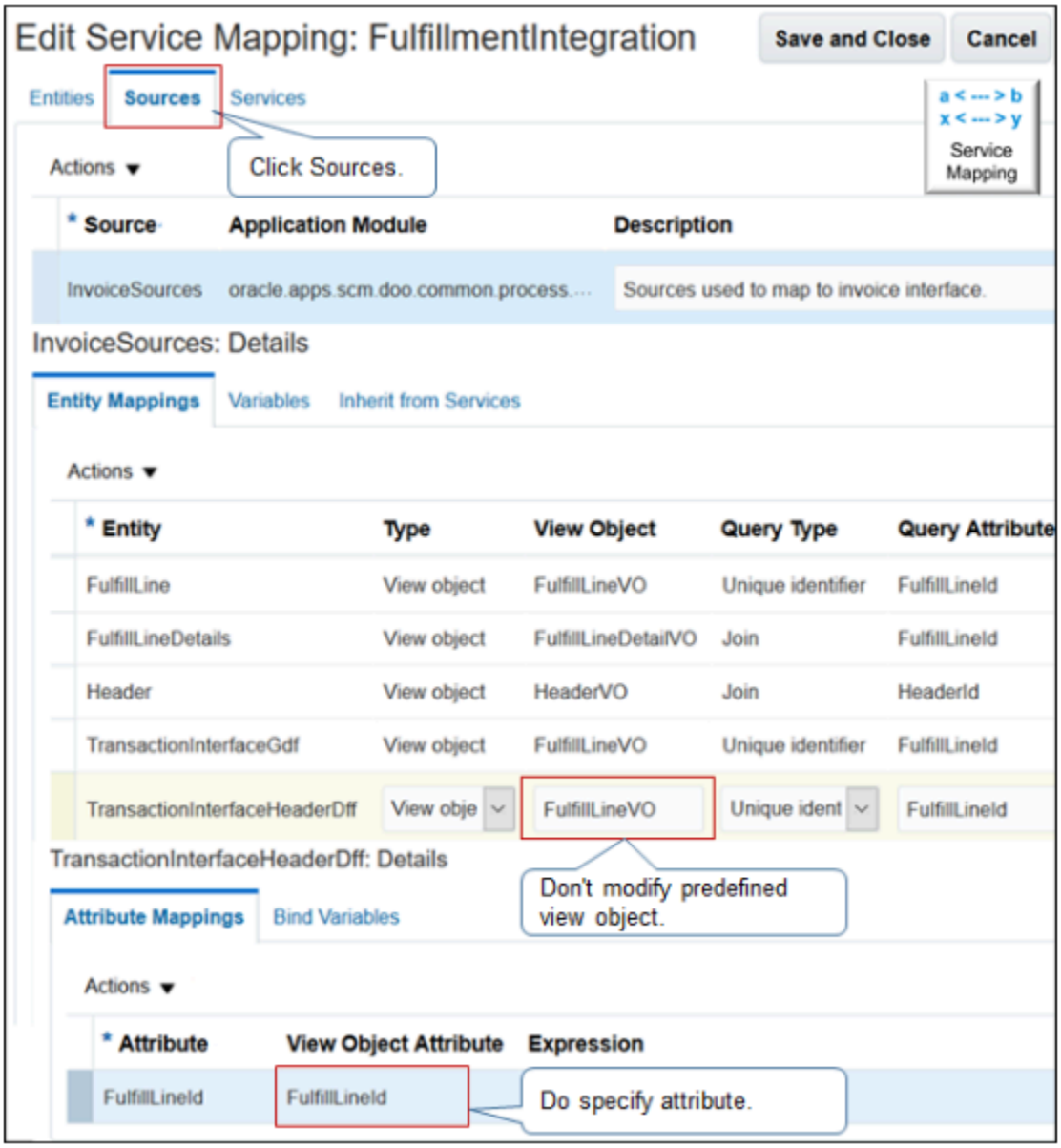
Use Predefined Mapping



Note

- Click **Manage Service Mappings**.
- Click **FulfillmentIntegration**. Use this predefined mapping for your deployment.
- Reduce maintenance and troubleshooting. Use predefined entities instead of creating new ones.
- If the entity name doesn't include suffix _Custom, then its predefined.
- If you create a new entity, then use suffix _Custom. Enter a complete description of what it does.
- To modify a service mapping, you must first create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.

Specify Source



Note

- Click the **Sources** tab on service mapping.
- Use a predefined source, when possible.
- Use predefined entities on each source, when possible.
- View object specifies the view object to use as the source for the value.
- Don't modify the view object of a predefined entity.
- View object attribute specifies the attribute in the view object to use as the source for the value. Do specify the view object attribute.

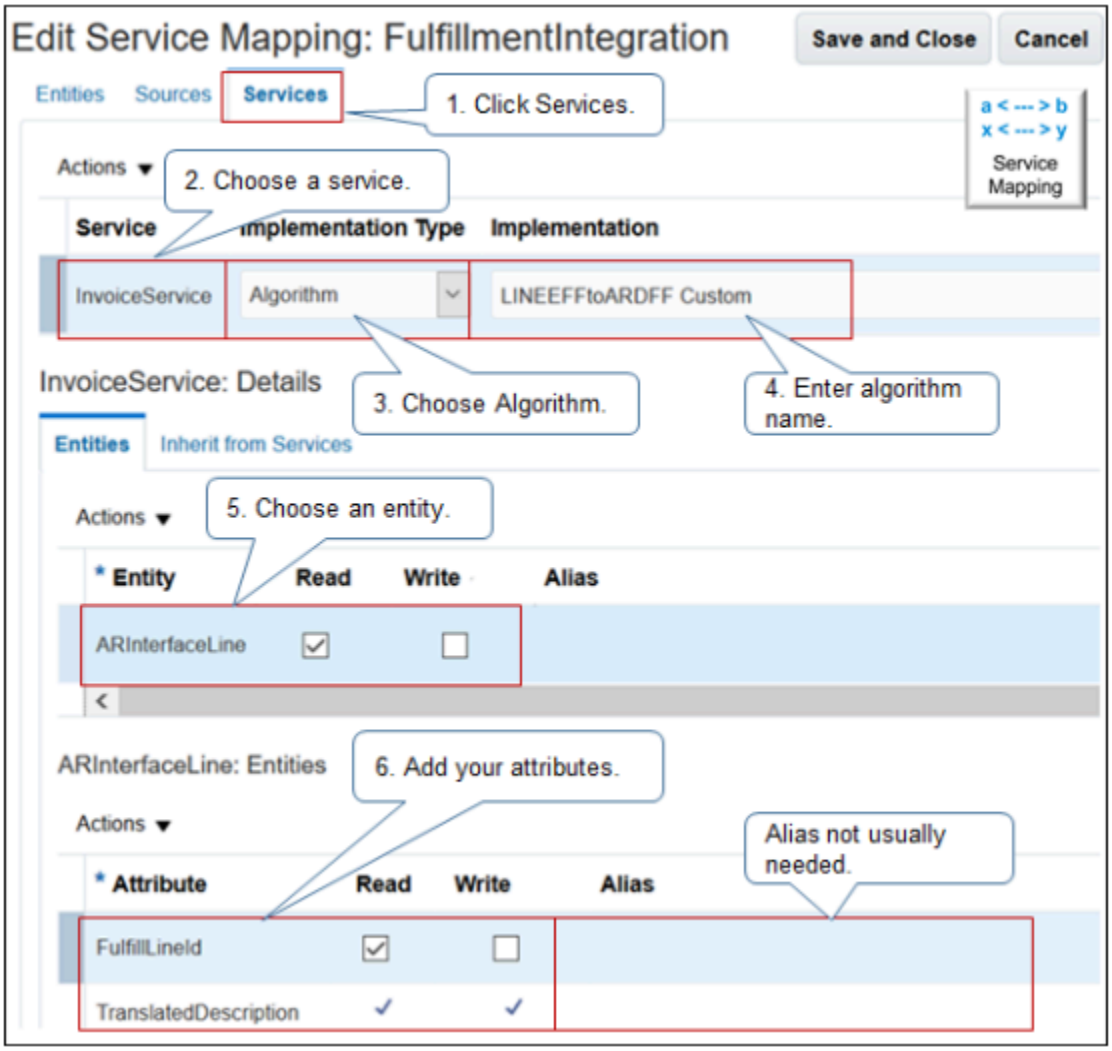
If you select a new source from a predefined entity, then you must add a suffix to entity name.

Flexfield	Suffix
Descriptive flexfield	DFF_Custom
Extensible flexfield	EFF_Custom
Global descriptive flexfield	GDF_Custom

To improve readability, include an underscore (_) immediately before the suffix.

```
myDescriptiveFlexfield_Dff_Custom
```

Specify Service



Note

1. Click the **Services** tab on service mapping.
2. Select a predefined service, such as InvoiceService.

3. Set the Implementation Type attribute to Algorithm.
4. If you plan to use an integration algorithm, then enter the name of your integration algorithm in attribute Implementation. If you don't set this value, then the service won't call your algorithm.
5. Select a predefined entity, such as ARInterfaceLine, when possible. Specify whether your fulfillment system can read or write the entity.
6. Add the attributes that you must integrate with your fulfillment system. Specify whether your fulfillment system can read or write the attribute.

   It typically isn't necessary to specify an alias for a predefined entity or attribute.

Related Topics

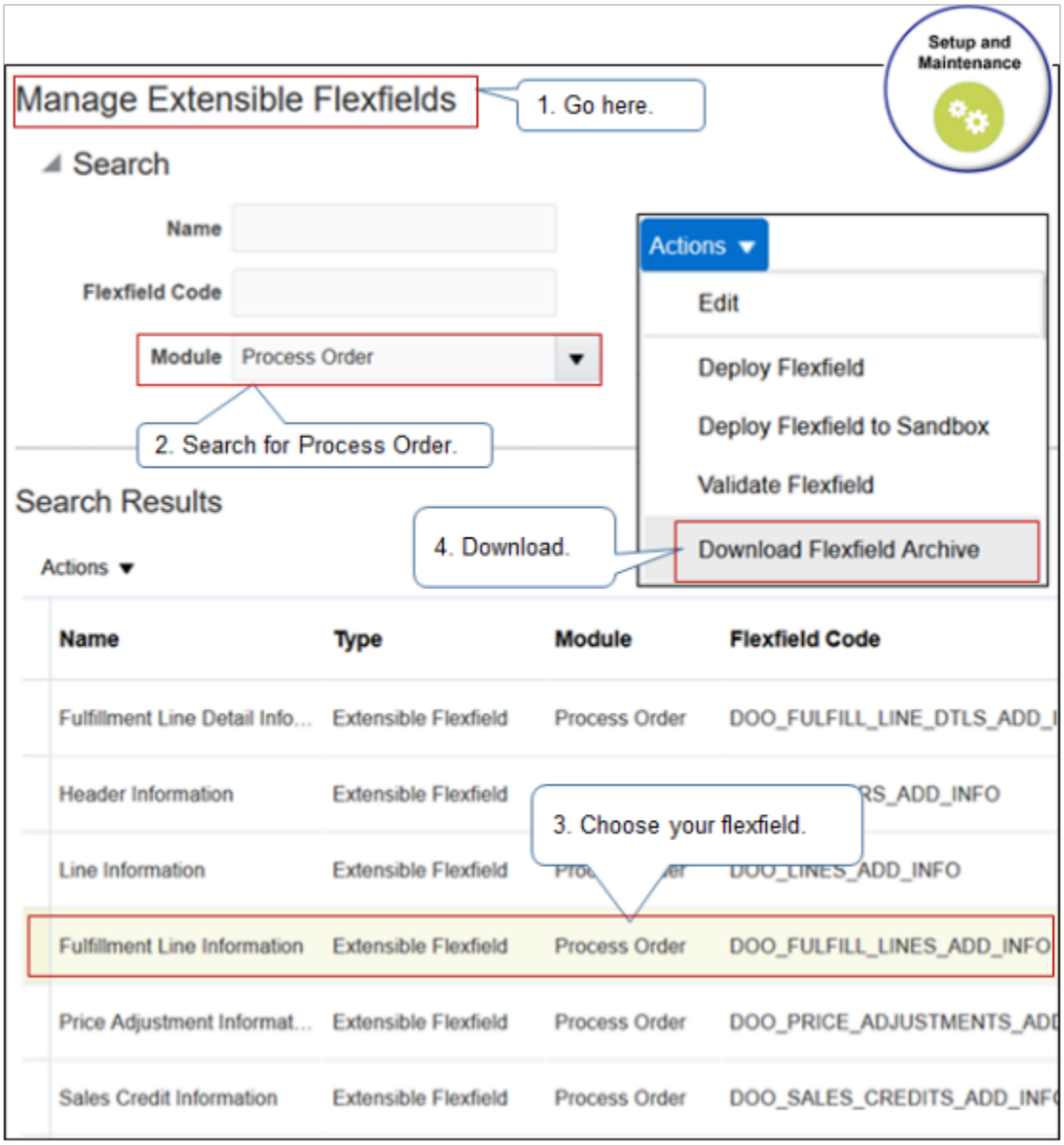
- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Use Extensible Flexfields in Your Integration

Use these guidelines to help you set up an extensible flexfield when you integrate Order Management with another Oracle application.

Get Your Flexfield Archive

The flexfield archive is a zip file that contains an XML file that you download when you create your extensible flexfield. You use details from the flexfield archive when you set up the service mapping. You must get these details now.



Note

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields
2. On the Manage Order Extensible Flexfields page, in the Module attribute, search for Process Order.
3. Select your extensible flexfield.
    - Header Information
    - Fulfillment Line Information
    - Fulfillment Line Detail Information

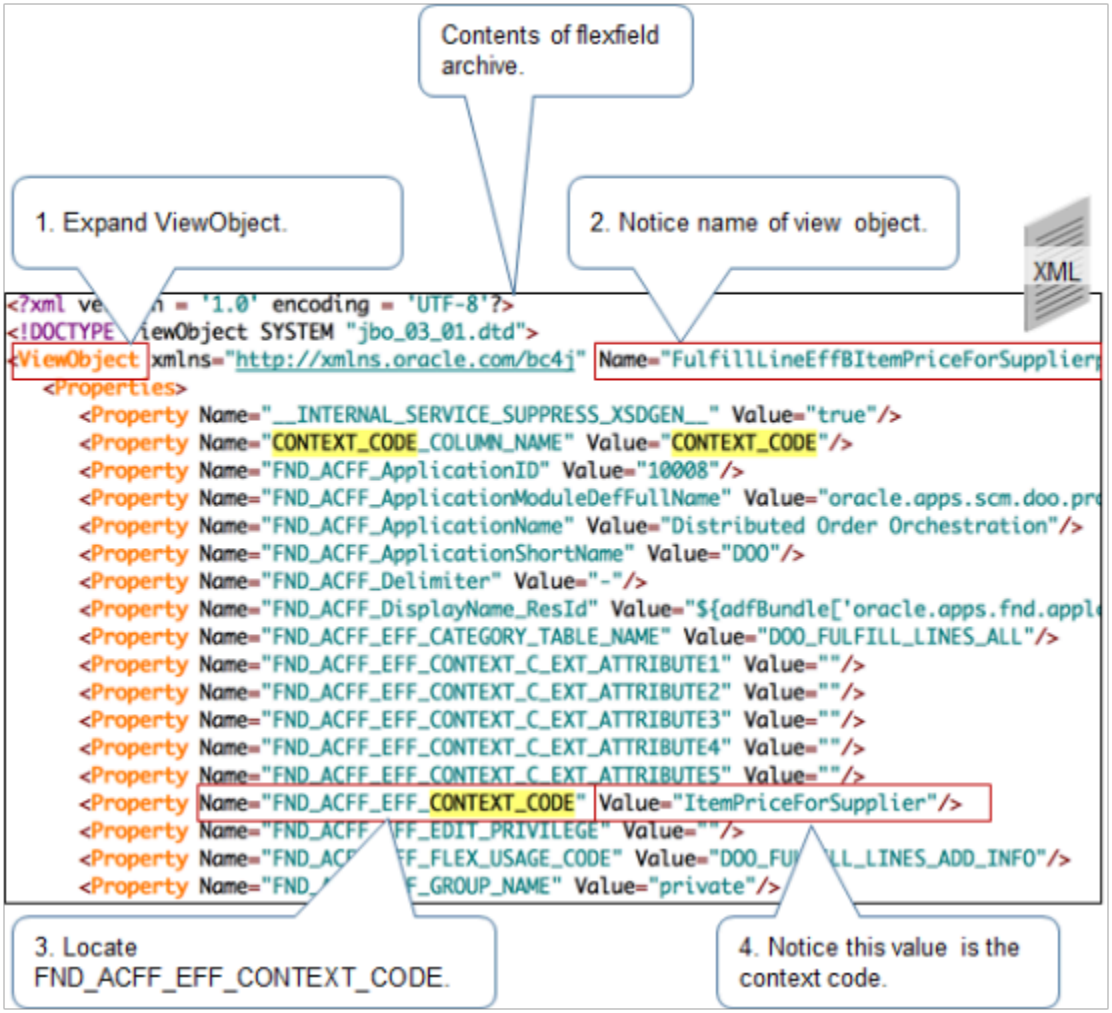
4. Click **Actions > Download Flexfield Archive**.
5. Open the zip file that you downloaded and navigate to oracle/apps/scm/doo/processOrder/flex/myObjectContextsB/view/myVO.xml.

   where

   - myObject is the name of your object.
   - myVO is the name of your virtual object.
6. Open the XML, then copy the contents into a text file.

Have a look at an example that downloads a flexfield archive. For details, see *Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications*.

Examine Flexfield Archive



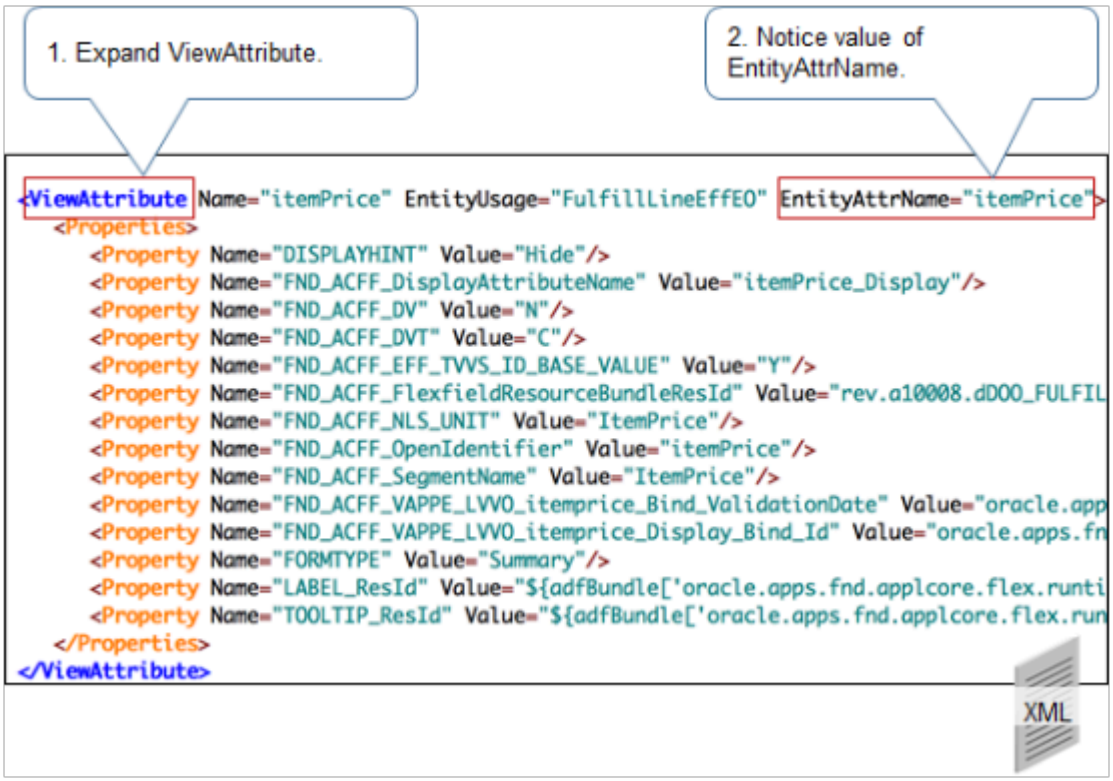
Note

1. Expand **ViewObject**.

2. Notice the value in the Name attribute of the view object. This is the name of the view object that you copy and paste into the View Object attribute of your service mapping.

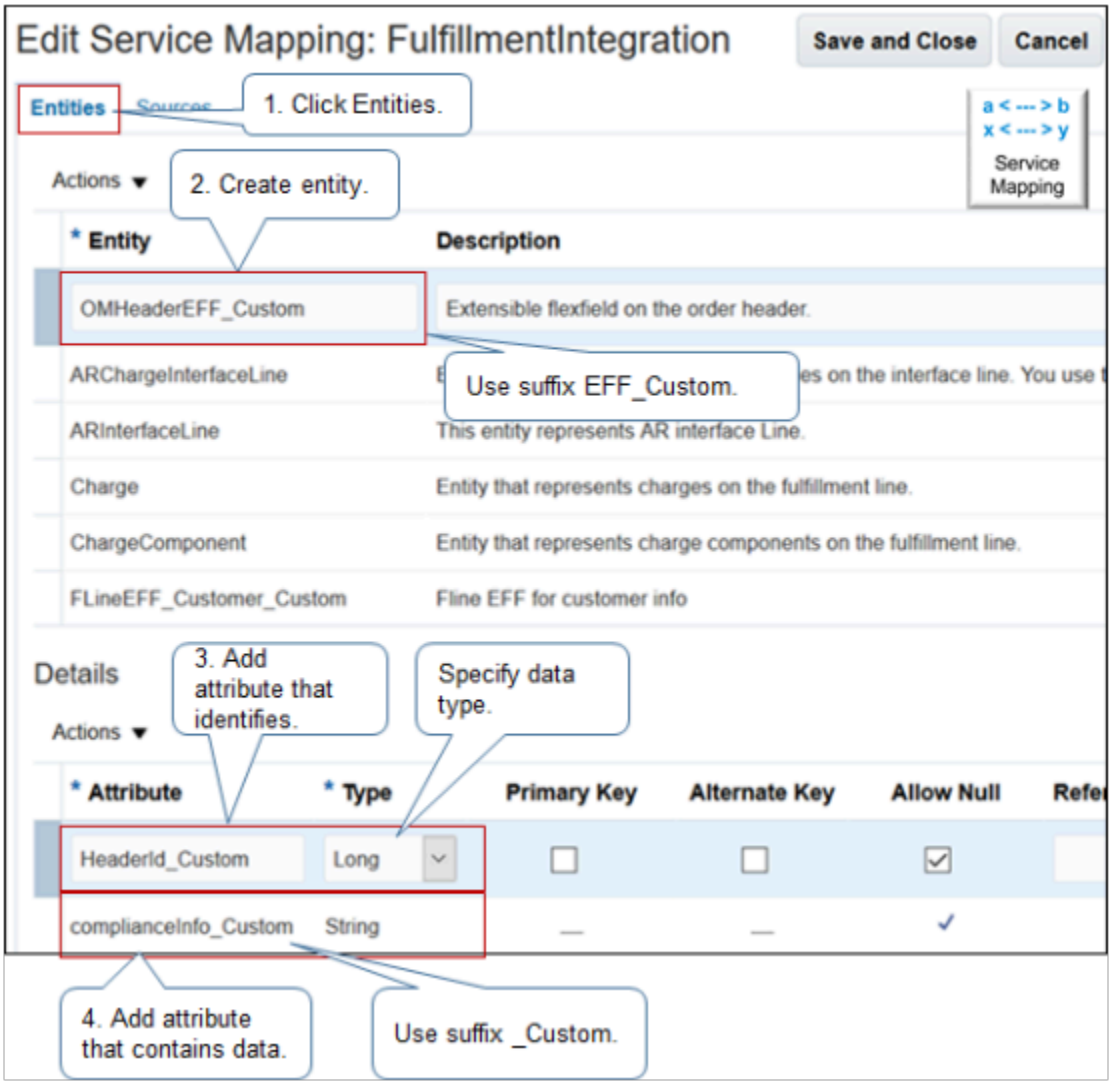
   The name in the screen capture above is truncated. The full name for this example is FulfillLineEffBItemPriceForSupplierprivateVO.

   The name for the example earlier in this topic is HeaderEFFBComplianceDetailsprivateVO.
3. Under ViewObject, locate FND_ACFF_EFF_CONTEXT_CODE.
4. Notice the value of the context code. In this example, the value is ItemPriceForSupplier. You will use it later.
5. Expand **ViewAttribute**.



6. Notice the value of EntityAttrName. This is the name of the view object that you copy and paste into the attribute View Object Attribute of your service mapping. EntityAttrName for the example earlier in this topic would equal `_complianceInfo`.

Specify Entity



Note

1. Navigate back to your service mapping, then click **Entities**.
2. Create one entity for each extensible flexfield context.

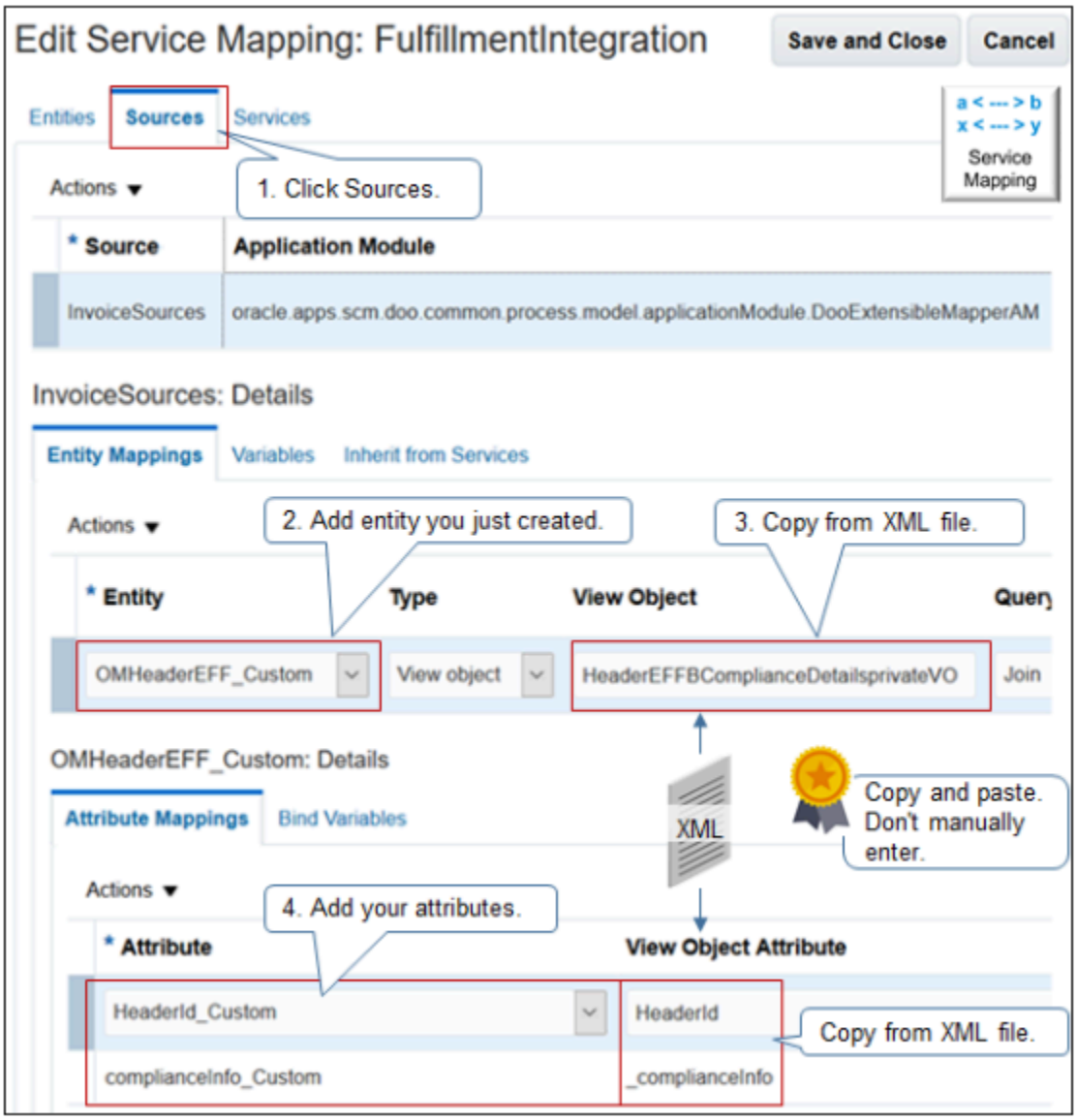
   You must use the EFF_Custom suffix for entity name. For example, OMHeaderEFF_Custom. If you don't use this suffix, then your service mapping will fail.
3. Add an attribute that identifies the entity. In this example, HeaderId identifies the sales order header.
4. Add an attribute that contains data.

Note

- Specify the data type. If you don't specify the correct data type for the attribute, then your service mapping will fail.

  This example uses the Long data type for the HeaderId attribute because identifiers are typically a Long numeric value. It uses String for the `complianceInfo` attribute because, in this example, `complianceInfo` contains text.
- Use the _Custom suffix for attributes.

Specify Source



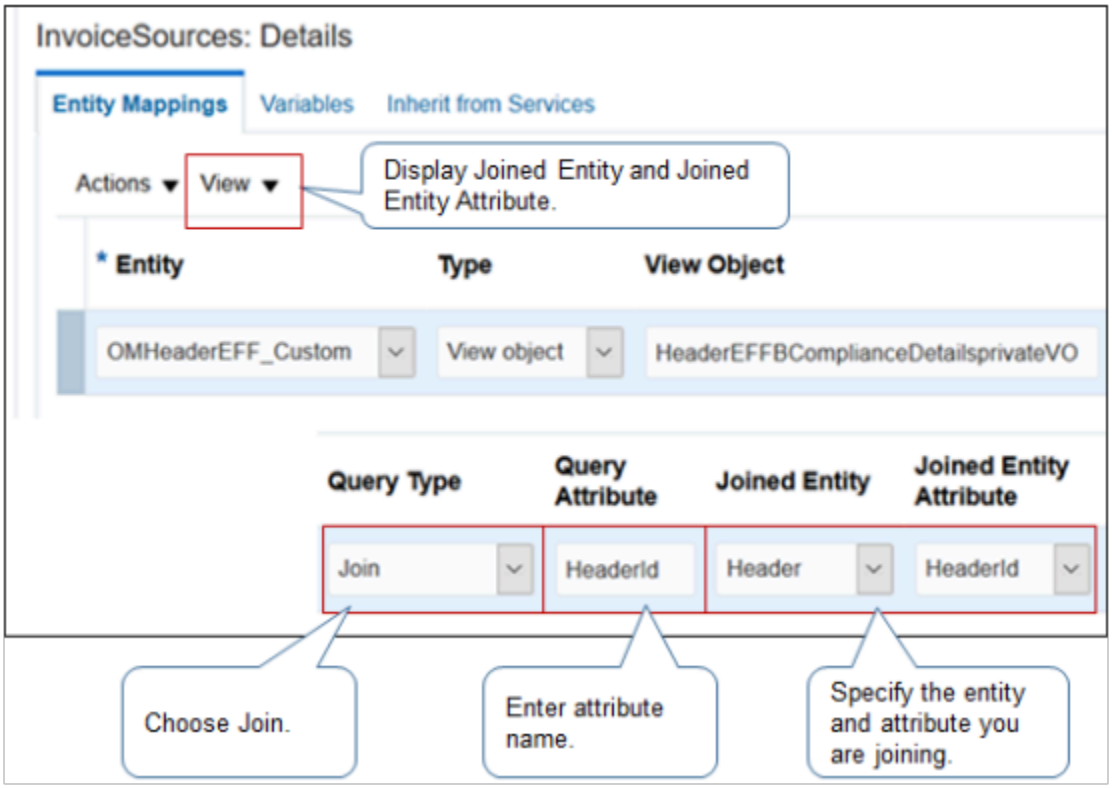
Note

1. Click **Sources**.
2. On the Entity Mappings tab, add your new entity, such as OMHeaderEFF_Custom.

3. Specify the view object.
    - Locate the view object name in the XML of the flexfield archive, copy it, then paste it into the View Object attribute. In this example, the view object name is HeaderEFFBComplianceDetailsprivateVO.
    - The values must match exactly, or your service mapping will fail. To avoid problems, use copy and paste. Don't manually enter it.
    - Add your attributes.

      For each attribute you add, locate the view object attribute in the XML, copy it, and paste it into the attribute View Object Attribute. In this example, notice that the `_complianceInfo` view object attribute begins with an underscore because the XML also contains an underscore.

Add Entity



Here's some tips when you add the entity.

- Before you add the entity, click **View > Joined Entity > Joined Entity Attribute**.
- Set the values.

Attribute	Value
Query Type	Select Join.
Query Attribute	Enter the name of your attribute, such as HeaderId.

Attribute	Value
Joined Entity	Specify the entity you're joining, such as Header.
Joined Entity Attribute	Specify the attribute in the entity you're joining, such as HeaderId.

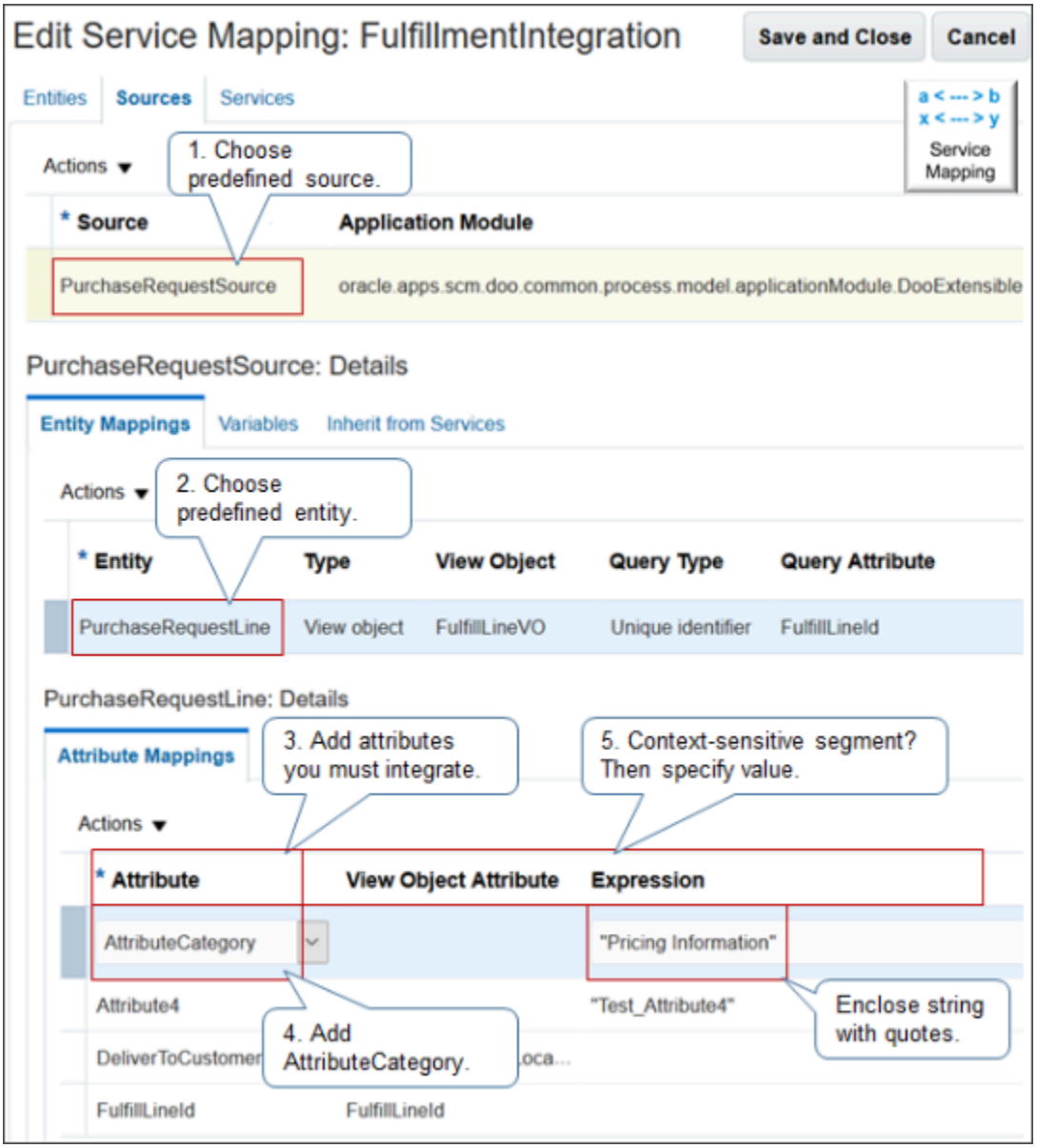
Related Topics

- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Use Descriptive Flexfields in Your Integration

Apply these guidelines when you use a descriptive flexfield to integrate Order Management with another Oracle application.

Understand How You Use a Descriptive Flexfield

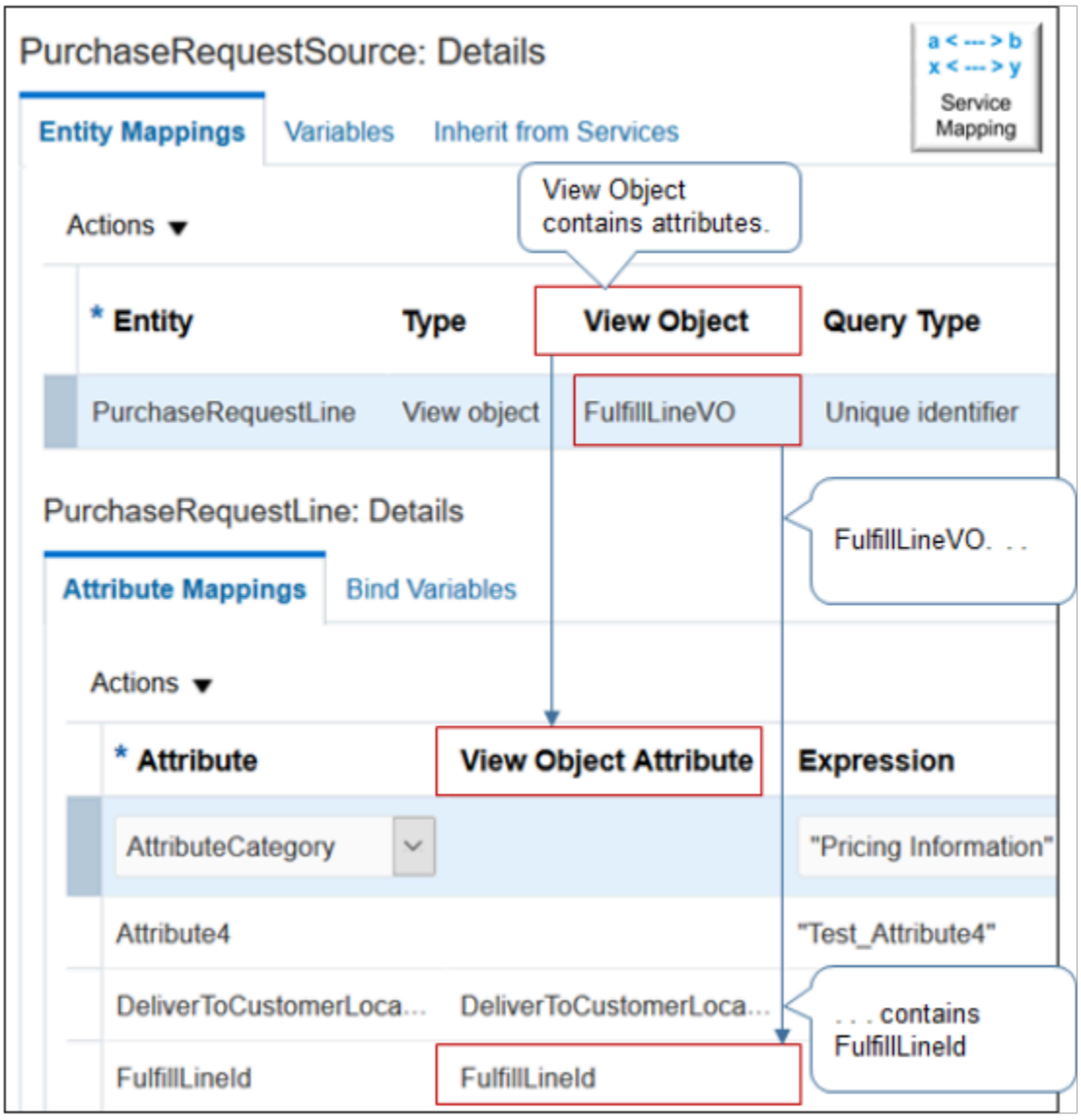


Note

1. To integrate Purchasing, Shipping, or Receiving, select a predefined source, such as PurchaseRequestSource. Order Management provides predefined descriptive flexfields that you can use. You don't need to create one.
2. Select a predefined entity, such as PurchaseRequestLine.
3. Add the attributes that you must integrate with your fulfillment system.
4. Add the AttributeCategory attribute. This attribute contains the value of the context code, such as ItemPriceForSupplier, of the descriptive flexfield.

5. If your descriptive flexfield uses a context sensitive segment.
    - Provide a value.
    - Reference an attribute in a view object or add an expression.
    - You typically add an expression.
    - If you add a string in your expression, then enclose it with double quotation marks (" ").

Reference the Attribute in the View Object

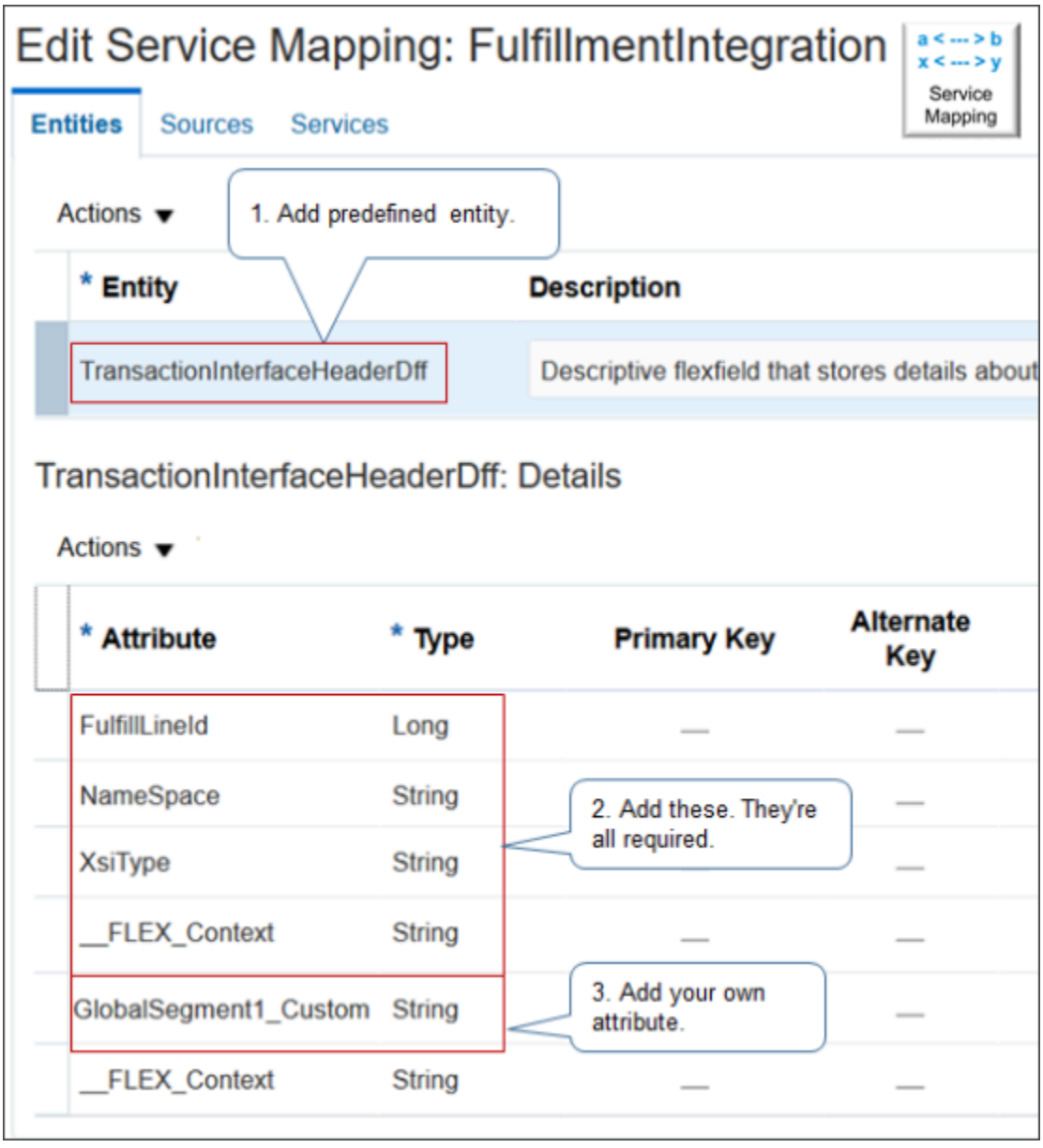


Note

- A view object, such as FulfillLineVO, contains attributes, such as FulfillLineId.
- VO is an abbreviation for view object.

- The value you chose for the View Object attribute determines the values you can select for the attribute View Object Attribute.

Use a Descriptive Flexfield When You Integrate With Accounts Receivable



Note

1. Use a predefined entity for accounts receivable, such as TransactionInterfaceHeaderDff, which is a descriptive flexfield that stores details about the receivables transaction header. Examine the entities that are available on the Entities tab to identify the one you need.

2. Add these attributes. They're all required.
    - FulfillLineId
    - NameSpace
    - XsiType
    - _FLEX_Context

    You must add these attributes for each context entity that you define for a descriptive flexfield.
3. Add the descriptive flexfield attributes that you must send to accounts receivable, such as GlobalSegment1_Custom.

Note

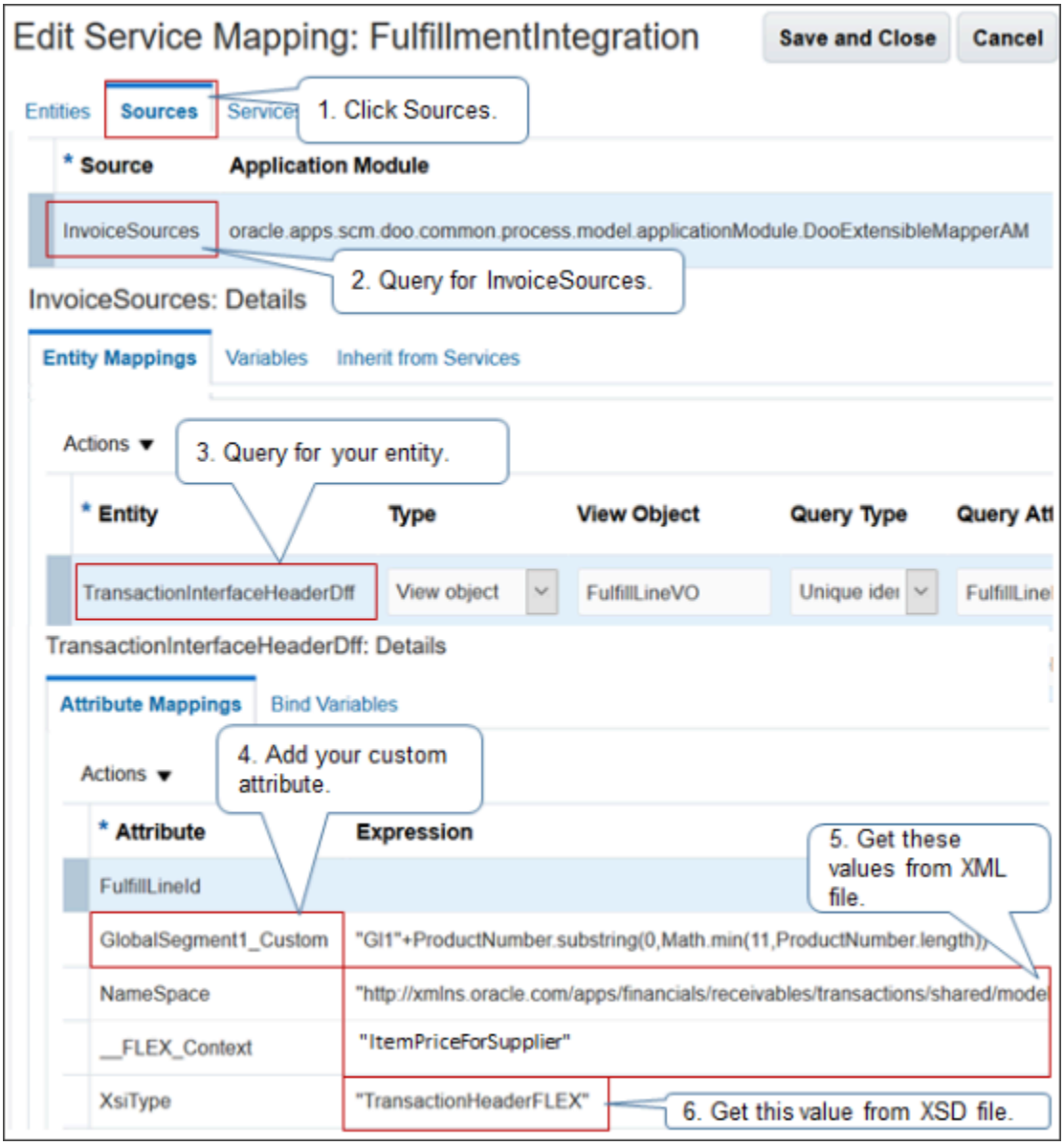
- You can map an interface line of type Line.
- You can't map an interface line of type Discount, Charge, or Freight.
- You can use only one context for each descriptive flexfield.
- To avoid an SQL exception, make sure the web service you use to send fulfillment lines from Receivables to Order Management includes no more than 10,000 fulfillment lines in each invoice response.

If you must map an attribute from the order header, extensible flexfield, or fulfillment line detail, then you must use an integration algorithm. Order Management uses this hierarchy.

```
Order header
 Fulfillment line
 Fulfillment line details
```

Other Oracle applications might not organize these objects into a hierarchy. Instead, they might represent them in a single object named ARLine. ARLine also includes descriptive flexfields, so it can represent the same data that the fulfillment line represents. If you must copy attributes from the fulfillment line to ARLine, then you require an integration algorithm only if you must implement conditional logic.

Specify the Source



Note

1. Click the **Sources** tab.
2. Query the Source for InvoiceSources.
3. Query the Entity for your entity, such as TransactionInterfaceHeaderDff.
4. Add your custom attributes. Create an expression for each of them.
5. Enter a value for NameSpace and _FLEX_Context from the XML file.
6. Enter a value for XSiType from the XSD file.

For example:

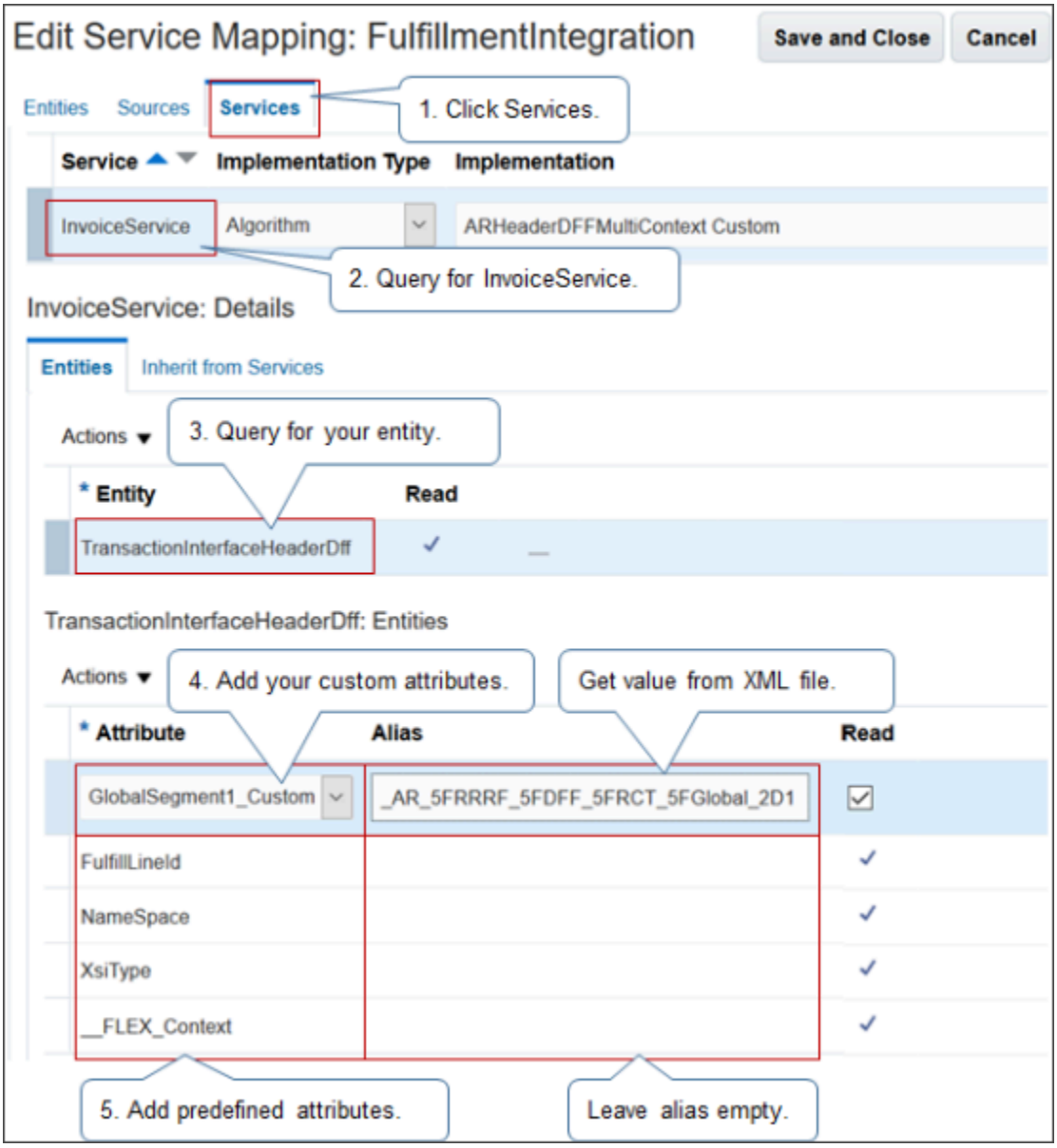
Attribute	Expression
GlobalSegment1_Custom	As an option, add an expression. For example:

Attribute	Expression

	`"G11"+ProductNumber.substring(0,Math.min(11,ProductNumber.length))`
	
	Note that this example is a concatenation.
	
	Here's the pseudocode.
	
	`Send the product number. If the product number is a number, then convert it to a string. Add the length of the string.`

NameSpace	Enter the value from the XML file you downloaded for your flexfield archive. For example: `"http://xmlns.oracle.com/apps/financials/receivables/transactions/shared/model/flex/TransactionHeaderDff/"`
XsiType	Enter the value from your XSD file. For example: `"TransactionHeaderFLEX"`
_FLEX_Context	Enter the value of the context code from the XML file you downloaded for your flexfield archive. This example uses GlobalSegment1, so you can leave _FLEX_Context empty. If you didn't include a global segment in this example, you would enter `"ItemPriceForSupplier"`.

Specify the Service



Note

1. Click the **Services** tab.
2. Query the Service for InvoiceService.
3. Query the Entity for your entity, such as TransactionInterfaceHeaderDff.
4. Add your custom attributes. Create an alias for each one. The alias is the name of the service. Get it from your XML file.

Attribute	Alias
GlobalSegment1_Custom	_AR_5FRRRF_5FDFF_5FRCT_5FGlobal_2D1 Order Management uses this value when it creates the payload to create the task.

Attribute	Alias
	If you prefer to send this value in your integration algorithm, then you add it to the algorithm instead.

5. Add the predefined attributes. It isn't necessary to define the alias for predefined attributes.

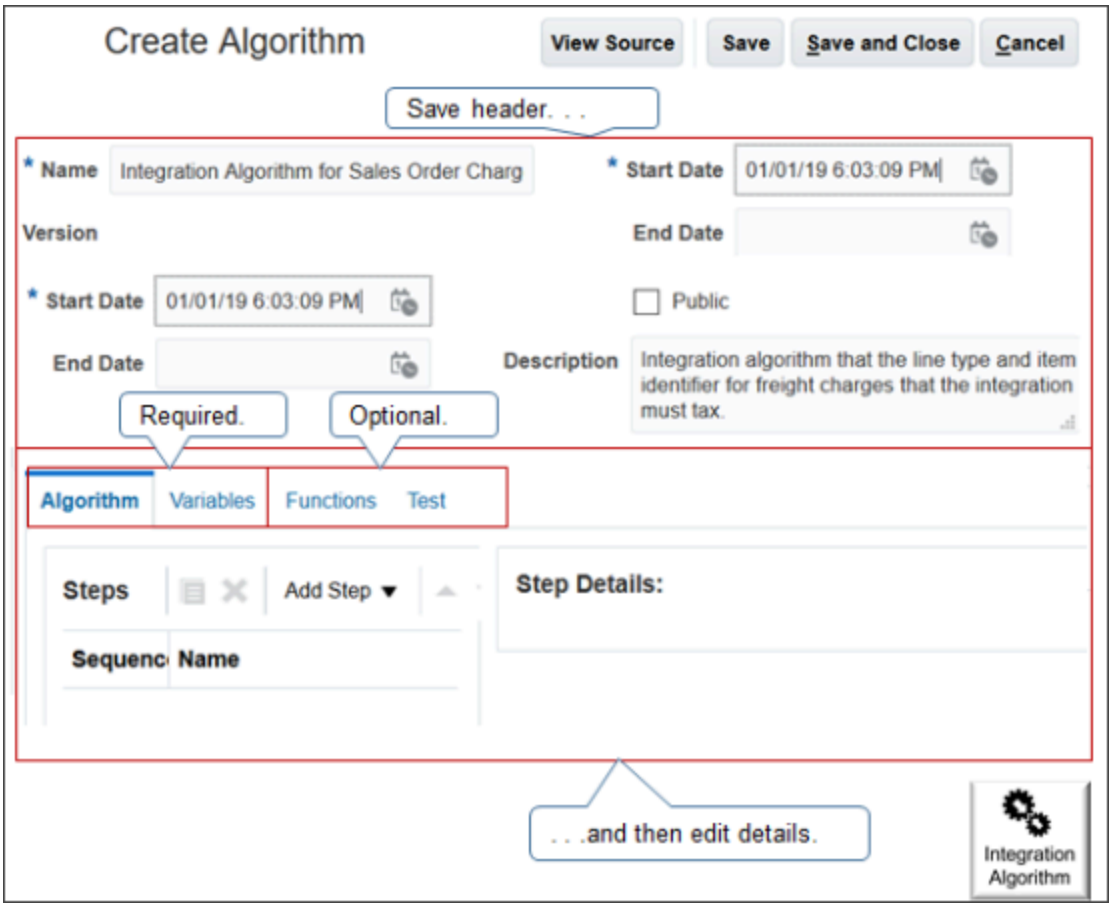
Related Topics

- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Set Up an Integration Algorithm

Apply guidelines when you use an integration algorithm to integrate Order Management with another Oracle application.

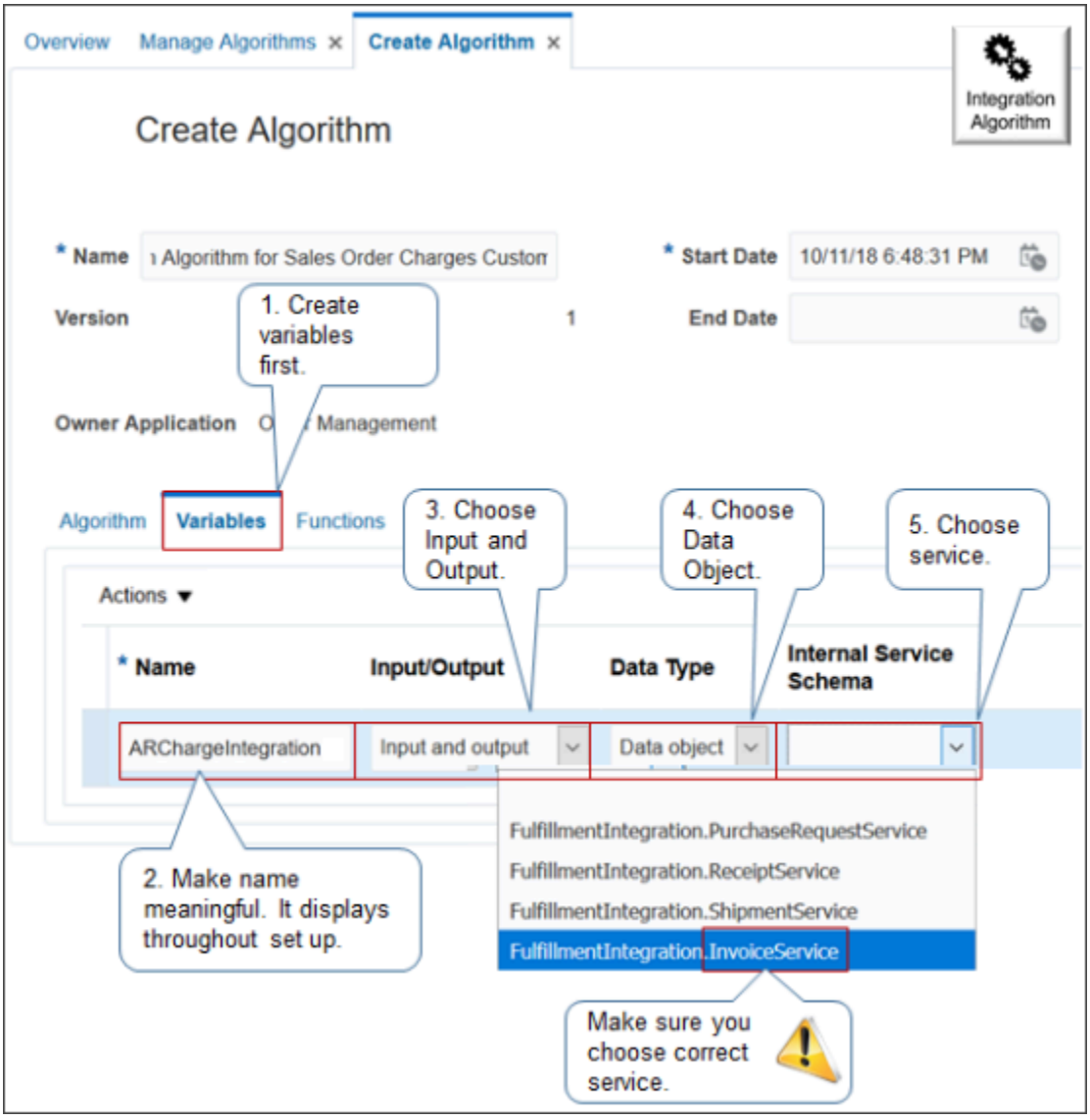
Create Algorithm



Note

- When you create the algorithm, save the header before you work on details.
- You must set up the algorithm and variables.
- Functions are optional.
- Test is a design time test. It isn't a run time test. It's optional, but we recommend it.

Use Variables

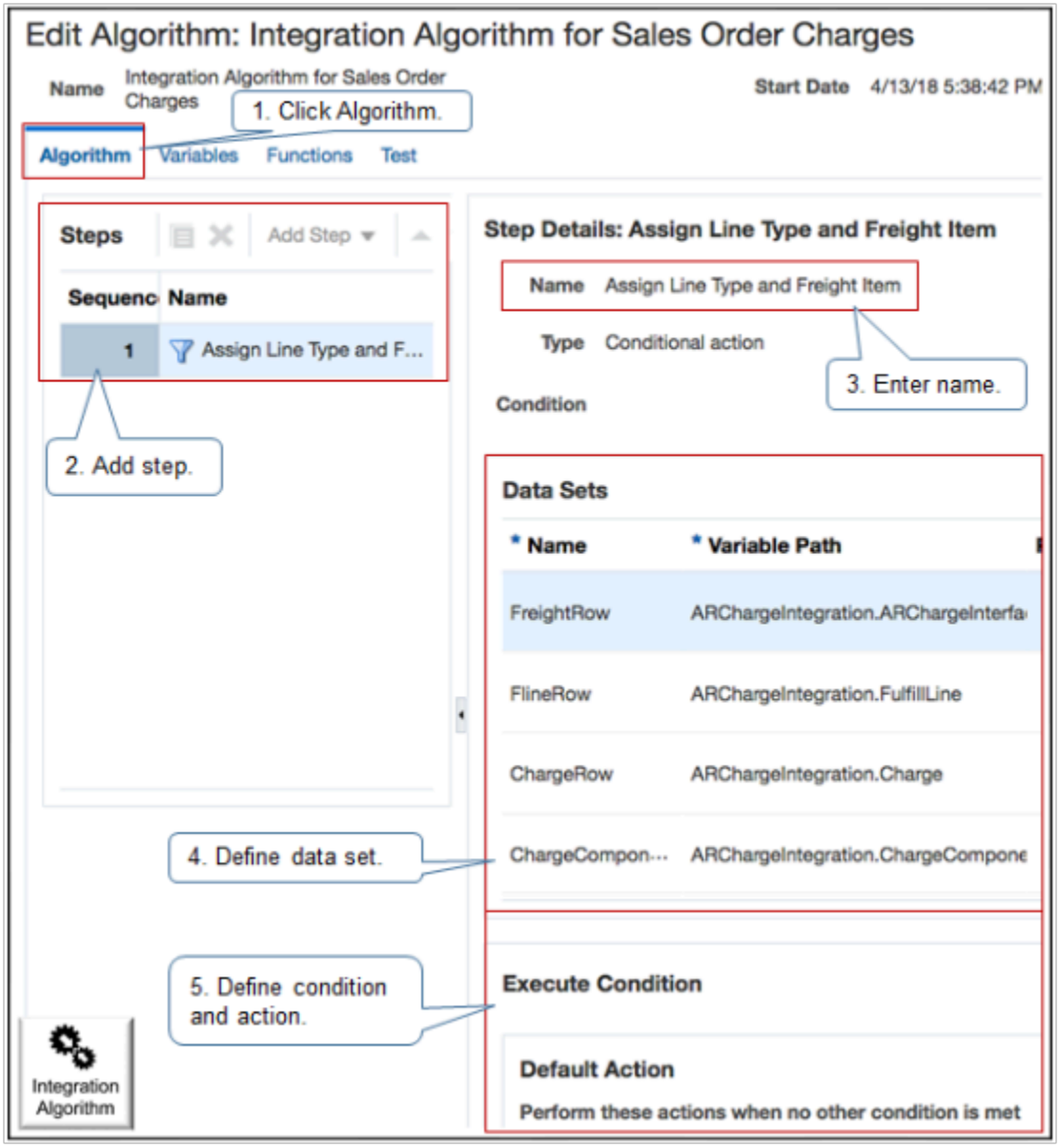


Note

- Create and save the header first, and then set up the variables. Don't define the algorithm yet because you use variables in the algorithm. You must set up the variables first.
- Click **Variables**.
- Set attributes.

Attribute	Value
Name	Enter a meaningful name because the value you enter will display throughout the set up.
Input/Output	Select Input and Output.
Data Type	Select Data Object.
Internal Service Schema	Use FulfillmentIntegration.YourService where o YourService specifies the service that your integration requires. This example uses FulfillmentIntegration.InvoiceService **CAUTION:** Its important to select the correct service. If you select the wrong service, the integration will fail.

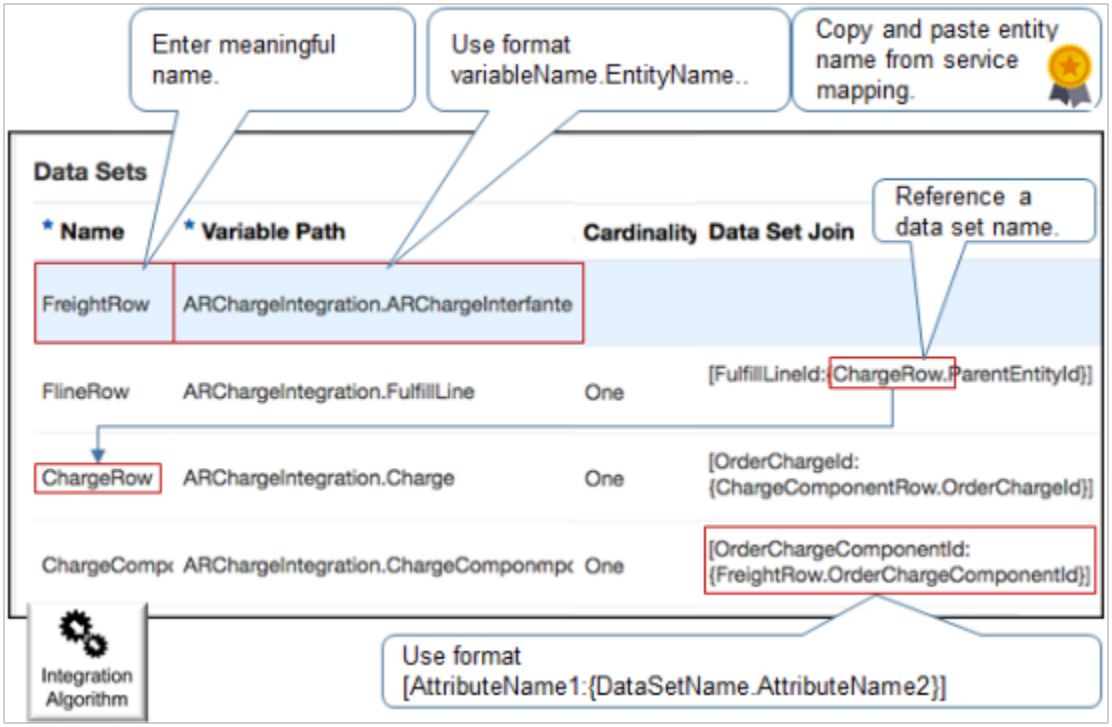
Add Step



Note

1. Click **Algorithm**.
2. Add steps. At run time, the algorithm runs the steps sequentially.
3. Enter a name.
4. Set up the data set that determines which records to process in this step.
5. Set up the condition and action that determines how to process records.

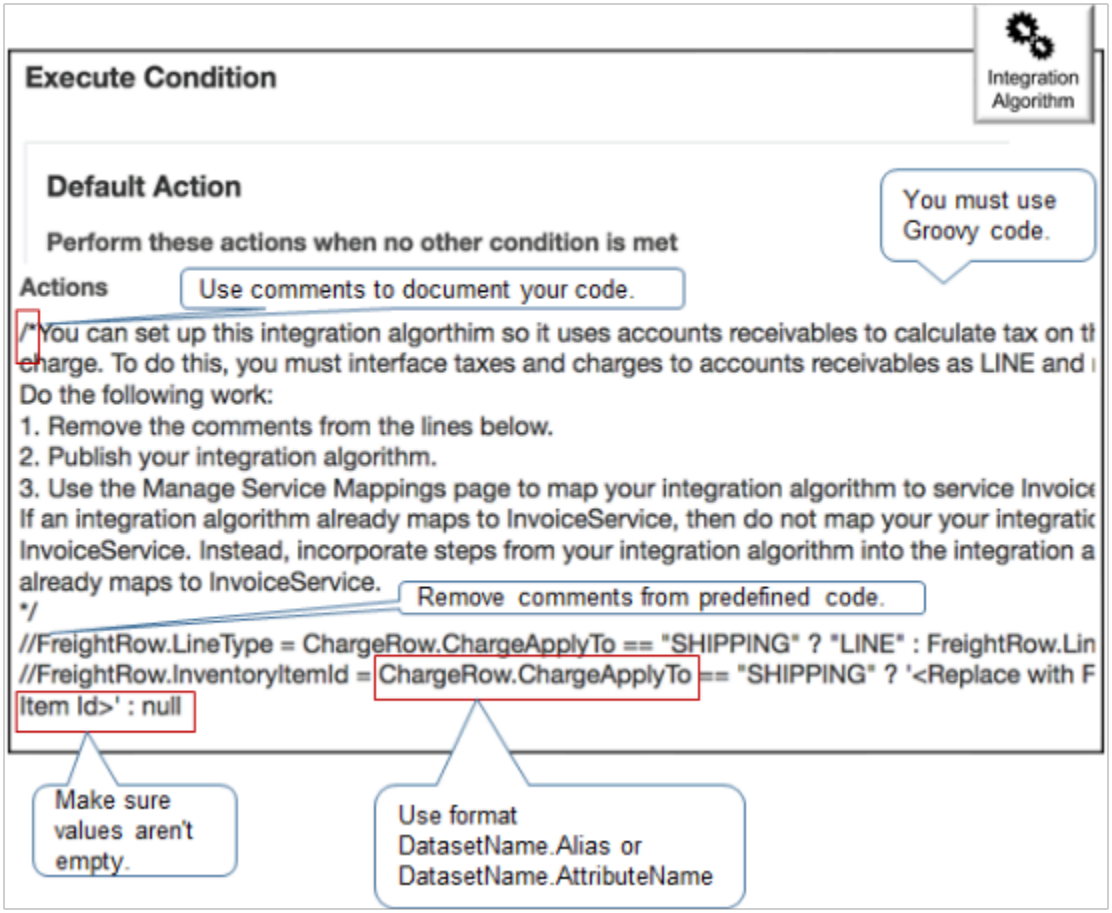
Create Data Set



Specify the attributes.

Attribute	Description
Name	Enter a meaningful name. Algorithm logic will reference it.
Variable Path	Use the VariableName.entity format. where • **`VariableName`**. Name of the variable you defined on the Variables tab, such as ARChargeIntegration. • **`Entity`**. Copy and paste the entity name from your service mapping.
Cardinality	For descriptive flexfield, you typically select **Zero or One**. If you join many order lines to one order header, then select **Many**.
Data Set Join	Use format **`[AttributeName1:{DataSetName.AttributeName2}]`** where • **`AttributeName1`**. Name of an attribute in the data set you're joining to. • **`DataSetName`**. Name of a data set you have defined. • **`AttributeName2`**. Name of an attribute in DataSetName.

Specify Condition and Action



Note

- The algorithm runs the default action when algorithm logic doesn't meet any other conditions.
- This predefined action interfaces freight charges as a separate line item on the invoice.
- Predefined code is commented. Remove comments to enable the action.
- Write your code in Groovy script.
- Use comments to document your code.
- Make sure the variable isn't empty before you use it as the source for a value. In particular, make sure your extensible flexfield contains a value. If it doesn't contain a value, then you can't get a value from it, and your algorithm might fail.
- Use format **`DatasetName.Alias`**.

  where

  - **`DatasetName`**. Name of the data set that you defined in the integration algorithm.
  - **`Alias`**. Alias or attribute name that you defined in your service mapping.

    For example, **`ChargeRow.ChargeApplyTo`**.

For details, including the exact content of the Actions that this example uses, see *Overview of Integrating Order Management with Accounts Receivable*.

Specify Attributes Between Service Mapping and Integration Algorithm

Note

- You must make sure the attribute names that you use in the service mapping and the attribute names that you use in the integration algorithm match each other exactly.
- If you specify an alias in the service mapping, then also use this alias in the integration algorithm.
- If you don't specify an alias, then use the attribute name that the integration algorithm uses.
- Make sure the attribute names use the same upper case and lower case.

For example:

Setup	Correct	Not Correct
You set up an attribute named TotalSale_Custom in the Sources tab of the service mapping, and you don't define an alias in the Services tab of the service mapping.	This example code won't cause an error. `Hdr.Attribute3 = HdrEFF.TotalSale_Custom`	This example code will cause an error. `Hdr.Attribute3 = HdrEFF.totalSale_Custom`
You set up TotalSale_Custom in the Sources tab of the service mapping, and you set up an alias as TotalSale in the Services tab.	This example code won't cause an error. • `Hdr.Attribute3 = HdrEFF.TotalSale_Custom` • `Hdr.Attribute3 = HdrEFF.TotalSale`	This example code will cause an error. • `Hdr.Attribute3 = HdrEFF.totalSale_Custom` • `Hdr.Attribute3 = HdrEFF.totalSale`

Here's a message that's an example of the type of error you might encounter at run time. In this example, the Sources tab of the service mapping includes the totalSale_Custom attribute, but code in the integration algorithm uses TotalSale_Custom.

```
01.03:llLine><PurchaseRequestService:FulfillLineId>300000002330472</
PurchaseRequestService:FulfillLineId><PurchaseRequestService:HeaderId>300000002330464</
PurchaseRequestService:HeaderId> </PurchaseRequestService:FulfillLine> <PurchaseRequestService:ChangeSummary
xmlns:sdo="commonj.sdo"/></PurchaseRequestService:PurchaseRequestServiceType>'-
oracle.apps.scm.pricing.priceExecution.algorithms.publicQuery.exception.SetQueryException:
Step 'Step 1' not executed properly. Failed to execute onEach Closure. DataObject
com.oracle.xmlns.apps.scm.pricing.priceexecution.servicemappings.publicmappings.purchaserequestservicetype.HeaderEff_Cus
does not have property 'TotalSale_Custom' defined in the schema. Can not get value from the property.
null Payload: VariableName:'PRCIntegration'DateType:'commonj.sdo.DataObject' IOType:'InOut' value:'<?xml
version="1.0"encoding="UTF-8"?> <PurchaseRequestService:PurchaseRequestServiceType xmlns:PurchaseRequestService
```

1. Go to the Pricing Administration work area.
2. Click **Tasks**, then under Order Management Configuration, click **Manage Service Mappings**.
3. On the Manage Service Mappings page, click **FulfillmentIntegration**.
4. On the Edit Service Mapping page, in the Entity list, click the **line** that has the value.

Attribute	Value
Entity	PurchaseRequestLine

Attribute	Value

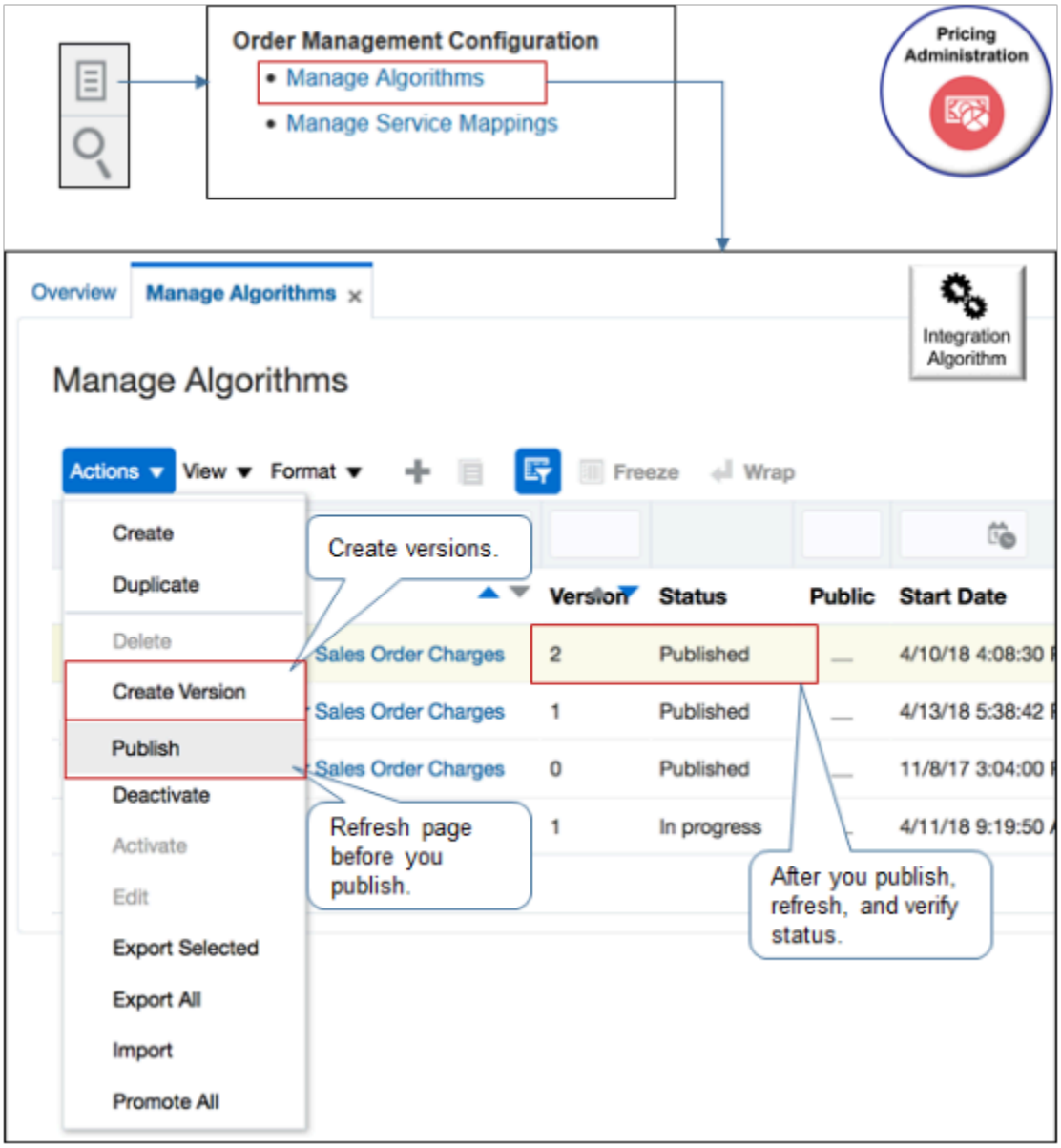
5. In the Details list, click **View > Columns**, then add a check mark to Description.

   Examine the description for the attribute you want to map.

If the Description Says	Then
`This attribute is available with the create and with the update operation`	You can use PurchaseRequestService to update the attribute.
`This attribute is available only with the create operation`	You can't use PurchaseRequestService to update the attribute.

Publish and Test

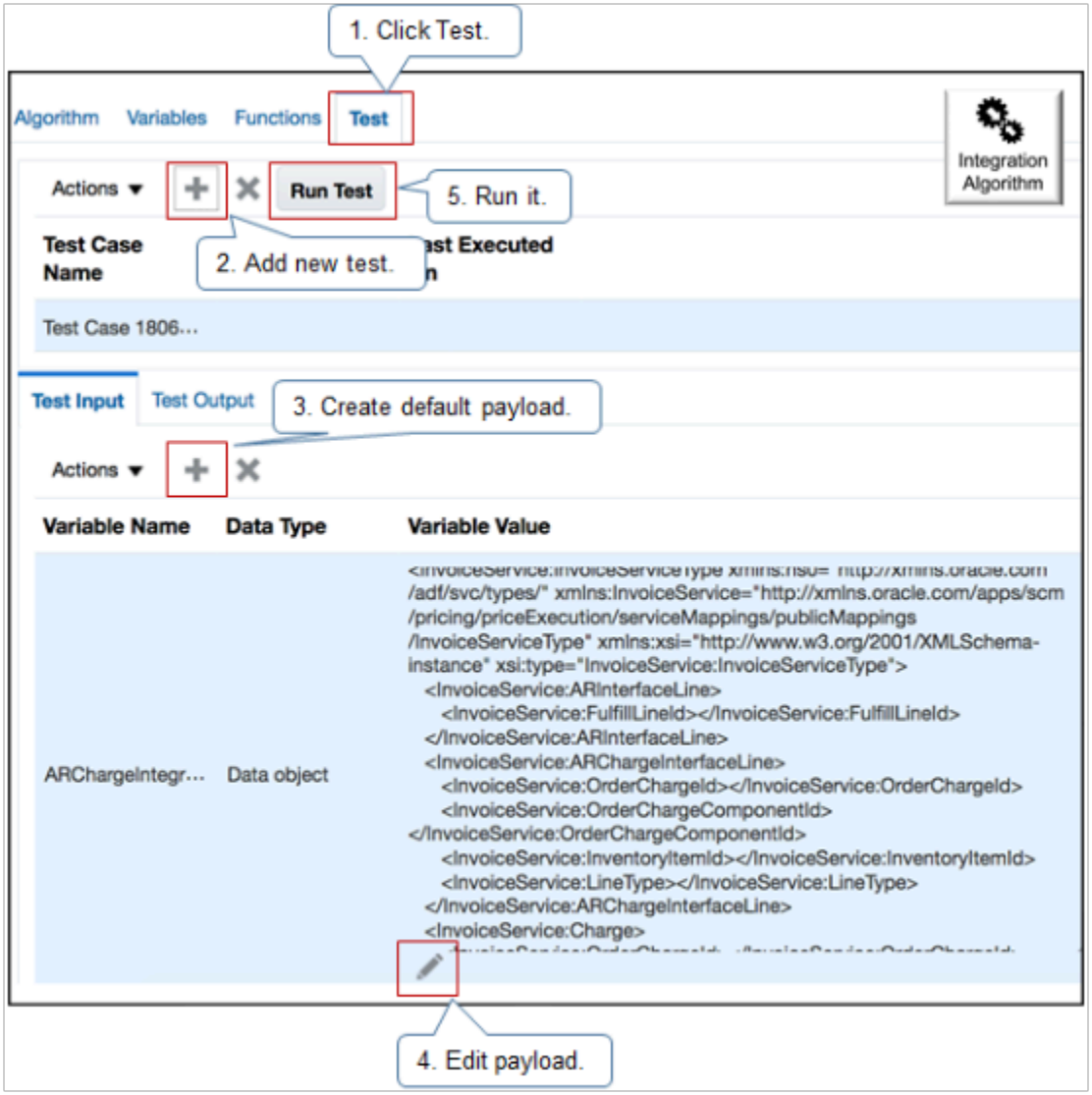
Publish your algorithm.



Note

- . Go to the Pricing Administration work area.
- Click **Tasks**, then under Order Management Configuration, click **Manage Algorithms**.
- Create different versions so you can test different set ups while maintaining a working copy.
- You must publish your algorithm.
- Before you publish, refresh the page: save, close the page, open the page, and requery algorithms.
- After you publish, refresh, and verify that the status is Published.
- You can't edit a published algorithm. You must create a new version instead, or deactivate, and then activate.

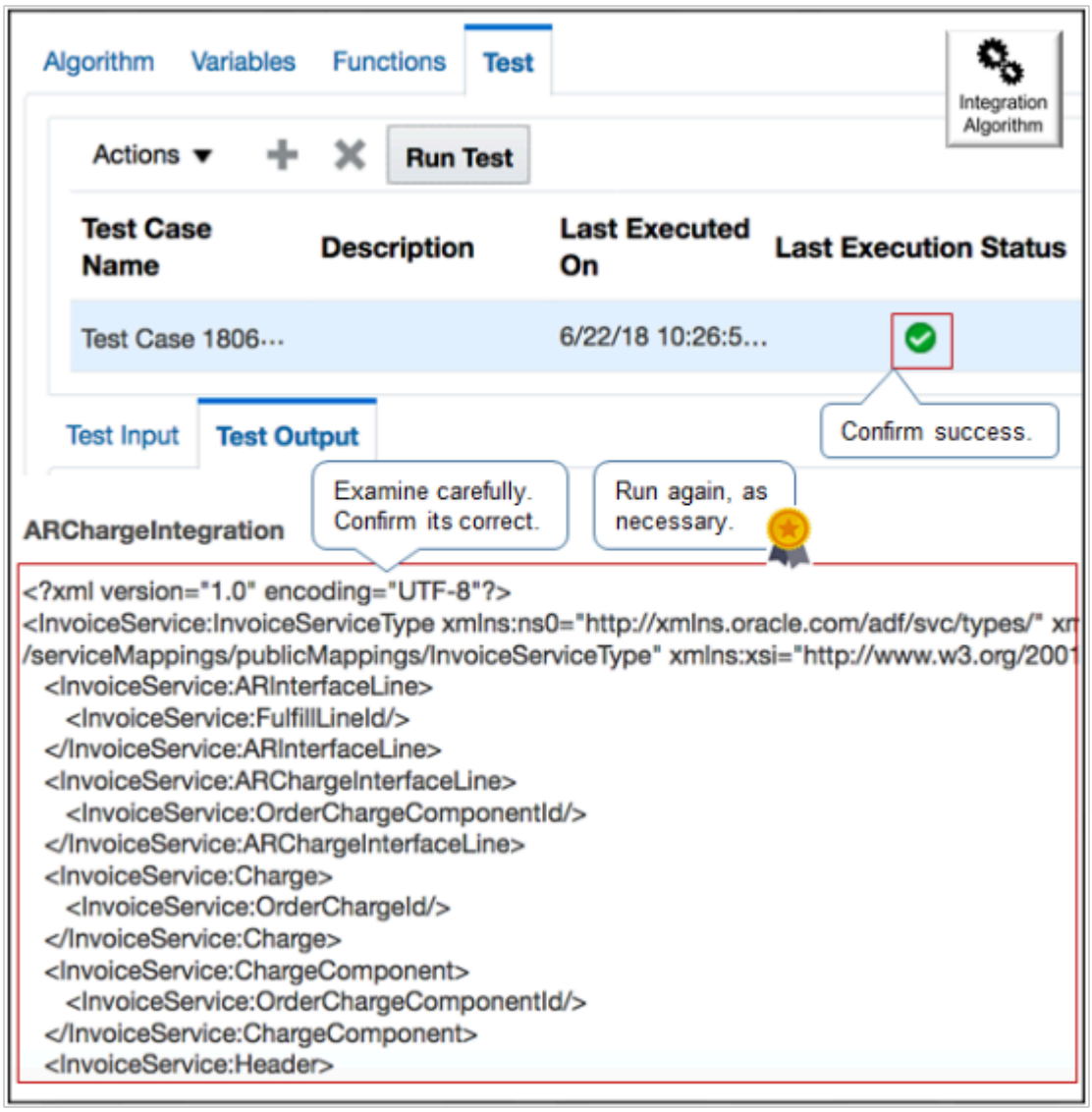
Test your algorithm.



Note

1. Click **Test** to run a design time test.
2. Add a new test.
3. Create a default payload. The test creates a default payload according to the structure you set up in the integration algorithm.
4. Edit the payload so it meets your requirements. For example, add values for important attributes that you know your fulfillment system expects.
5. Click **Run Test**.

6. Examine the results.



Note

- Notice value of Last Execution Status. Green check mark means success.
- Examine test output carefully. Make sure output not only exists, but that it contains values you expect from your integration. For example, if you logic concatenates an item description, then make sure the test displays concatenated values that you expect.
- If you modify logic or a data set, then delete your test data, recreate test data, and run test again.
- You must submit your test sales orders from the same sandbox that you use for your service mapping. An integration algorithm doesn't require a sandbox but a service mapping does.
- Test the business unit that you use to submit the sales order.
- Verify attribute values throughout the sales order lifecycle, including before you submit it and in fulfillment views up through the point where the sales order is closed.

Related Topics

- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Procedures

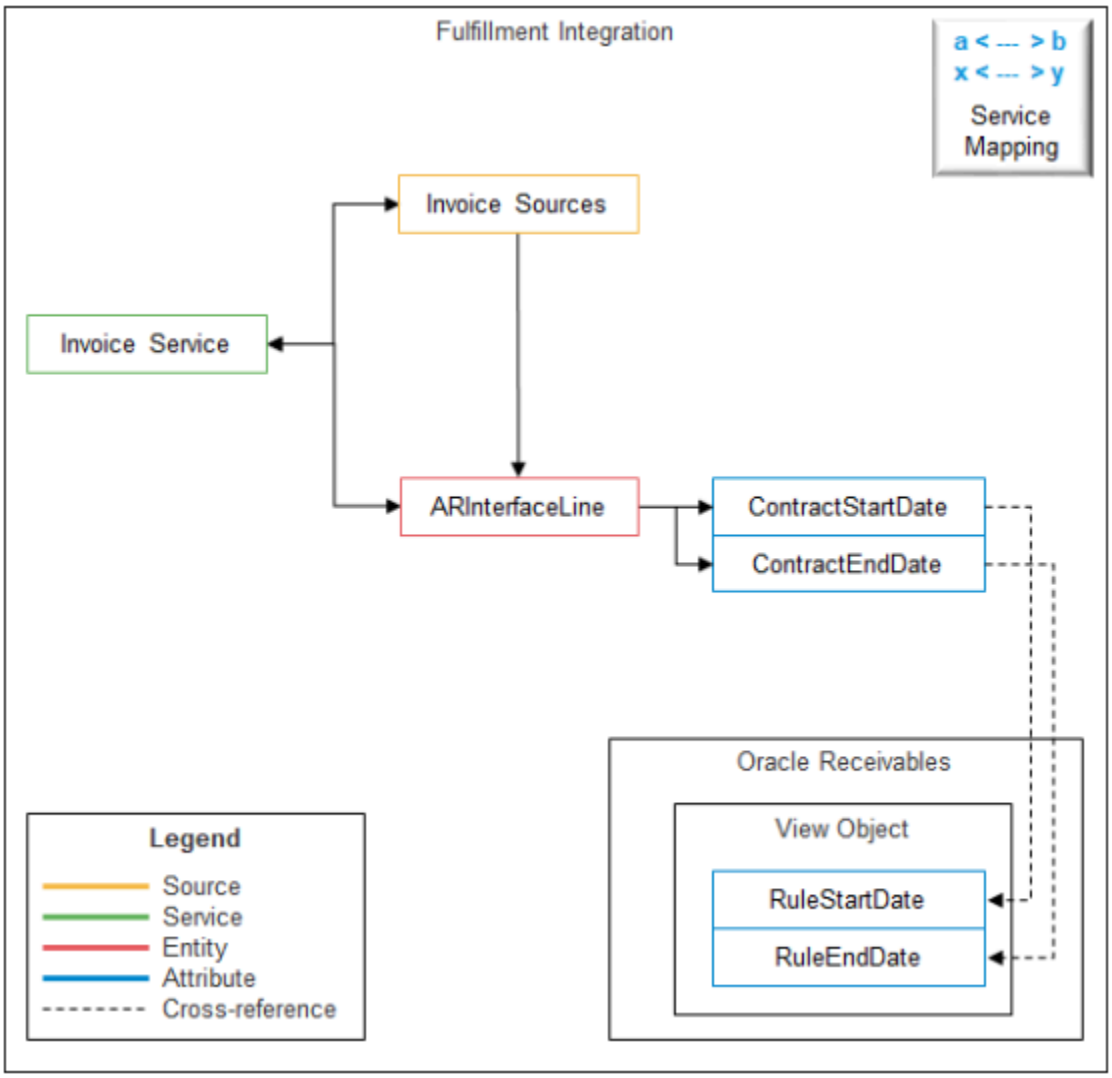
Use a Service Mapping to Integrate Order Management with Other Oracle Applications

Set up Order Management so it uses a view object to integrate with Oracle Receivables, Oracle Shipping, Oracle Receiving, or Oracle Procurement.

Assume you must.

- Integrate ContractStartDate in Order Management with RuleStartDate in Oracle Receivables
- Integrate ContractEndDate in Order Management with RuleEndDate in Oracle Receivables

You will modify the predefined FulfillmentIntegration service mapping so it uses these cross-references.



This topic uses example values. You might need different values, depending on your business requirements.

Try it.

1. Make sure you have the privileges that you need to administer Order Management.
2. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Go to the Pricing Administration work area.

   You use the Pricing Administration work area to set up an integration algorithm, but you aren't setting up pricing. Order Management leverages the Pricing Administration work area because an integration algorithm's structure and logic is very similar to a pricing algorithm. If you sign in with:

   - **Privileges That Allow You to Administer Oracle Order Management** . The Pricing Administration work area displays only integration algorithms.
   - **Privileges That Allow You to Administer Oracle Pricing**. the Pricing Administration work area displays only pricing algorithms.
4. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.

5. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.

   Notice the entities, such as ARInterfaceLine, and attributes, such as AccountRuleDuration. The FulfillmentIntegration service mapping comes predefined with these entities and attributes so you can use them to add attributes when you integrate Order Management with these Oracle applications.

Oracle Application	Example Entities
Oracle Receivables	ARInterfaceLine
Oracle Shipping	ShipmentRequestHeader ShipmentRequestLine
Oracle Receiving	ReceiptAdvice ReceiptAdviceLine ReceiptAdviceLineLot ReceiptAdviceLineLotWithSerials
Oracle Procurement	PurchaseRequestHeader PurchaseRequestLine PurchaseRequestConfig

6. Click **Sources**.

   In this example, you examine the entity or the view object that defines the view object as FulfillLineVO, and then map an attribute of this entity or view object from the fulfillment line.

   Note that Order Management uses this hierarchy.

```
Sales order header
 fulfillLine
```

```
Fulfill_line Details
```

ARLine is on the same level as fulfillLine.

The Entity Mappings tab lists entities, each entity references a view object, and each entity on the Attribute Mappings tab also references one or more attributes. You will create a map that references one of these entities from this view object without using an integration algorithm.

However, if you must map an attribute from some other entity that doesn't reside at the same hierarchy level, then you must use an integration algorithm. For example, if an attribute references an extensible flexfield from the order header or the fulfillment line, then you must use an integration algorithm.

Other Oracle Applications use a similar hierarchy. For example, here's that hierarchy that Purchasing uses for a purchase order.

```
Purchase order
 fulfillLine
 purchaseLine
```

Here's the hierarchy that Purchasing uses for shipping.

```
Shipping header
 shipping line
 fulfillLine
```

Here's the hierarchy that Purchasing uses for the receiving receipt.

```
Receipt header
 ReceiptLine
 FulfillLine
 Lot or lot serial details
```

7. In the InvoiceSources Details area, click the **row** that contains ARInterfaceLine in the Entity attribute.
8. In the ARInterfaceLine Details area, add values, then click **Save**.

Attribute	View Object Attribute
RuleStartDate	ContractStartDate
RuleEndDate	ContractEndDate

The RuleStartDate and RuleEndDate are now integrated to reference the view objects that you specified. Here's what Order Management does at run time.

- Get the value of the ContractStartDate attribute from the Order Management work area, then display it in the RuleStartDate attribute in Oracle Receivables.
- Get the value of ContractEndDate from the Order Management work area, then display it in the RuleEndDate attribute in Oracle Receivables.

You can use Expression Language (EL) in the Expression column to implement logic or a constant value. For example, you can implement this logic.

- If the value of variable x is greater than the value of variable y, then populate an attribute.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Pricing Algorithms
- Service Mapping

Use Different Line Types

Order Management comes predefined to send the ORA_BUY line type to Pricing Administration. If you use any other line type in your price list, then you must specify it on your service mapping.

Assume you:

- Add an item to a price list in Pricing Administration and set the item's Line Type to any value that isn't ORA_BUY on the price list. In this example, assume you set it to MY_LINE_TYPE_1.
- Use the predefined Sales service mapping.

Here's what you need to do.

1. Go to the Pricing Administration work area, then click **Tasks > Manage Service Mappings**.

   For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.
2. On the Manage Service Mappings page, query for, then open the Sales service mapping.
3. On the Edit Service Mappings page, click **Sources**, then click the **row** that has OrderCatalogLine in the Source column.
4. In the OrderCatalogLine Details area, on the Entity Mappings tab, click the **row** that has Line in the Entity column.
5. On the Attribute Mappings tab, notice the row that has this value.

Attribute	Expression
`LineTypeCode`	`(LineTypeCode.equals('ORA_BUY'))?'ORA_BUY':(LineTypeCode.equals('ORA_RETURN')||LineTypeCode.equals('ORA_CREDIT_ONLY')||LineTypeCode.equals('ORA_CANCEL')?'ORA_RETURN':'ORA_BUY')`

6. Add your line type to the expression (the bold text):

Attribute	Expression
`LineTypeCode`	`(LineTypeCode.equals('ORA_BUY'))?'ORA_BUY':(LineTypeCode.equals('ORA_RETURN')||LineTypeCode.equals('ORA_CREDIT_ONLY')||LineTypeCode.equals('ORA_CANCEL')?'ORA_RETURN':(LineTypeCode.equals('MY_LINE_TYPE_1'))?'MY_LINE_TYPE_1':'ORA_BUY')`

How Come the Price on the Catalog Line and the Order Line Aren't the Same?

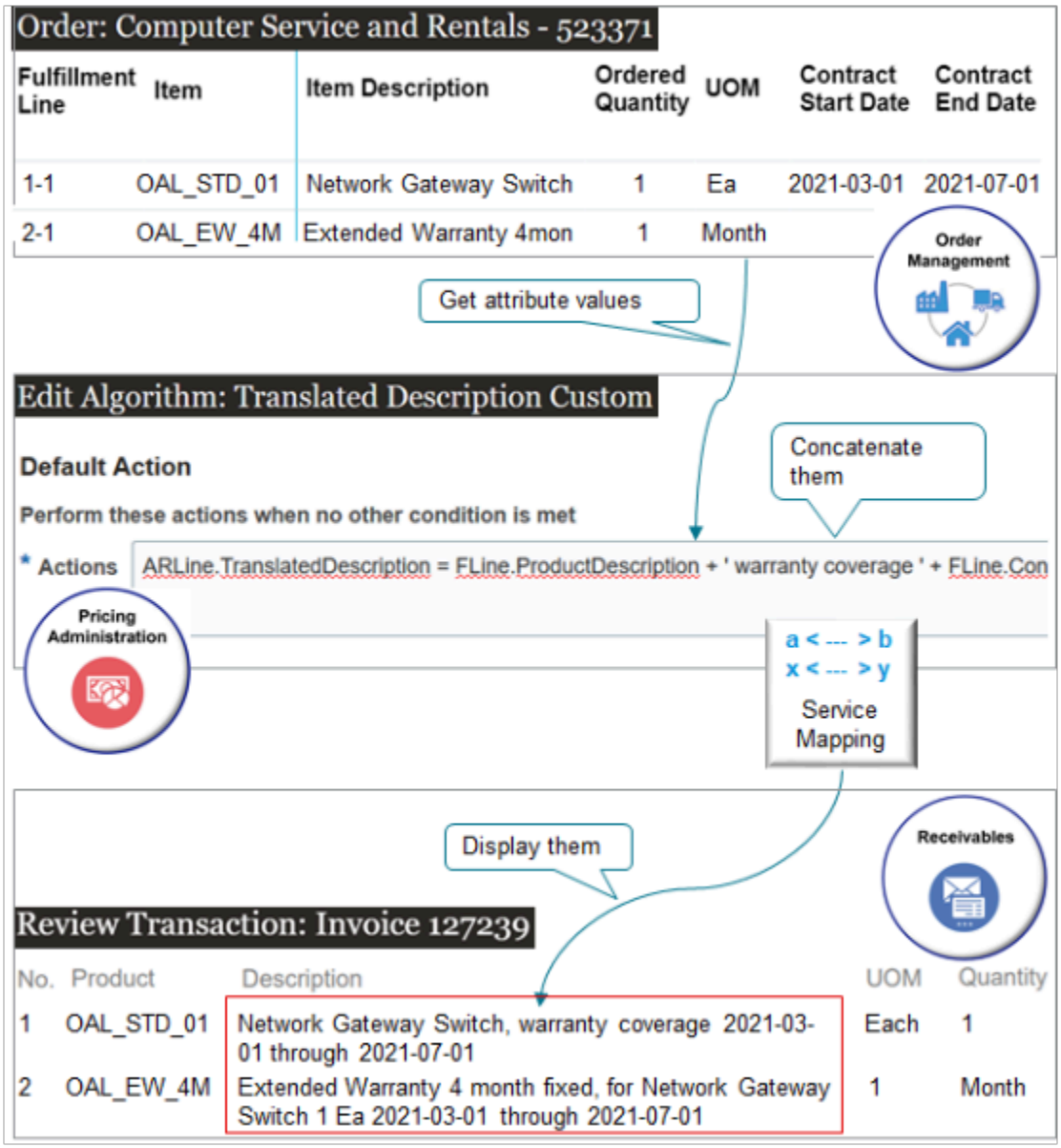
Order Management comes predefined to price an item on the catalog line according to the ORA_BUY line type. If your set up uses any other line type, then the price that you see on the order line might be different from the price that you see on the catalog line.

For example, you create a pretransformation rule that sets the line type to any value other than ORA_BUY, such as MY_LINE_TYPE_1. Next, you set up a different price on your price list or discount list for your item on a line type that isn't ORA_BUY, such as MY_LINE_TYPE_1. If you add the item to an order line, then Order Management will display the price for the MY_LINE_TYPE_1 line type on the order line. For details, see *How Pricing Calculates the Catalog Line*.

Use Integration Algorithms to Implement Complex Logic

Use an integration algorithm to implement specialized behavior when you integrate Order Management.

Here's an example.



Note

- You get attributes value on the fulfillment line from Order Management, including the Item Description, Quantity, Unit of Measure, Contract Start Date, and Contract End Date.
- You use an integration algorithm in the Pricing Administration work area to concatenate the values for the attributes into a single string of text, and then use a service mapping to map it to Accounts Receivable.
- The Description on the invoice in Accounts Receivable displays the string.
- If the sales order contains a coverage item, such as a warranty, then you create another concatenated string for the description of the coverage item. This string includes the description of the coverage item plus the same attribute values that the string for the covered line has.

Your mapping requirement is too complex to meet with only a service mapping, so you set up an integration algorithm that creates the concatenated string and gets the data you'll need from the order line that has the covered item.

Here are the items that you use in this example.

- The OAL_STD_01 Network Gateway Switch is the covered item.
- The OAL_EW_4M_Fixed Extended Warranty is a coverage item that covers the gateway switch.

Summary of the Setup

1. Create integration algorithm.
2. Add functions to your integration algorithm.
3. Add a step to your integration algorithm.
4. Manage service mapping.
5. Test your setup.

For a multimedia demonstration of a similar set up, go to *Order Management Enhancements*. The demonstration starts at 49:35 and ends at 59:03.

Create Integration Algorithm

1. Create your algorithm.
    - Make sure you have the privileges that you need to administer Order Management.
    - Go to the Pricing Administration work area.
    - In the Pricing Administration work area, click **Tasks**.
    - Under Order Management Configuration, click **Manage Algorithms**.
    - On the Manage Algorithms page, create a new algorithm.

Attribute	Value
Name	Translated Description Custom
Description	Get the description for a covered line, and then add it to the description for the coverage line on the invoice.

2. Click **Variables**, click **Actions > Add Row**, then set the values.

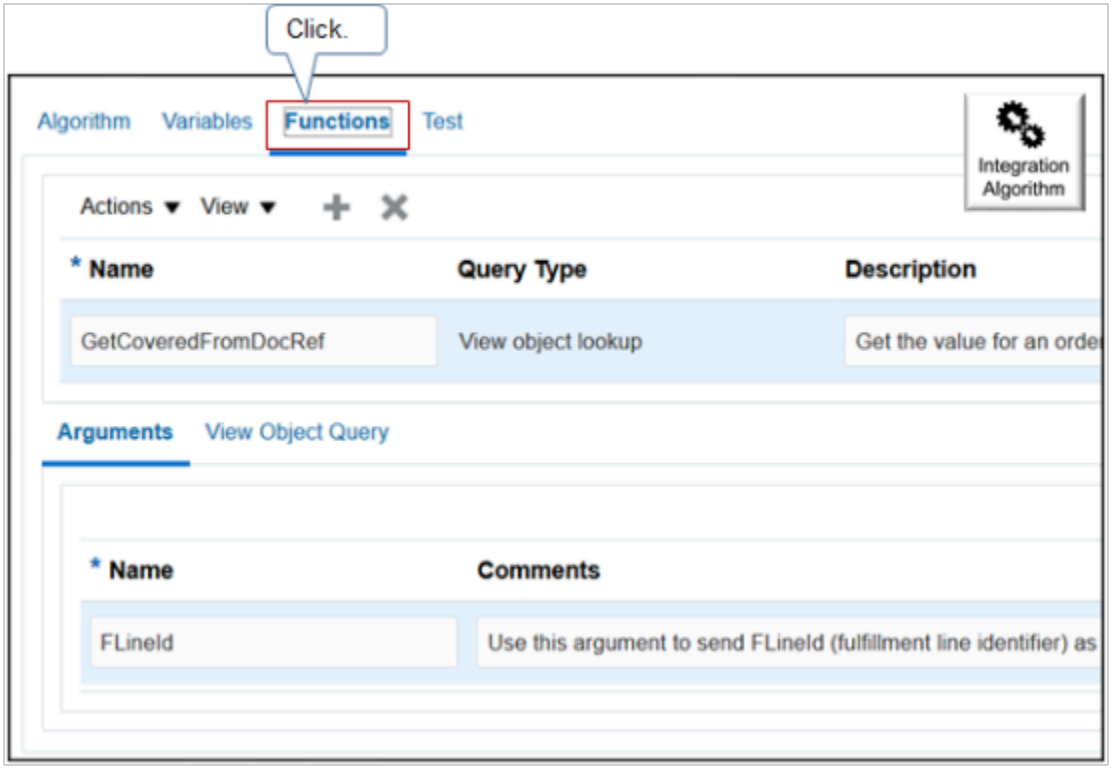
Attribute	Value
Name	ARIntegration
Data Type	Data Object
Required	Contains a check mark.
Input/Output	Input and Output
Internal Service Schema	FulfillmentIntegration.InvoiceService

Attribute	Value
	This value instructs the algorithm to get the attribute values you need from the invoice service.

This variable will store the object definition that you need to integrate with Accounts Receivable.

3. Click **Save**.

Add Functions to Your Integration Algorithm



Try it.

1. Add a function.
    - Click **Functions**, click **Actions > Add Row**, then set the values.

Attribute	Value
Name	GetCoveredFromDocRef
Query Type	View Object Lookup
Description	Get the value that identifies the order line that has the covered item.

- In the Arguments area, click **Add Row**, then add an argument.

Attribute	Value
Name	FLineId
Comments	Use this argument to send the FLineId (fulfillment line identifier) that identifies the fulfillment line that contains the coverage item, and then return a value that identifies the covered item.

- Click **View Object Query**, then add a new query.

Attribute	Value
Application Module	oracle.apps.scm.doo.common.process.model.applicationModule.DooExtensibleMapperAM This is a predefined value. You must not use any other value.
Application Configuration	DooExtensibleMapperAMLocal This is a predefined value. You must not use any other value.
View Object	DocumentReferenceVO This view object contains mapping values from coverage item and the covered item.
View Criteria	getCoveredLine The getCoveredLine view criteria on the DocumentReferenceVO view object helps the algorithm identify the fulfillment line that has the covered item.
Single Row	Contains a check mark.

- In the Bind Variables area, create a new bind variable.

Attribute	Value
Bind Variable Name	flineId
Bind Variable Value	FLineId

Attribute	Value

- Click **Save**.

2. Add another function.

Attribute	Value
Name	GetCoveredFLine
Query Type	View Object Lookup
Application Module	oracle.apps.scm.doo.common.process.model.applicationModule.DooExtensibleMapperAM This is a predefined value. You must not use any other value.
Application Configuration	DooExtensibleMapperAMLocal This is a predefined value. You must not use any other value.
View Object	FulfillLineVO
View Criteria	getFLineByFLineId
Single Row	Contains a check mark.
Bind Variable Name	FLineId
Bind Variable Value	FLineId

3. Add another function.

Attribute	Value
Name	GetItemDesc
Query Type	View Object Lookup
Application Module	oracle.apps.scm.doo.common.process.model.applicationModule.DooExtensibleMapperAM

Attribute	Value
	This is a predefined value. You must not use any other value.
Application Configuration	DooExtensibleMapperAMLocal This is a predefined value. You must not use any other value.
View Object	oracle.apps.scm.productModel.items.publicView.ItemPVO
View Criteria	GetItem
Single Row	Contains a check mark.
Bind Variable Name	InventoryItemId,OrganizationId
Bind Variable Value	InventoryItemId,OrganizationId

4. Click **Save**.

Add a Step to Your Integration Algorithm

Add a conditional step that runs differently depending on whether the order line that the algorithm is currently processing contains a covered item or a coverage item.

Try it.

1. Click the Algorithm tab, click **Add Step > Conditional Action**, then set the values.

Attribute	Value
Name	Map Translated Description
Description	Concatenate the descriptions for the covered item and the coverage item.

2. In the Data Sets area, add three new data sets.

Name	Variable Path	Primary	Cardinality	Data Set Join
FLine	ARIntegration.FulfillLine	Contains a check mark.	Leave empty.	Leave empty.
ARLine	ARIntegration.ARInterface	Doesn't contain a check mark.	One	FulfillLineId: {Fline.FulfillLineId}
AllFLines	ARIntegration.FulfillLine	Doesn't contain a check mark.	Many	Leave empty.

The data sets help the algorithm to filter and find the data that it can access.

- **FLine**. Will contain details from the fulfillment line that the algorithm will use to write the description onto the covered item.
- **ARLine**. Identifies the destination where the algorithm will write the concatenated string.
- **AllFLines**. Use data from the coverage line to go and get the description from the fulfillment line that contains the covered item. AllFLines allows the algorithm to traverse all fulfillment lines in the sales order until it finds the line that has the covered item.

3. In the Execute Condition area, click **Add Condition > Add Local Variable**, then add two local variables.

Variable Name	Default
CoveredFlineDocRefRow	Leave empty.
CoveredFlineRow	Leave empty.

You will use these variables to store temporary values that the algorithm needs while processes data.

4. In the Execute Condition area, click **Add Condition > Default Action**.

You will add the code that creates the concatenated string for the covered item.

In the Default Action area, add code in the Actions window.

```
ARLine.TranslatedDescription = FLine.ProductDescription + ' warranty coverage ' +
 FLine.ContractStartDate + ' through ' + FLine.ContractEndDate
```

where

Code	What it Does
`ARLine.TranslatedDescription =`	Create a variable named TranslatedDescription and add it to the ARLine data set.

Code	What it Does
	Store the value of the concatenated string in the TranslatedDescription variable.
`FLine.ProductDescription`	Get the value of the description for the fulfillment line that we're currently processing. This is the description for the covered item from the FLine data set.
`FLine.ContractStartDate`	Get the value of the contract start date for the covered item from the FLine data set. If you add coverage to a covered item, then you use the Contract Start Date and Contract End Date on the covered item, not the coverage item. The coverage item provides coverage for a duration, such as 1 Year, but not for the specific start date and end date. Instead, you set the start date and end date on the covered item. This way, you can use the coverage item for a variety of covered items.
`FLine.ContractEndDate`	Get the value of the contract end date for the covered item from the FLine data set.
`+ ' ' +`	Concatenate the value of one string to another. Insert a space between these concatenated strings.

`+ ' warranty coverage ' +`
 	
`+ ' through ' +`
	Concatenate the value of one string to another. Insert text between these concatenated strings. For example, insert the phrase **`warranty coverage`** between these concatenated strings.

This action applies to fulfillment lines that contain a standard item or shippable item.

This code specifies how to concatenate the attribute values. Here's an example of the run time value that this code renders.

```
Network Gateway Switch, warranty coverage 2021-03-01 through 2021-07-01
```

5. Add the code that creates the concatenated string for the coverage item.

   You will add a new conditional action that applies only if the item is a coverage item. The concatenation is slightly different for a coverage item than it is for a covered item. You also need to get some details from the line that has the covered item.

   - In the Execute Condition area, click **Add Condition > True Condition**.
   - In the Conditional Actions area, add code in the True Condition window.

     ```
     FLine.SalesProductTypeCode in
     ['COVERAGE','PREVENTIVE_MAINTENANCE','SERVICE_LEVEL_AGREEMENT','SOFTWARE_MAINTENANCE','INCLUDED_WARRANTY']
     ```

     The SalesProductTypeCode attribute on the fulfillment line specifies whether the line contains a coverage item, such as a maintenance agreement, warranty, and so on. So this code specifies to do this conditional action only if the line contains a coverage item.
   - Add code in the Actions window.

     ```
     CoveredFlineDocRefRow = GetCoveredFromDocRef(FLine.FulfillLineId)
     if (CoveredFlineDocRefRow?.DocSublineId != null) {
      // Locate the covered line according to DocSublineId
      CoveredFLineRow = AllFLines.locate([FulfillLineId:
      Long.parseLong(CoveredFlineDocRefRow?.DocSublineId)])

      // If we can't find the covered line in memory then get it
      if (CoveredFLineRow == null)
      CoveredFLineRow = GetCoveredFLine(CoveredFlineDocRefRow?.DocSublineId)

      def item = CoveredFLineRow?.ProductDescription
      // If we can't find the description for the covered line in memory then get it
      if (item == null) {
      def itemRow = GetItemDesc(CoveredFLineRow.InventoryItemId,
      CoveredFLineRow.InventoryOrganizationId)
      item = itemRow.Description;
      }
      ARLine.TranslatedDescription = FLine.ProductDescription + ' for ' + item + ' ' +
      FLine.OrderedQty.toString() + ' ' + FLine.OrderedUom + ' ' + FLine.ContractStartDate + ' through
      ' + FLine.ContractEndDate
     }
     ```

     where

Code	Description
`CoveredFlineDocRefRow = GetCoveredFromDocRef(FLine.Fu`	Use the GetCoveredFromDocRef function that you created earlier on the algorithm to get the value the identifies the fulfillment line that contains the covered item. Store the value in the CoveredFlineDocRefRow variable that you created earlier on the algorithm.
`if (CoveredFlineDocRefRow?.DocSu =null)`	Ask whether `DocSublineId` contains a value. If it does contain a value, then it means the sales order contains a covered line and we need to process it.

Code	Description
`CoveredFLineRow = AllFLines.locate([FulfillLine Long.parseLong(CoveredFlineDo`	Set the value of the CoveredFLineRow attribute that you created earlier on the algorithm to the fulfillment line that contains the covered item. `CoveredFlineDocRefRow?.DocSublineId` means to get this value from the `DocSublineId` attribute of the AllFLines data set.
`if (CoveredFLineRow == null) CoveredFLineRow = GetCoveredFLine(CoveredFlineI`	If we can't find the covered line in memory then get it. Use the `DocSublineId` attribute with the GetCoveredFLine function that you created earlier to find it, then use the results to set the value for CoveredFLineRow to the line that contains the covered item.
`def item = CoveredFLineRow?.ProductDesc`	Define a variable named `item` and set its value to the contents of the ProductDescription attribute. Get the value of ProductDescription from the fulfillment line that CoveredFLineRow identifies.
`if (item == null) { def itemRow = GetItemDesc(CoveredFLineRow.I CoveredFLineRow.InventoryOrga item = itemRow.Description; }`	If your `item` variable is empty, then it means we still need to find the description for the covered line in memory. Define a variable named itemRow. Use the GetItemDesc method that you added earlier to the algorithm to get the values of the InventoryItemId attribute and the InventoryOrganizationId attribute from the fulfillment line that CoveredFLineRow contains. Use these values to verify that the row contains the item, then set the value of itemRow to this fulfillment line. Set the `item` variable to the value of the Description on the fulfillment line that itemRow identifies.
`ARLine.TranslatedDescription = FLine.ProductDescription + ' for ' + item + ' ' + FLine.OrderedQty.toString() + ' ' + FLine.OrderedUom + ' ' + FLine.ContractStartDate + ' through ' + FLine.ContractEndDate`	Create a variable named TranslatedDescription and add it to the ARLine data set. Store the value of the concatenated string in TranslatedDescription. FLine contains attribute values for the line that we're currently processing, which is the coverage item. - `FLine.ProductDescription` is the description of the coverage item. - `item` is a variable that contains the description of the covered item, such as Desktop Computer. - `FLine.OrderedQty.toString()` is the quantity of the coverage item, converted to a string. - `FLine.ContractStartDate` is the start date of the coverage item. - `FLine.ContractEndDate` is the end date of the coverage item. This description is going on the invoice line that has the coverage item, so its useful to identify the item that the coverage covers.

Code	Description

Here's an example of the run time value that this code renders.

```
Extended Warranty 4 month fixed, for Network Gateway Switch 1 Ea 2021-03-01 through 2021-07-01
```

- Click **Save and Close**, then close the Manage Algorithms tab.

Manage Service Mapping

Modify the service mapping so it provides the data attributes that your new integration algorithm.

1. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
2. In the Tasks pane, under Order Management Configuration, click **Manage Service Mappings**.
3. On the Manage Service Mappings page, click **FulfillmentIntegration**.
4. Click **Sources**.
5. Click the **row** that contains InvoiceSource in the Source column.
6. Add an attribute that you will use to send the translated description to Accounts Receivable.
   - In the Entity Mappings list, click the **row** that contains ARInterfaceLine in the Entity column.
   - In the ARInterfaceLine Details area, click **Actions > Add Row**, then set the value.

Attribute	Value
Attribute	TranslatedDescription
View Object Attribute	Leave empty
Expression	Leave empty

7. Add attributes for the fulfillment line.
   - In the Entity Mappings list, click the **row** that contains FulfillLine in the Entity column.
   - In the FulfillLine Details area, add new attributes.

Attribute	View Object Attribute
SalesProductTypeCode	SalesProductTypeCode
ProductDescription	ProductDescription
OrderedQty	OrderedQty

Attribute	View Object Attribute
OrderedUom	OrderedUom
ContractStartDate	ContractStartDate
ContractEndDate	ContractEndDate

8. Set attributes on the service so it calls your algorithm.
    - Click **Services**.
    - Click the **row** that contains InvoiceService in the Service column.
    - Set the values.

Implementation Type	Implementation
Algorithm	Translated Description Custom This is the name of the integration algorithm that you created earlier in this procedure.

9. Add your attributes to the service so the service includes them in the payload that it sends to Accounts Receivable.
    - In the Entities list, click the row that contains ARInterfaceLine in the Entity column.
    - In the ARInterfaceLine Entities area, add an attribute.

Attribute	Read	Write
Translated Description	Contains a check mark.	Contains a check mark.

    - In the Entities list, click the row that contains FulfillLine in the Entity column.
    - In the FulfillLine Entities area, add your attributes.
        - SalesProductTypeCode
        - ProductDescription
        - OrderedQty
        - OrderedUom
        - ContractStartDate
        - ContractEndDate

      Make sure the Read attribute and the Write attribute contain a check mark for each attribute that you add.
    - Click **Save and Close**

10. Publish your algorithm.
    - Click **Tasks**, and then under Order Management Configuration, click **Manage Algorithms**
    - On the Manage Algorithms page, click the row that contains Translated Description Custom in the Name column.
    - Click **Actions > Publish**.

Test Your Setup

1. Create a sales order.
    - Go to the Order Management work area, create a sales order, then add two order lines.

Item	Quantity
OAL_STD_01 Network Gateway Switch	1
OAL_EW_4M_Fixed Extended Warranty	1

    - Click **Submit**
    - Notice the sales order number. For this example, assume its 529986.
2. Verify that Order Management correctly sends your data to Oracle Receivables.
    - Make sure you have the privileges that you need to manage Accounts Receivable. For details, see *Privileges That You Need to Implement Order Management*.
    - Go to the Billing work area, then click **Tasks > Manage Transactions**.

      For details, see *Requirements for Completing a Receivables Transaction*.
    - On the Manage Transactions page, search for the transaction.

Attribute	Value
Business Unit	Vision Operations
Transaction Source	Distributed Order Orchestration
Reference	529986

Attribute	Value

- On the Review Transaction page, click **View Image**.
- On the printed invoice that displays, verify the description for the invoice line that has the network switch.

Attribute	Value
Description	`Network Gateway Switch, warranty coverage 2021-03-01 through 2021-07-01` Notice that this value is a concatenation of the description of the covered item, and then the start date and end date of the coverage. It is the result of the Default action that you created in the algorithm.

- Verify the description for the invoice line that has the extended warranty.

Attribute	Value
Description	`Extended Warranty 4 month fixed, for Network Gateway Switch 1 Ea 2021-03-01 through 2021-07-01` Notice that this value is a concatenation of the description of the coverage item, and then the description of the covered item, and then the quantity, unit of measure, start date, and end date of the coverage item. This line contains a coverage item, so it meets the condition from the algorithm. It is the result of the Conditional action that you created in the algorithm.

Related Topics

- Integrate Order Management with Accounts Receivable
- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Other Oracle Applications

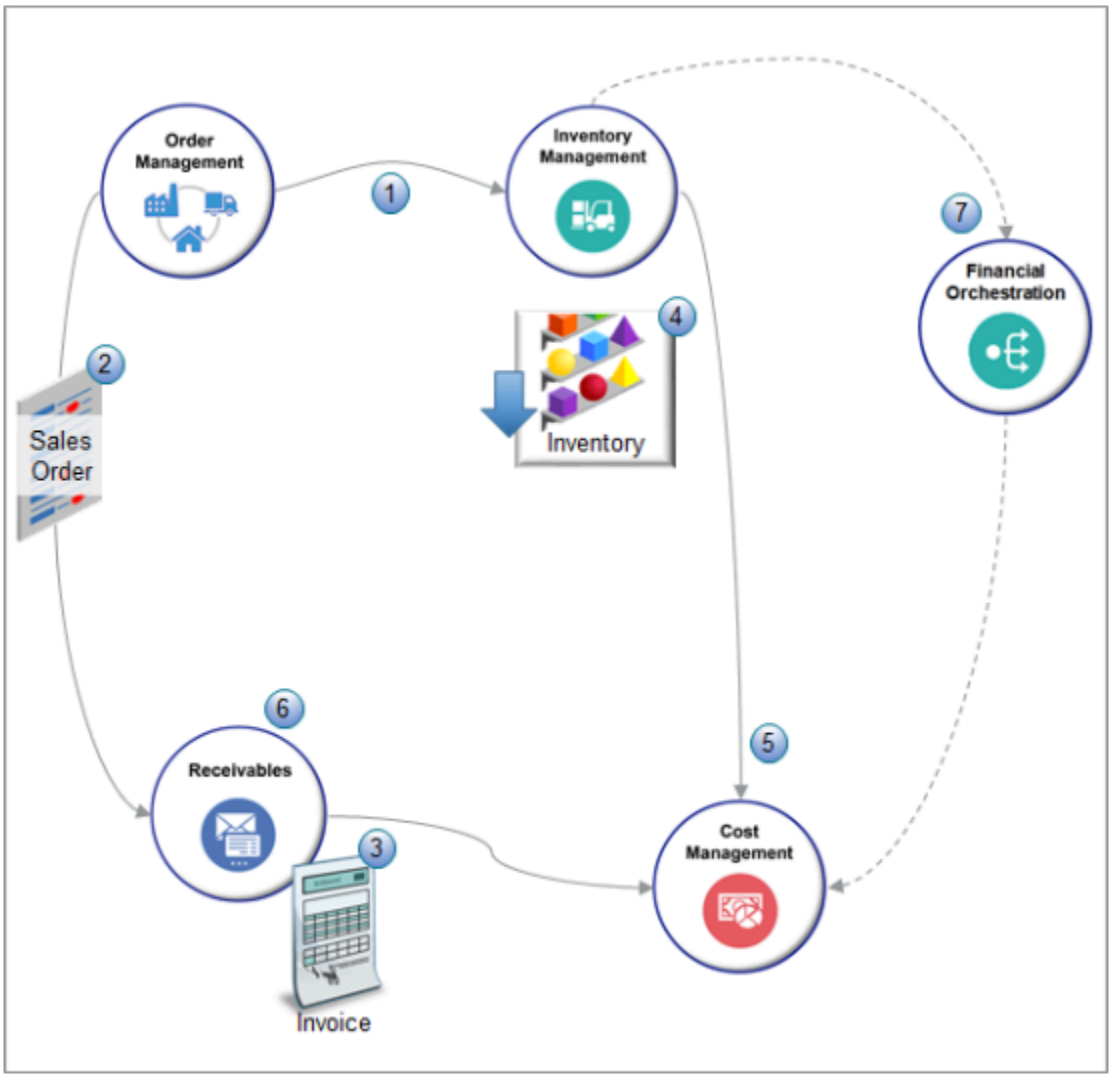
Inventory

Reduce Inventory When a Sales Order Doesn't Require Picking or Shipping

Reduce the amount of on-hand quantity in your warehouse to fulfill sales orders that you submit in Order Management.

Use this feature to help you reduce on-hand quantity in your warehouse, allocate cost and recognize cost of goods sold when you don't pick, pack, and ship your sales orders. For example, when you deliver the item through an over-the-counter transaction that doesn't require picking and shipping, or when you create and fulfill a sales order in Order Management for a point-of-sale transaction that you already finished.

Here's how it works.



What the Numbers Mean

1. Oracle Order Management validates and submits a sales order to fulfill it from your warehouse without picking or shipping, then sends the order details and a request to reduce inventory to Oracle Inventory Management.
2. Order Management sends the sales order to Oracle Accounts Receivable.
3. Receivables invoices the sales order.
4. Inventory Management reduces the amount of on-hand quantity to fulfill the sales order. The transaction type is Direct Sales Order Issue.
    - Reduces it immediately after Order Management sends the order line to inventory, or in the background.
    - Reduces it on the current date or on a date that already occurred.

5. Inventory Management sends the finished transaction for the inventory reduction to Oracle Cost Management, then Cost Management allocates cost for the transaction.
6. Receivables recognizes the revenue, then sends revenue details to Cost Management. Cost Management recognizes the cost of goods sold. As an option, you can use Oracle Revenue Management to recognize revenue, then send revenue details to Cost Management.
7. If a change in ownership happens from one business unit to another business unit, then Inventory Management sends a Direct Sales Order Issue transaction to Oracle Financial Orchestration. Financial Orchestration orchestrates financial details, processes the transaction, then sends the transaction to Cost Management.

You can realize these benefits.

- Reduce the amount of on-hand quantity that you need to fulfill each sales order when you don't pick and ship the item that you're selling.
- Avoid having to manage details about picking and shipping transactions when you don't pick and ship the item.
- Reduce the number of steps and the time that you need to process and fulfill sales orders from your warehouse.
- Fulfill a large volume of sales orders, increase revenue, and increase customer satisfaction.
- Manage change on the sales order.
- Return an order line that has an inventory transaction in the same way that you return an order line that doesn't have an inventory transaction.

For end-user details, see *Manage Transactions That Reduce Inventory*.

To get a presentation that includes a demonstration, see *Presentation to Reduce Inventory*. You can also *download the slides* for this presentation.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Accounts Receivable
- Overview of Supply Chain Financial Orchestration
- Considerations for Creating Material Statuses
- Considerations for Creating Subinventories and Locators

Setup Options for Reducing Inventory

Get details about some of the options you have when you set up inventory reduction for Order Management.

Use Extensible Flexfields

Use an extensible flexfield on an order line in Order Management to send an attribute value to a descriptive flexfield on the inventory transaction in Inventory Management.

You create the INVENTORY_TRX_INFORMATION_LINE context in Order Management and in Inventory Management, and can add up to 35 attributes in this context when you set up your flexfields.

Here are the attributes that you can use.

Extensible Flexfield on the Order Line	Descriptive Flexfield on the Inventory Transaction	Data Type
ATTRIBUTE_CHAR1 through ATTRIBUTE_CHAR20	ATTRIBUTE1 through ATTRIBUTE20	VARCHAR2
ATTRIBUTE_NUMBER1 through ATTRIBUTE_NUMBER10	ATTRIBUTE1 through ATTRIBUTE10	NUMBER
ATTRIBUTE_DATE1 through ATTRIBUTE_DATE5	ATTRIBUTE1 through ATTRIBUTE5	DATE

Use the Allow Inventory Transaction Attribute in Your Processing Logic

- Use the Allow Inventory Transaction attribute in an assignment rule. For example, assign the predefined orchestration process to an order line only if the Allow Inventory Transaction attribute is Y.
- Use the Allow Inventory Transaction attribute and the Sent to Inventory attribute in a payload that you use with the Order Information web service to get sales orders that have an inventory transaction.
- Use the Allow Inventory Transaction attribute and the Sent to Inventory attribute in the line-selection criteria or branching condition on an orchestration process. Here are some examples.
    - If the Allow Inventory Transaction attribute is Y, then don't use a schedule task to process the order line.
    - Branch an orchestration process according to the value in the Allow Inventory Transaction attribute.

  The names that you use for these attributes in your rule sets and web service payloads are different. If you use a web service:
    - Use InventoryTransactionFlag instead of Allow Inventory Transaction.
    - Use InventoryInterfacedFlag instead of Sent to Inventory.
- Create a validation rule set for the Allow Inventory Transaction attribute. Use it or the predefined Fulfillment Lines Are Interfaced to Inventory Management rule set as a condition in a processing constraint that you create. For example:
    - If Allow Inventory Transaction equals Yes on an order line, or if Order Management already sent the line to inventory, then don't allow an update on an attribute.

Transformation Rules and Order Management Extensions

Create a transformation rule or order management extension that sets the Allow Inventory Transaction attribute to Y on the order line.

Consider an order management extension that says:

- If the order type is STD, and if the item is a standard and shippable item, then set the Allow Inventory Transaction attribute to Y.
- It uses the On-Save event.

Here's the extension code.

```
def lines = header.getAttribute("Lines");
def orderType = header.getAttribute("TransactionTypeCode");
```

```
if(!orderType.equals("STD"))
return;
while (lines.hasNext()) {
 def line = lines.next();
 def itemSubTypeCode = line.getAttribute("ItemSubTypeCode");
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 if(itemSubTypeCode.equals("STANDARD")){
 //Derive Shippable Flag value
 def item = getItem(inventoryItemId, orgId);
 String shippableFlag = item.getAttribute("ShippableItemFlag");
 if("Y".equals(shippableFlag)){
 line.setAttribute("InventoryTransactionFlag","Y");
 }
 }
}

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 def rowset = itemPVO.findByViewCriteria(vc, -1);
 def item = rowset.first();
 return item;
}
```

where

Code	Description
`line.setAttribute("InventoryTrans "Y")`	Sets the value for the Allow Inventory Transaction attribute.
`if(!orderType.equals("my_value")`	specifies the order type. For example, test for the standard order type.
`if(!orderType.equals("my_value")`	Specifies the order type. For example, test for the standard order type. `if(!orderType.equals("STD")`
`if(itemSubTypeCode.equals("STANDA`	Specifies the item subtype. You must use STANDARD.
`if("Y".equals(shippableFlag)`	Specifies that the item is shippable. You must set shippableFlag to Y.

Use this extension only for a standard item that you can ship.

Related Topics

- Overview of Using Extensible Flexfields in Order Management
- Guidelines for Assigning Orchestration Processes
- Web Services That You Can Use to Integrate Order Management
- Overview of Creating Order Management Extensions

Guidelines for Reducing Inventory

Use these guidelines when you set up Order Management to reduce inventory.

- Order Management doesn't validate the amount of on-hand quantity that's available in your inventory management system for an inventory reduction. So, make sure your inventory management system correctly reflects the on-hand quantity that's available. If it doesn't, the inventory transaction will fail in Inventory Management.
- The supply task that creates supply doesn't process the order line that has the inventory transaction. So, make sure you have enough on-hand quantity available for the item to do the inventory reduction even if you set the Back-to-Back Enabled attribute to Yes for the item in the Product Information Management work area.
- Order Management doesn't send an order revision, or an update to Inventory Management, and it doesn't accept an update from Inventory Management. So, you can fix errors that happen after Order Management successfully sends the order line to Inventory Management only in Inventory Management. You can't fix them in Order Management.
- You can't use an inventory transaction with a model or a kit. If you must reduce on-hand inventory for a child item in a model or a kit, and if the child is a standard, shippable item, and if you don't need to pick or ship the child, then you can place an order for the child.
- You can't use a fulfillment tolerance with an order line that has an inventory transaction. Order Management will ignore any fulfillment tolerance for the line.
- Don't set the Allow Inventory Transaction attribute value to Yes on the order line for sales orders that you create before update 21B. If you import a source order, then don't set the InventoryTransactionFlag attribute to true.
- You can't use file-based data import (FBDI), REST API, or a SOAP web service to create an inventory transaction for this feature in Inventory Management.

If you opt into the Reduce Inventory When a Sales Order Doesn't Require Picking or Shipping feature, then:

- You can't opt out of it after you submit a sales order that has an inventory transaction on the order line. At this point, you're committed to using the feature.
- You can use the Order Management work area to set the subinventory on any order line from the subinventories that you maintain in the Manage Subinventories task, but setting the subinventory doesn't affect how Order Management reserves inventory. You can't use Order Management to reserve the quantity on an order line from the specified subinventory.
- And if you haven't shipped or fulfilled the order line, then you can use the Order Import web service to remove lot and serial details from the line.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Guidelines for Setting Up Shipment Tolerances
- Overview of Configure-to-Order
- Overview of Importing Orders Into Order Management

Reduce Inventory Without Picking or Shipping

Do these steps so you can start using Order Management to reduce inventory without picking or shipping.

Summary of the Set Up

1. Enable the feature.

2. Set up the order management parameters.
3. Orchestrate fulfillment.

1. Enable the Feature

1. Go to the Setup and Maintenance work area, then select the Order Management offering.
2. Click **Change Feature Opt In**.
3. On the Opt In page, in the Order Management row, click the **pencil**.
4. On the Edit Features page, enable the Reduce Inventory When a Sales Order Doesn't Require Picking or Shipping feature.

2. Set Up the Order Management Parameters

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Parameters

    For details, see *Manage Order Management Parameters*.
2. On the Manage Order Management Parameters page, set the values.

Parameter	Description
Inventory Transaction Date for Order Lines	Specify the transaction date that Order Management sends to Inventory Management. Inventory Management uses this date as the transaction date for the order line when it reduces on-hand quantity - Current Date - Ordered Date - Scheduled Ship Date - Requested Ship Date The default value is Current Date. If you set it to Ordered Date, Scheduled Ship Date, or Requested Ship Date, and if this date is empty on the order line or if it happens in the future, then Order Management uses the current date.
Process Inventory Transactions Immediately	Reduce on-hand quantity for each order line that you send to Inventory Management. Set a value. - **Yes**. Process inventory reduction immediately. Use this setting to keep your inventory accurate and up-to-date with real-time data. If an error happens while processing the inventory transaction, then the transaction lines are available as pending transactions in Inventory Management and the orchestration process will move the order line to the next fulfillment task. You can manually modify and process the transactions later in Inventory Management. - **No**. Process inventory reduction in the background. Use this setting to process each sales order more quickly. The transaction lines are available as pending transactions in

Parameter	Description
	Inventory Management and the orchestration process will move the order line to the next fulfillment task. Inventory Management will automatically process the transactions in the background. The default value is Yes.
Process Inventory Transaction Lines as a Group	Set a value. o **Yes**. Order Management will send order lines as a group to Inventory Management when you submit the sales order, and Inventory Management will process them together. Use this setting to improve efficiency when you must process a large volume of sales order. If you set it to Yes, then you can't use a hold on the order line to prevent Order Management from sending the line to inventory. Order Management will send lines that allow inventory transactions to inventory when you submit the order even if the line is on hold. o **No**. Order Management will send each line individually to Inventory Management, and Inventory Management will process them separately. The default value is No.

Inventory Transaction Date for Order Lines

If the inventory transaction task is a fulfillment completion step in your orchestration process, then Order Management sets this date as the fulfillment date on the order line.

If you set the parameter value as the ordered date, scheduled ship date, or requested ship date, and if that date happens in the past, then make sure you set up the profile options in Inventory Management to avoid errors when processing the inventory transaction.

Profile Option	Description
Transaction Date Validation Enabled	Set it to No Validation or to Validate Transaction Date.
Maximum Number of Days Prior to Current Date in Which a Transaction Can Be Created	If you set the Transaction Date Validation Enabled profile option to No Validation, then you must set up the Maximum Number of Days Prior to Current Date in Which a Transaction Can Be Created profile option. The default value is 5. The value that you use depends on how many days you want to allow Inventory Management to create the transaction before the current date.

Assume today's date is July 3 and you set these values.

Profile Option	Value
Transaction Date Validation Enabled	No Validation
Maximum Number of Days Prior to Current Date in Which a Transaction Can Be Created	5

You can send the inventory transaction date from Order Management as June 28 or later. If Order Management sends an inventory transaction date before June 28, then you will encounter an error message when Inventory Management processes the transaction.

If you set Transaction Date Validation Enabled to Validate Transaction Date, then make sure the inventory transaction date that Order Management sends happens when the cost accounting period is open.

Assume today's date is July 3, you set Maximum Number of Days Prior to Current Date in Which a Transaction Can Be Created to any value, and the cost accounting period is open for July but closed for June.

Order Management can send an inventory transaction date of July 2 or July 3 because these dates happen when the cost accounting period is open. If Order Management sends the transaction in June, then you will encounter an error when Inventory Management processes the transaction because the cost accounting period for June is closed.

Process Inventory Transaction Lines as a Group

If you set the Process Inventory Transaction Lines as Group parameter to Yes, then:

- Order Management sends the order lines to Inventory Management to reduce inventory when you submit the sales order, and then the orchestration process moves the order line to the next fulfillment task. However, if a failure happens, then Order Management won't send the lines and the order will remain in Draft status.
- Order Management sends details to Inventory Management, such as the item, lot, serial, warehouse, subinventory, quantity, and so on. It also sends values for any project attributes or extensible flexfields that you provide.
- You must use an orchestration process that includes a fulfillment task that updates the Actual Fulfillment Date attribute and the Fulfilled Quantity attribute. For example, you can use the predefined DOO_BillOnlyGenericProcess orchestration process because it uses an invoice task as the fulfillment completion step, and that step updates the Actual Fulfillment Date and the Fulfilled Quantity when Order Management invoices your order line. If you can't use DOO_BillOnlyGenericProcess for some reason, then create your own orchestration process and make sure it has a fulfillment completion step that the process runs during order fulfillment.
- Don't use the DOO_InventoryTransactionProcess orchestration process on your order line. If you do, then Order Management doesn't use the DOO_InventoryTransaction task to reduce inventory, so it won't update fulfillment attributes on the order line, such as Actual Fulfillment Date or Fulfilled Quantity. You might encounter a problem when you receive a return item because Oracle Receiving needs a value in the Actual Fulfillment Date attribute to receive the return line.
- Make sure the orchestration process doesn’t have an inventory transaction, schedule, reservation, shipment, return, procurement, or supply task. These tasks don't process order lines that Order Management already sent to inventory.
- Don't include tasks in the orchestration process that must run before you reduce inventory.
- You can't use a hold on the order line to prevent Order Management from sending the line to inventory. Order Management will send lines that allow inventory transactions to inventory when you submit the order even if the line is on hold.

3. Orchestrate Fulfillment

Deploy the predefined orchestration process that comes with this feature.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders

- Task: Manage Orchestration Process Definitions

2. On the Manage Orchestration Process Definitions page, query for the value.

Attribute	Value
Process Name	DOO_InventoryTransactionProcess

3. Deploy the process.
   For details about how, see *Deploy Orchestration Processes*.

Here's what the process does.

- Uses the predefined DOO_InventoryTransaction task type and Invoice task type to fulfill the order line. The DOO_InventoryTransaction task is the fulfillment completion step in this process.
- Sends the order line to Inventory Management, fulfills the order line, then sends the order line to Accounts Receivable to invoice the line.
- Sends the required details to Inventory Management, such as details about the item, lot, serial, warehouse, subinventory, quantity, and so on. Order Management also sends values for any project attributes or extensible flexfields that you provide.

If you set the Process Inventory Transaction Lines as Group parameter to No, then we recommend that you use the predefined DOO_InventoryTransactionProcess orchestration process. If you create your own orchestration process instead of using the predefined one, then:

- Reference the predefined Inventory Transaction task type in your process. To get details about this task type, go to the Manage Task Types page in the Setup and Maintenance work area, then query for the value.

Attribute	Value
Task Type	DOO_InventoryTransaction

- Don't use a scheduling task or a reservation task with DOO_InventoryTransaction in the same orchestration process. If you can't meet this requirement, then make sure your orchestration process has a branching condition or a line-selection criteria that prevents the schedule task or reservation task from running on an order line that has an inventory transaction.
- Include a fulfillment task in your process that makes sure Order Management updates the fulfilled quantity and the actual fulfillment date on the order line during fulfillment. Note that the inventory transaction task is the fulfillment completion step in the predefined orchestration process.

Import Your Sales Order

As an option, you can use file-based data import (FBDI), the Application Development Framework (ADF) web service, or REST API to import your sales order.

Here's an example ADF payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
```

```
<ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
<ns2:BatchName/>
<ns2:Order>
<ns2:SourceTransactionIdentifier>Inv_37_210321080411</ns2:SourceTransactionIdentifier>
<ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
<ns2:SourceTransactionNumber>Inv_37_210321080411</ns2:SourceTransactionNumber>
<ns2:BuyingPartyName>FOM-Customer-001</ns2:BuyingPartyName>
<ns2:BuyingPartyContactName>James Pattison</ns2:BuyingPartyContactName>
<ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
<ns2:TransactionOn>2021-03-21T08:04:12</ns2:TransactionOn>
<ns2:RequestingBusinessUnitName>Vision Operations</ns2:RequestingBusinessUnitName>
<ns2:OrigSystemDocumentReference>DOO_InventoryTransactionProcess</ns2:OrigSystemDocumentReference>
<ns2:TransactionTypeCode>SHOP</ns2:TransactionTypeCode>
<ns2:FreezePriceFlag>false</ns2:FreezePriceFlag>
<ns2:FreezeShippingChargeFlag>false</ns2:FreezeShippingChargeFlag>
<ns2:FreezeTaxFlag>false</ns2:FreezeTaxFlag>
<ns2:ShipToPartyName>FOM-Customer-001</ns2:ShipToPartyName>
<ns2:ShipToAddress1>1045, 5th Avenue</ns2:ShipToAddress1>
<ns2:ShipToCity>San Diego Country Estate</ns2:ShipToCity>
<ns2:ShipToPostalCode>92065</ns2:ShipToPostalCode>
<ns2:ShipToState>CA</ns2:ShipToState>
<ns2:ShipToCountry>US</ns2:ShipToCountry>
<ns2:ShipToPartyContactName>James Pattison</ns2:ShipToPartyContactName>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:BillToCustomerName>FOM-Customer-001</ns2:BillToCustomerName>
<ns2:BillToCustomerIdentifier>300100046859202</ns2:BillToCustomerIdentifier>
<ns2:BillToAddress1>1045, 5th Avenue</ns2:BillToAddress1>
<ns2:BillToCity>San Diego Country Estate</ns2:BillToCity>
<ns2:BillToPostalCode>92065</ns2:BillToPostalCode>
<ns2:BillToState>CA</ns2:BillToState>
<ns2:BillToCountry>US</ns2:BillToCountry>
<ns2:CustomerPONumber>SHOPPO1</ns2:CustomerPONumber>
<ns2:BillToAccountContactName>James Pattison</ns2:BillToAccountContactName>
<ns2:FreezePriceFlag>FALSE</ns2:FreezePriceFlag>
<ns2:FreezeShippingChargeFlag>FALSE</ns2:FreezeShippingChargeFlag> <ns2:FreezeTaxFlag>FALSE</
ns2:FreezeTaxFlag>
<ns2:SalesCredit>
<ns2:SourceTransactionSalesCreditIdentifier>OSC-001</ns2:SourceTransactionSalesCreditIdentifier>
<ns2:SalesPerson>Paul Robert Scholes</ns2:SalesPerson>
<ns2:Percent>100</ns2:Percent>
<ns2:SalesCreditTypeCode>1</ns2:SalesCreditTypeCode>
</ns2:SalesCredit>
<ns2:Line>
<ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
<ns2:ParentLineReference/>
<ns2:RootParentLineReference/>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:ProductNumber>AS92888</ns2:ProductNumber>
<ns2:OrderedQuantity>10</ns2:OrderedQuantity>
<ns2:OrderedUOM>Each</ns2:OrderedUOM>
<ns2:RequestedFulfillmentOrganizationCode>M1</ns2:RequestedFulfillmentOrganizationCode>
<ns2:CustomerPONumber>SHOPPO1</ns2:CustomerPONumber>
<ns2:CustomerPOLineNumber>1</ns2:CustomerPOLineNumber>
<ns2:RequestedShipDate>2021-03-21T08:04:12</ns2:RequestedShipDate>
<ns2:PaymentTerms>IMMEDIATE</ns2:PaymentTerms>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:Comments/>
<ns2:TaxExempt>S</ns2:TaxExempt>
<ns2:ShipToPartyName>FOM-Customer-001</ns2:ShipToPartyName>
<ns2:ShipToAddress1>1045, 5th Avenue</ns2:ShipToAddress1>
<ns2:ShipToCity>San Diego Country Estate</ns2:ShipToCity>
<ns2:ShipToPostalCode>92065</ns2:ShipToPostalCode>
<ns2:ShipToState>CA</ns2:ShipToState>
```

```
<ns2:ShipToCountry>US</ns2:ShipToCountry>
<ns2:ShipToPartyContactName>James Pattison</ns2:ShipToPartyContactName>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:BillToCustomerName>FOM-Customer-001</ns2:BillToCustomerName>
<ns2:BillToCustomerIdentifier>300100046859202</ns2:BillToCustomerIdentifier>
<ns2:BillToAddress1>1045, 5th Avenue</ns2:BillToAddress1>
<ns2:BillToCity>San Diego Country Estate</ns2:BillToCity>
<ns2:BillToPostalCode>92065</ns2:BillToPostalCode>
<ns2:BillToState>CA</ns2:BillToState>
<ns2:BillToCountry>US</ns2:BillToCountry>
<ns2:BillToAccountContactName>James Pattison</ns2:BillToAccountContactName>
<ns2:InventoryTransactionFlag>true</ns2:InventoryTransactionFlag>
<ns2:SubInventoryCode>SC19091511</ns2:SubInventoryCode>
</ns2:Line>
<ns2:Line>
<ns2:SourceTransactionLineIdentifier>102</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>102</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>2</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>2</ns2:SourceTransactionScheduleNumber>
<ns2:ParentLineReference/>
<ns2:RootParentLineReference/>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:ProductNumber>INV-110</ns2:ProductNumber>
<ns2:OrderedQuantity>3</ns2:OrderedQuantity>
<ns2:OrderedUOM>Each</ns2:OrderedUOM>
<ns2:RequestedFulfillmentOrganizationCode>M1</ns2:RequestedFulfillmentOrganizationCode>
<ns2:CustomerPONumber>SHOPPO1</ns2:CustomerPONumber>
<ns2:CustomerPOLineNumber>2</ns2:CustomerPOLineNumber>
<ns2:RequestedShipDate>2021-03-21T08:04:12</ns2:RequestedShipDate>
<ns2:PaymentTerms>IMMEDIATE</ns2:PaymentTerms>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:Comments/>
<ns2:TaxExempt>S</ns2:TaxExempt>
<ns2:ShipToPartyName>FOM-Customer-001</ns2:ShipToPartyName>
<ns2:ShipToAddress1>1045, 5th Avenue</ns2:ShipToAddress1>
<ns2:ShipToCity>San Diego Country Estate</ns2:ShipToCity>
<ns2:ShipToPostalCode>92065</ns2:ShipToPostalCode>
<ns2:ShipToState>CA</ns2:ShipToState>
<ns2:ShipToCountry>US</ns2:ShipToCountry>
<ns2:ShipToPartyContactName>James Pattison</ns2:ShipToPartyContactName>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:BillToCustomerName>FOM-Customer-001</ns2:BillToCustomerName>
<ns2:BillToCustomerIdentifier>300100046859202</ns2:BillToCustomerIdentifier>
<ns2:BillToAddress1>1045, 5th Avenue</ns2:BillToAddress1>
<ns2:BillToCity>San Diego Country Estate</ns2:BillToCity>
<ns2:BillToPostalCode>92065</ns2:BillToPostalCode>
<ns2:BillToState>CA</ns2:BillToState>
<ns2:BillToCountry>US</ns2:BillToCountry>
<ns2:BillToAccountContactName>James Pattison</ns2:BillToAccountContactName>
<ns2:InventoryTransactionFlag>true</ns2:InventoryTransactionFlag>
<ns2:SubInventoryCode>SC19091512</ns2:SubInventoryCode>
<ns2:LotSerial>
<ns2:SourceTransactionLotIdentifier>1</ns2:SourceTransactionLotIdentifier>
<ns2:LotNumber>LL10139</ns2:LotNumber>
<ns2:SerialNumberFrom>WOS202887</ns2:SerialNumberFrom>
<ns2:SerialNumberTo>WOS202887</ns2:SerialNumberTo>
<ns2:ItemRevisionNumber>A</ns2:ItemRevisionNumber>
<ns2:Locator>R1.R1.B1</ns2:Locator>
<ns2:Quantity>1</ns2:Quantity>
</ns2:LotSerial>
<ns2:LotSerial>
<ns2:SourceTransactionLotIdentifier>2</ns2:SourceTransactionLotIdentifier>
<ns2:LotNumber>LL10139</ns2:LotNumber>
<ns2:SerialNumberFrom>WTS202893</ns2:SerialNumberFrom>
<ns2:SerialNumberTo>WTS202894</ns2:SerialNumberTo>
<ns2:ItemRevisionNumber>A</ns2:ItemRevisionNumber>
```

```
<ns2:Locator>R2.R2.B2</ns2:Locator>
<ns2:Quantity>2</ns2:Quantity>
</ns2:LotSerial>
</ns2:Line>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Specify the same values that you specify when you create a sales order in the Order Management work area. For example, set InventoryTransactionFlag to true, specify Requested Fulfillment Organization, Subinventory, or if a lot controls the item, then set a value in the Lot attribute. If you don't, or if your value isn't correct, you will receive an error.

Here's a REST API that does the same thing.

```
{
"SourceTransactionNumber":"Inv_36_210321080246",
"SourceTransactionSystem":"GPR",
"SourceTransactionId":"Inv_36_210321080246",
"BusinessUnitName":"Vision Operations",
"BuyingPartyName":"FOM-Customer-001",
"BuyingPartyNumber":"CDRM_78619",
"BuyingPartyContactName":"James Pattison",
"TransactionType":"Shop Orders",
"SubstituteAllowedFlag":false,
"ShipsetFlag":false,
"PartialShipAllowedFlag":false,
"RequestedShipDate":"2021-03-21T08:02:4600:00",
"RequestingBusinessUnitName":"Vision Operations",
"RequestedFulfillmentOrganizationCode":"M1",
"PaymentTerms":"IMMEDIATE",
"TransactionalCurrencyName":"US Dollar",
"CanceledFlag":false,
"FreezePriceFlag":false,
"FreezeShippingChargeFlag":false,
"FreezeTaxFlag":false,
"CustomerPONumber":"SHOPPO1",
"SubmittedFlag":true,
"PreCreditCheckedFlag":false,
"SourceTransactionRevisionNumber":1,
"OrigSystemDocumentReference":"DOO_InventoryTransactionProcess",
"additionalInformation":[
{
"Category": "DOO_HEADERS_ADD_INFO",
"HeaderEffBComplianceDetailsprivateVO": [
{
"ContextCode": "ComplianceDetails",
"_ComplianceInfo": "Compliance info",
"_ComplianceDate": null,
"_CompleteCompliancedate": null,
"_ComplianceReason": "Some reason",
"_ComplianceValue": 201
}
]
}
],
"billToCustomer":[
{
"PartyName":"FOM-Customer-001",
"AccountNumber":"CDRM_11118",
"Address1":"3486, Saratoga Road",
"City":"SUNNYVALE",
"State":"CA",
"PostalCode":"94004",
```

```
"County":null,
"Province":null,
"Country":"US",
"ContactName":"James Pattison",
"ContactFirstName":"James",
"ContactLastName":"Pattison"

}
],
"shipToCustomer":[
{
"PartyName":"FOM-Customer-001",
"SiteId":300100046859204,
"Address1":"3486, Saratoga Road",
"City":"SUNNYVALE",
"State":"CA",
"PostalCode":"94004",
"County":null,
"Province":null,
"Country":"US",
"ContactName":"James Pattison",
"ContactFirstName":"James",
"ContactLastName":"Pattison"

}
],
"lines":[
{
"SubinventoryCode": "SC18193721",
"InventoryTransactionFlag":true,
"SourceTransactionLineId":"1",
"SourceTransactionLineNumber":"1",
"SourceTransactionScheduleId":"1",
"SourceScheduleNumber":"1",
"TransactionCategoryCode":"ORDER",
"RequestedFulfillmentOrganizationCode": "M1",
"ProductNumber":"AS92888",
"TransactionLineType":"Buy",
"OrderedQuantity":10,
"OrderedUOM":"Each",
"PaymentTerms":"IMMEDIATE",
"RequestedShipDate":"2020-07-23T10:41:10+00:00",
"SubstitutionAllowedFlag":false,
"TransactionBusinessCategoryName":"Sales Transaction",


"billToCustomer":[
{
"PartyName":"FOM-Customer-001",
"AccountNumber":"CDRM_11118",
"Address1":"3486, Saratoga Road",
"City":"SUNNYVALE",
"State":"CA",
"PostalCode":"94004",
"County":null,
"Province":null,
"Country":"US",
"ContactName":"James Pattison",
"ContactFirstName":"James",
"ContactLastName":"Pattison"
}
],
"shipToCustomer": [
{
"PartyName":"FOM-Customer-001",
"SiteId":300100046859204,
```

```
"Address1":"3486, Saratoga Road",
"City":"SUNNYVALE",
"State":"CA",
"PostalCode":"94004",
"County":null,
"Province":null,
"Country":"US",
"ContactName":"James Pattison",
"ContactFirstName":"James",
"ContactLastName":"Pattison"

}

]


},
{
"lotSerials":[
{
"SourceLotSerialId":"1",
"Quantity":1,
"ItemSerialNumberFrom":"WQA201578",
"ItemSerialNumberTo":"WQA201578",
"ItemRevisionNumber":"A",
"LotNumber":"LL10131",
"Locator":"1.1.1"

},
{
"SourceLotSerialId":"2",
"Quantity":2,
"ItemSerialNumberFrom":"WRT201591",
"ItemSerialNumberTo":"WRT201592",
"ItemRevisionNumber":"A",
"LotNumber":"LL10132",
"Locator":"1.1.1"

}
],
"SubinventoryCode": "SD18193835",
"InventoryTransactionFlag":true,
"SourceTransactionLineId":"2",
"SourceTransactionLineNumber":"2",
"SourceTransactionScheduleId":"2",
"SourceScheduleNumber":"2",
"TransactionCategoryCode":"ORDER",
"RequestedFulfillmentOrganizationCode": 'M1',
"ProductNumber":"INV-110",
"TransactionLineType":"Buy",
"OrderedQuantity":3,
"OrderedUOM":"Each",
"PaymentTerms":"IMMEDIATE",
"RequestedShipDate":"2020-07-23T10:41:10+00:00",
"SubstitutionAllowedFlag":false,
"TransactionBusinessCategoryName":"Sales Transaction",


"billToCustomer":[
{
"PartyName":"FOM-Customer-001",
"AccountNumber":"CDRM_11118",
"Address1":"3486, Saratoga Road",
"City":"SUNNYVALE",
"State":"CA",
"PostalCode":"94004",
```

```
 "County":null,
 "Province":null,
 "Country":"US",
 "ContactName":"James Pattison",
 "ContactFirstName":"James",
 "ContactLastName":"Pattison"

 }

],
 "shipToCustomer":[
 {
 "PartyName":"FOM-Customer-001",
 "SiteId":300100046859204,
 "Address1":"3486, Saratoga Road",
 "City":"SUNNYVALE",
 "State":"CA",
 "PostalCode":"94004",
 "County":null,
 "Province":null,
 "Country":"US",
 "ContactName":"James Pattison",
 "ContactFirstName":"James",
 "ContactLastName":"Pattison"

 }

 ]
 }

 ]
 }
```

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Related Topics

- Opt Into Features in Order Management
- Manage Order Management Parameters
- Use Order Profiles to Control Order Management Behavior
- Overview of Orchestration Processes
- Overview of Importing Orders Into Order Management

Constrain Updates According to Inventory Management's Status

Use a predefined validation rule set to constrain the updates that you can make on a fulfillment line according to the shipment line's status in Inventory Management.

Use a predefined validation rule set to constrain the updates that you can make on a fulfillment line according to the shipment line's status in Inventory Management.

You don’t need to run the *Send Intermediate Shipment Status Update* scheduled process or create an order management extension to constrain these changes. Instead use one of these predefined rule sets:

- Fulfillment Line is Picked or Staged
- Fulfillment Line is Backordered
- Fulfillment Line is Released to Warehouse

Assume you need a condition.

- If Inventory Management picks or stages a fulfillment line, then don't allow your users to update the Warehouse attribute on the fulfillment line in Order Management

Try It.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, click **Constraints**.
3. Click **Actions > Add Row**, then set the values.

Attribute	Values
Constraint Name	Inventory Management Constraint
Display Name	Constrain According to Status in Inventory Management
Constraint Entity	Order Fulfillment Line
Constrained Operation	Update
Attribute Name	Warehouse
Enabled	Contains a check mark.

4. In the Details area, in the Conditions list, click **Actions > Add Row**, then set the values.

Attribute	Values
Group Number	10
Validation Entity	Order Fulfillment Line
Validation Rule Set	Fulfillment Line is Picked or Staged
Record Set	Fulfillment Line Default Record Set
Message	You can't update the Warehouse attribute because Inventory Management already picked or staged the fulfillment line.

5. Click **Applicable Roles**, make sure the All Roles option is enabled, then click **Save**.

For more, see *Processing Constraints*.

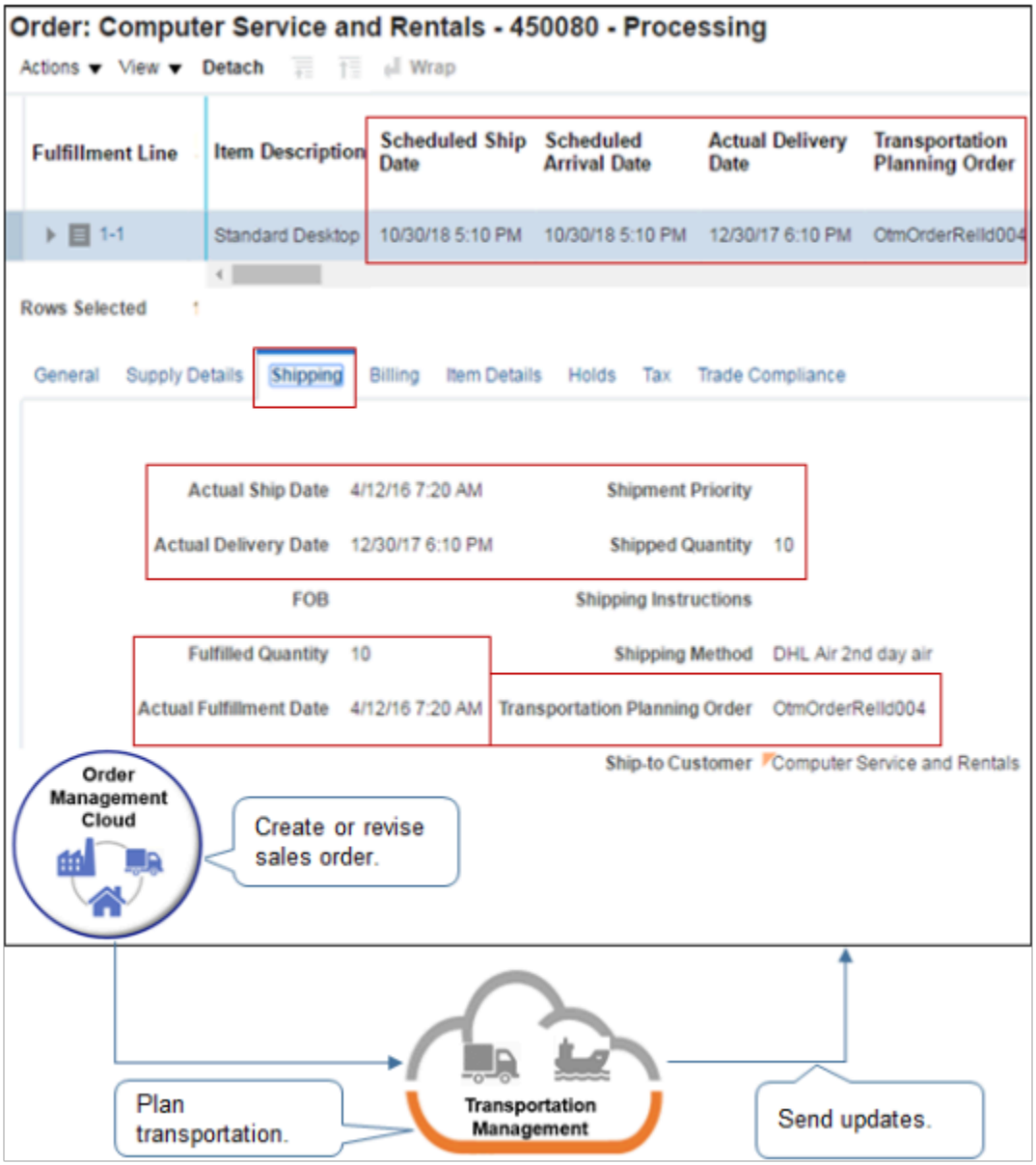
Transportation

Overview of Integrating Order Management with Transportation Management

Integrate Order Management with Oracle Transportation Management or some other transportation management system that resides outside of Oracle so it sends sales orders for transportation planning, and receives updates at fulfillment milestones, such as plan complete or proof of delivery.

- Leverage your transportation management system so it can efficiently plan transportation.
- Optimize the shipment that ships each sales order and minimize freight costs while meeting your customer delivery requirements.
- Use Transportation Management to schedule and optimize shipments, finalize the transportation plan, tender shipments to carriers, and send updated schedule dates to Order Management, providing the Order Manager with visibility to the new shipment schedule.
- Enable your users to search sales orders according to scheduled ship date, scheduled arrival date, actual delivery date, or transportation planning order.
- Update Order Management after the customer receives shipment. Use updated dates and statuses to identify exceptions and control orchestration process flow.
- Provide final delivered status after the item ships.
- Enable Order Management to use the same process for a change order that it uses for the original sales order.
- Invoice each sales order only after delivery.
- Get a delivery notification when fulfillment delivers the shipment.
- Implement an end-to-end flow with Oracle Inventory for shipments.

Order Management sends order status details to Transportation Management when the user creates, revises, or cancels the sales order. Transportation Management plans and does fulfillment, sends updated scheduling details to Order Management, then Order Management displays them in fulfillment views in the Order Management work area.



This example illustrates how the fulfillment line and Shipping tab display updated values for shipping attributes.

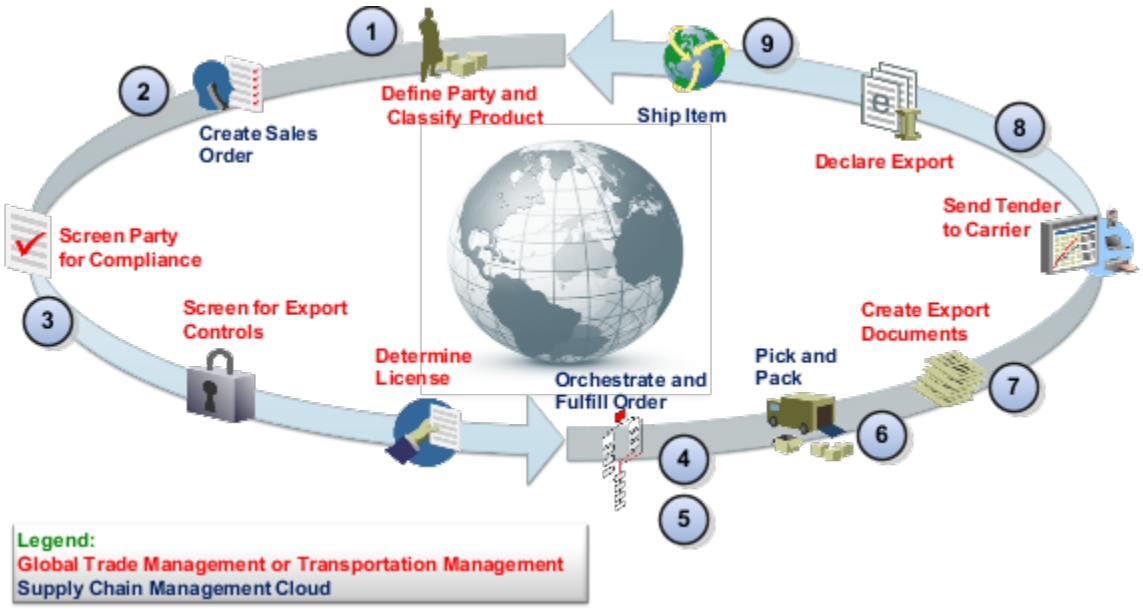
- Scheduled Ship Date
- Scheduled Arrival Date
- Actual Ship Date
- Actual Delivery Date
- Transportation Planning Order
- Shipped Quantity
- Fulfilled Quantity
- Actual Fulfillment Date

Related Topics

- Integrate Order Management with Transportation Management
- How Order Management Integrates with Transportation Management

How Order Management Integrates with Transportation Management

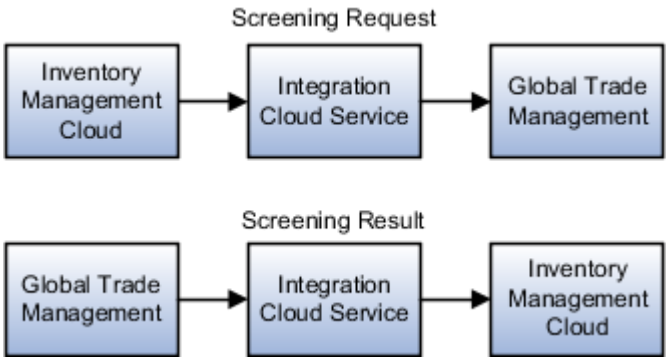
An integration between Order Management and Oracle Transportation Management can include more than one Oracle solution, such as Oracle Global Trade Management, Oracle Inventory Management, and so on.



For example, here's how an integration with trade compliance works.

1. **Define Party and Classify Product**. A Compliance Manager uses the administrative interface in Global Trade Management to set up the party that identifies the customer and to classify the item.
2. **Create Sales Order**. An Order Entry Specialist creates a sales order in the Order Management work area. The sales order references the customer and item that the Compliance Manager created.

3. **Screen Party for Compliance, Screen for Export Control, and Determine License**.



Note

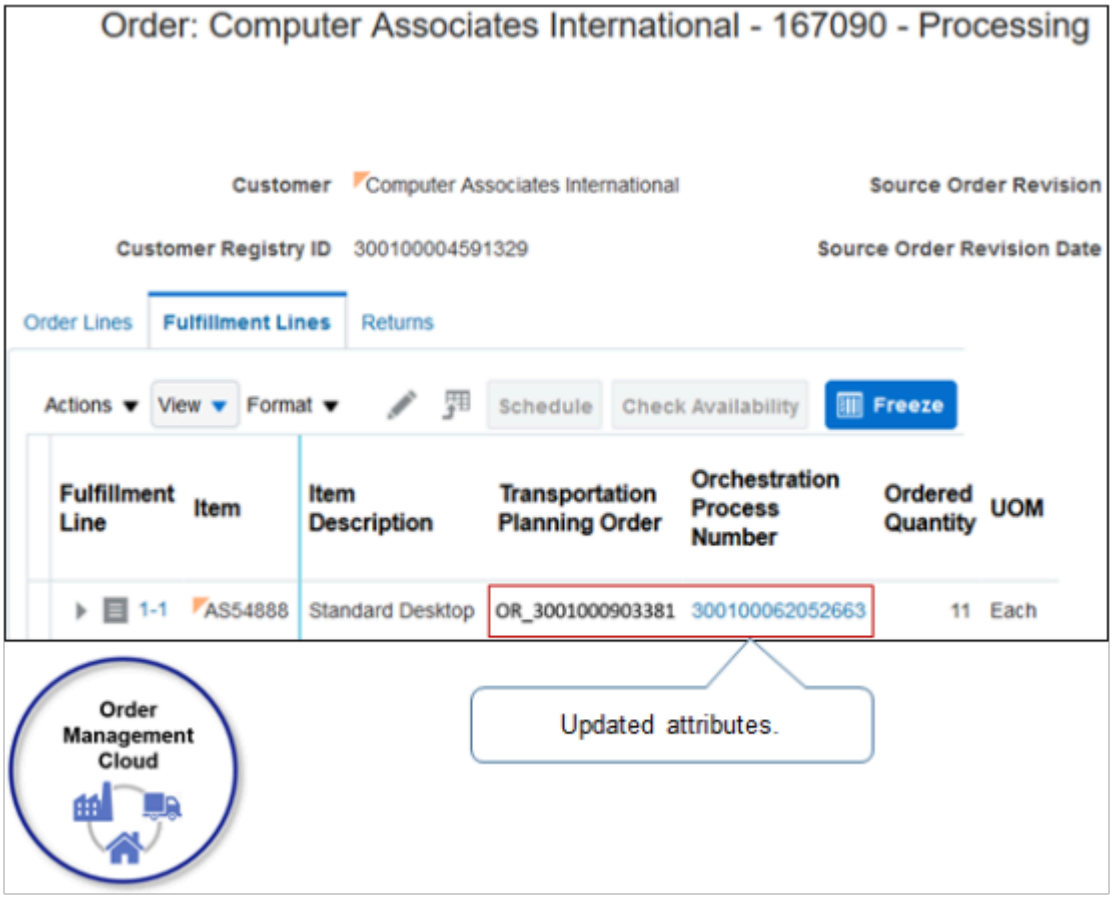
- Inventory Management sends a screening request through Integration Cloud Service to Global Trade Management. You can also set up Order Management to do trade compliance screening before submitting the sales order, and you can use Inventory Management to screen during shipping.
- Global Trade Management screens the party for trade compliance according to request details that you set up in Inventory Management. For example, you can request to screen shipment lines on a specific step during the fulfillment process according to your business requirements, such as after fulfillment staged fulfillment lines but hasn't shipped them. Global Trade Management can also screen the transaction for export controls, and can screen the party and transaction for licensing.
- Global Trade Management sends screening results through Integration Cloud Service to Inventory Management.

4. **Orchestrate Order**. Order Management orchestrates fulfillment for the sales order, including creating a shipment request in Shipping.
5. **Fulfill Order**. Transportation Management receives a transportation management request for the fulfillment line from Order Management, acknowledges the request, and converts it to an order release.

   - Transportation Management also creates a transportation planning order.
   - Transportation Management examines the supply chain, creates an optimal transportation shipment for one or more fulfillment lines, creates a transportation plan, then creates and sends a shipment request to Oracle Shipping.
   - Transportation Management can create different types of direct or consolidated shipments, including single origin truckload, single destination truckload, multistop truckload, less than truckload (LTL), parcel, ocean, rail, and so on.

The Order Management work area updates the screening status on the Order page during orchestration. It also updates the transportation planning order and orchestration process number on the Fulfillment Lines tab, and

it updates the status of the orchestration plan on the Orchestration Process tab with the details it receives from Transportation Management.



6. **Pick and Pack**. Oracle Shipping picks and packs the sales order in the warehouse.
7. **Create Export Documents**. Optional. If the flow includes Global Trade Management, then it creates export documents, if necessary, and also creates the Shipment Trade Transaction. As an option, it can also screen for compliance according to how you set it up for screening during order fulfillment, and then send screening results and reason to Shipping.
8. **Send Tender to Carrier**. Oracle Transportation Management creates and sends the tender to carrier. If necessary, Global Trade Management declares the export.
9. **Ship Item**. Oracle Shipping ships the item and sends the shipment advice to Order Management and Transportation Management. The customer receives and acknowledges receipt, then Transportation Management sends shipment delivery confirmation to Order Management. Order Management sends the fulfillment line to Oracle Accounts Receivable for invoicing, then sets the fulfillment line status to Closed.

Related Topics

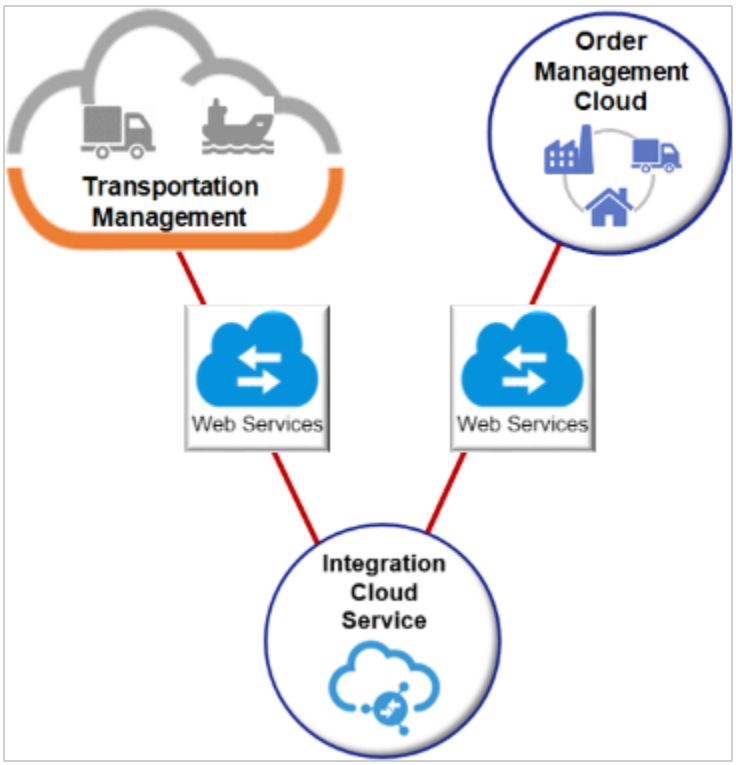
- Overview of Integrating Order Management with Transportation Management
- Integrate Order Management with Transportation Management
- Use Integration Cloud Service with Order Management

Guidelines for Integrating Order Management with Transportation Management

Set up an orchestration process, manage connectors, business rules, constraints, and use web services to integrate Oracle Order Management with Oracle Transportation Management.

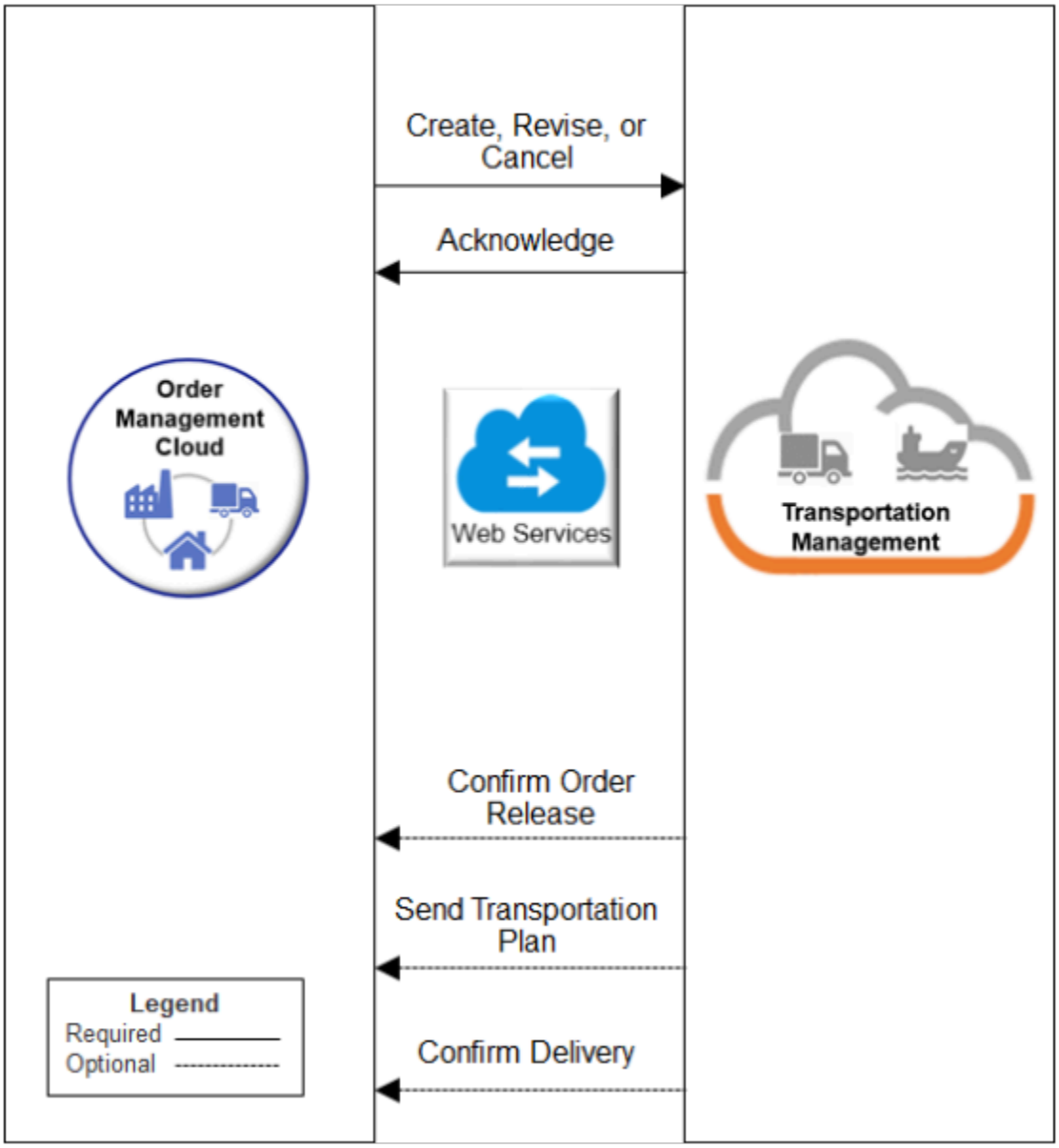
Use Web Services

Order Management doesn't come predefined to integrate with Transportation Management or Global Trade Management. Starting in Update 13, you can use Integration Cloud Service, a synchronous web service, or some other asynchronous web service, to integrate with your transportation management system. Before Update 13, you must use only an asynchronous web service.



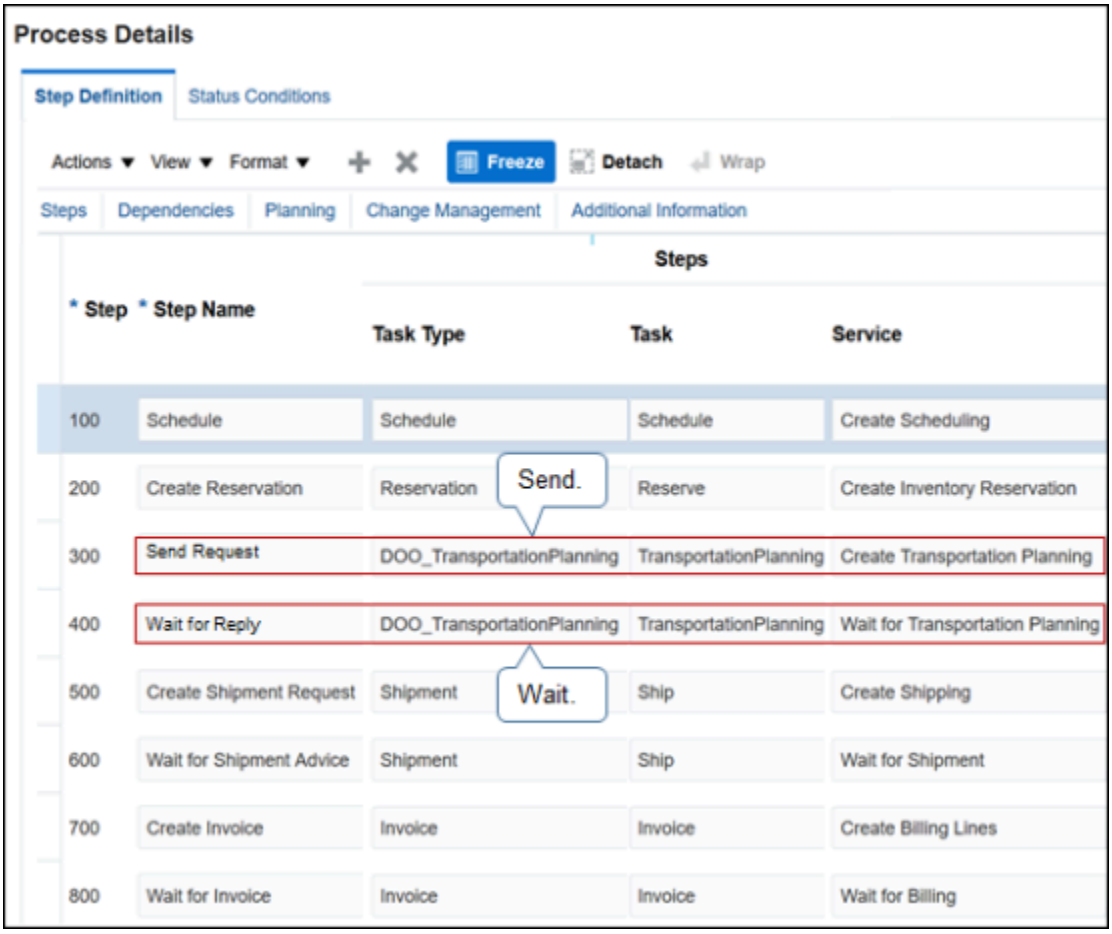
Communicate details through a web service.

- Required.
  - Send details from Order Management to Transportation Management when the user creates, revises, or cancels a sales order.
  - Send `acknowledgement` from Transportation Management to Order Management.
- Optional.
  - As an option, send fulfillment details from Transportation Management to Order Management. For example, confirm that Transportation Management created the order release, send transportation plan that includes updated schedule dates, or send delivery confirmation details, such as delivery date.



Set Up Orchestration Process

Add an orchestration process step that sends a request to Transportation Management. Add another step that waits for the reply.



Note

- Use the predefined DOO_TransportationPlanning task type in the orchestration process. This task type allows Order Management to send a new request to the transportation management system, and to update or cancel a request that already exists.
- Reference DOO_TransportationPlanning at any point in the stepwise sequence of an orchestration process, depending on your business requirements. For example, set up an orchestration process so it identifies the source of supply, reserves inventory, then plans transportation.

Wait for the response.

- It might be necessary to wait for a response from Transportation Management in some situations. For example, an order release is a representation of the sales order in Transportation Management. Transportation Management uses the sales order as input to create the order release.

  If Transportation Management can't successfully create the release, for example, an attribute that it requires for planning transportation is missing a value, then it might be useful to prevent the orchestration process from proceeding, such as to a ship item step, until the Order Manager can determine whether to allow the process to proceed without the release.
- To add a wait step, copy the Send Request step, then modify the copy as necessary.
- Use the DOO_TP_REQ_ACCEPTED value or the CANCELED value in the Status Code attribute as the exit criteria for the wait step.

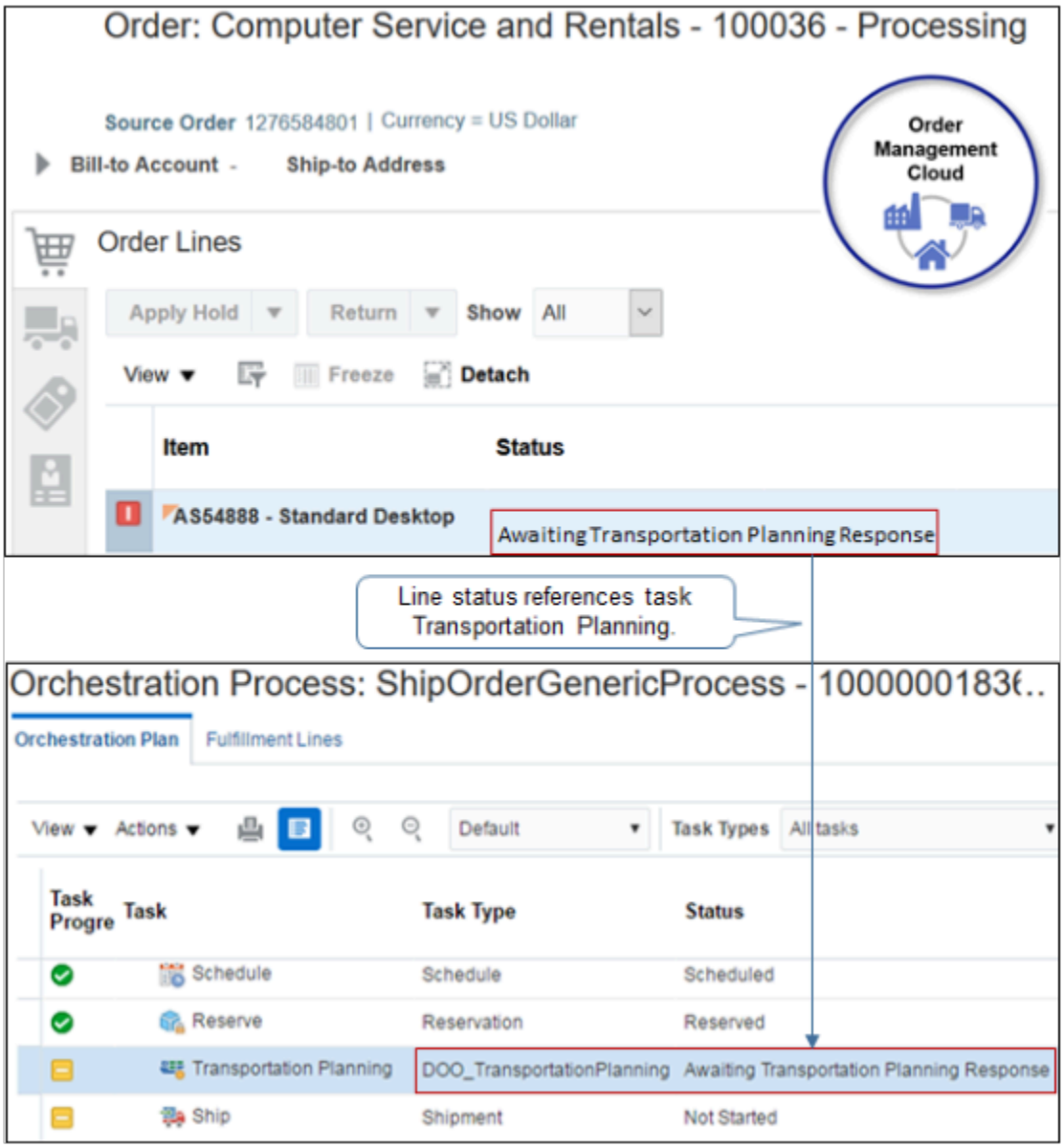
- The flow won't proceed to the next step until Transportation Management replies that it successfully created the sales order release.
- Don't wait for a response unless you need it to meet a business requirement. Waiting for the response increases the possibility that the orchestration process might take a long time to finish, or get stuck waiting for the response.

Process the response.

- Process the response from Transportation Management at different steps in the orchestration process, such as after the transportation plan finishes, or after Transportation Management sends proof of delivery. Order Management can use the response after the transportation plan finishes even if the orchestration process isn't currently running the DOO_TransportationPlanning task.
- DOO_TransportationPlanning uses an indicator to mark each fulfillment line that Transportation Management planned so some other system can process it.

  For example, the predefined integration with Oracle Inventory sets a transportation planning hold on each shipment request that references a fulfillment line that contains this indicator. The integration releases the hold only after Transportation Management receives the transportation shipment plan.

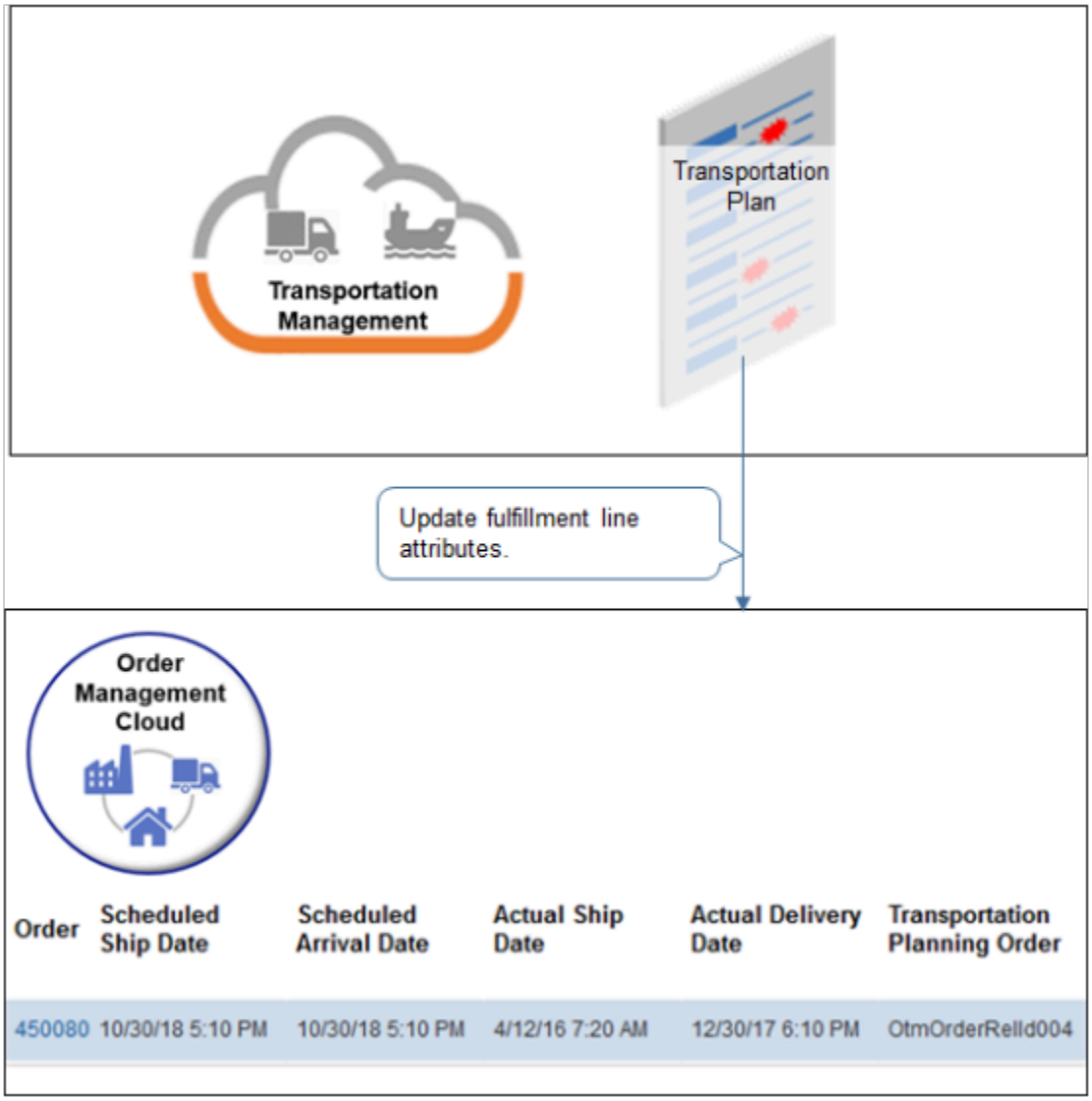
At run time, the Status attribute on the order line references DOO_TransportationPlanning on the orchestration process to get the status for Awaiting Transportation Planning Response.



- The DOO_TransportationPlanning task and Shipment task come predefined to use different attributes to identify change. You must modify them so they use the same attributes. For details, see *Manage Order Attributes That Identify Change*.
- If your orchestration process includes branching, and if you include the Shipment task in one or more branch, and if you expect transportation and shipping will process the order line, then you must include the DOO_TransportationPlanning task in each branch that includes the Shipment task. For Example, if you include Shipment in branch x for normal shipping and branch y for back-to-back shipping, then you must include DOO_TransportationPlanning in x and y instead of including DOO_TransportationPlanning in a step that happens before the branch.
- Make sure the DOO_TransportationPlanning task and the Shipment task use the same compensation pattern. For details, see *Overview of Managing Change That Occurs During Order Fulfillment*.

Get Transportation Plan

As an option, get details about the transportation plan from Transportation Management to update attributes in Order Management.



Note

- Get updated scheduled dates and order release number from Transportation Management.
- Prevent Order Management from updating the sales order after it sends it to Transportation Management. See the section about constraints later in this topic.

Note these points about getting the transportation plan and delivery confirmation.

- The transportation plan uses the DOO_TP_PLANNED value in the TaskInstanceStatusCode attribute.
- The transportation plan updates the Scheduled Ship Date attribute and the Scheduled Arrival Date attribute.
- If Transportation Management splits the transportation plan into two separate shipments, then Transportation Management sends the latest date of these shipments to Order Management. For example, if shipment

one delivers on January 1, and shipment two delivers on January 15, then the transportation plan that Transportation Management sends to Order Management will include only the latest date of the finished shipment, which is January 15.

- If the orchestration process receives the transportation plan when its not on a wait step, then it will reject the plan and will reject the request to update.

  To make sure the orchestration process reaches a wait step, allow about 10 minutes between the most recent update that Order Management sends to Transportation Management, and the time when Transportation Management sends the plan update to Order Management.

Get Delivery Confirmation

As an option, add a pause step to make sure invoicing finishes before confirming delivery.

- Add a pause step that's similar to the Shipping step in the orchestration process.
- Get delivery confirmation at different points in the orchestration process, such as Awaiting Billing.

  For example, to prevent the Invoice step from running before confirming delivery, add a pause step before the invoice step, then set up the pause so it waits until Order Management receives delivery confirmation.
- Use the SAC_SYSTEM_EVENT_POD_PAUSE event and the TransportationPlannedFlag attribute.
- Use SAC_SYSTEM_EVENT_POD_PAUSE to release the pause when Transportation Management sends the DOO_TP_DELIVERED status.
- Make sure the orchestration process is on a wait step when it receives the response.
- The status description for DOO_TP_DELIVERED is Shipment Delivered. However, if you use a pause to get delivery status, then Shipment Delivered doesn't display. Instead, set up a pause status that displays Shipment Delivered.

Click **Click for Rule** on the pause step, then create two pause rules.

Order Management releases SAC_SYSTEM_EVENT_POD_PAUSE after it receives delivery confirmation so you can use this event to process a subsequent sales order.

Rule 1

Create the rule that releases the pause task.

```
If
"D" equals ignore case DooSeededOrchestrationRules.DOOFLine.TransporationPlannedFlag
then
assert new DooSeededOrchestrationRules.SacResult
 (sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_IMMEDIATE)
```

where

Code	Description
`"D"`	D means Delivered.

Code	Description
`If "D" equals ignore case DooSeededOrchestrationRules.DOOFI`	If the TransporationPlannedFlag attribute on the fulfillment line contains D.
`assert new DooSeededOrchestrationRules.SacRe (sacType:DooSeededOrchestrationRu TYPE_IMMEDIATE)`	Then immediately release the pause task and continue to the next step in the orchestration process.

Rule 2

Create the rule that pauses the orchestration process.

```
If
NOT ("D" equals ignore case DooSeededOrchestrationRules.DOOFLine.TransporationPlannedFlag )
then
assert new DooSeededOrchestrationRules.SacResult
 (eventName:DooSeededOrchestrationRules.SacResult.SAC_SYSTEM_EVENT_POD_PAUSE,
 reevaluateFlag:"Y",sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_EVENT)
```

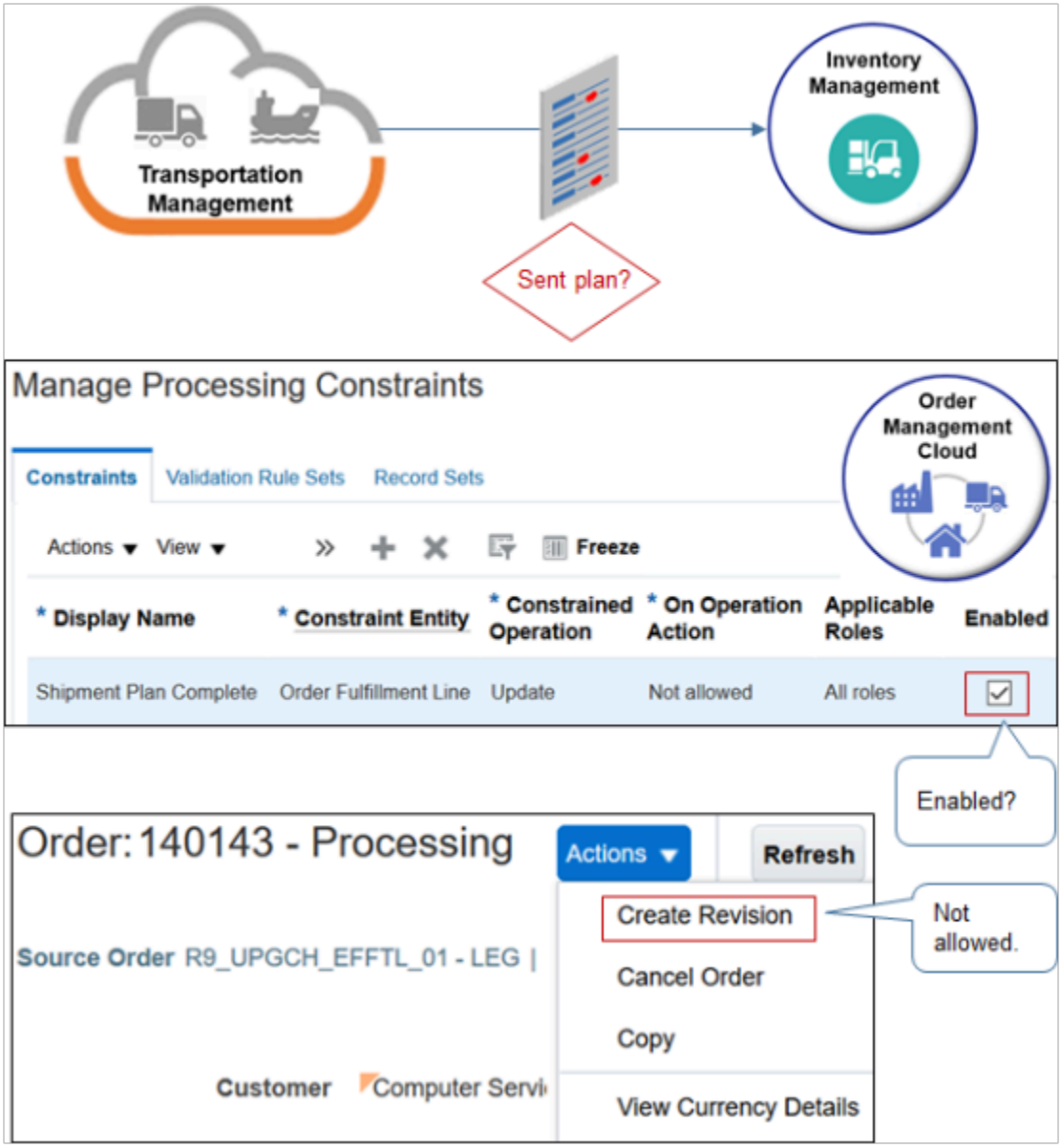
where

Code	Description
`"D"`	D means Delivered.
`If NOT ("D" equals ignore case DooSeededOrchestrationRules.DOOFI`	If TransporationPlannedFlag on the fulfillment line doesn't contain D.
`assert new DooSeededOrchestrationRules.SacRe (eventName:DooSeededOrchestratior SYSTEM_EVENT_POD_PAUSE, reevaluateFlag:"Y", sacType:DooSeededOrchestrationRul TYPE_EVENT)`	Then • Use the SAC_SYSTEM_EVENT_POD_PAUSE business event to pause the orchestration process. • Use **`reevaluateFlag:"Y"`** to evaluate the condition every time the orchestration process finishes a step. If the condition doesn't evaluate to true, then release the pause task. • Use SAC_TYPE_EVENT to pause the pause task until Order Management releases SAC_SYSTEM_EVENT_POD_PAUSE.

For details about how to create a pause rule, see *Pause Orchestration Processes Until Events Happen*.

Prevent Changes After Transportation Planning Finishes

If you use Oracle Inventory Management before Update 18C, then enable the predefined Shipment Plan Complete processing constraint.



Note

- Shipment Plan Complete prevents the user from revising the order line in the Order Management work area after Transportation Management sends the transportation plan to Inventory Management. This constraint comes predefined as disabled. You can enable it during set up.

  Before Update 18C, Inventory Management supports only a quantity change or order cancel after Transportation Management sends the transportation plan to Inventory Management. Inventory Management doesn't support any other order revisions before Update 18C.

  Inventory Management supports revisions starting with Update 18C. You can disable Shipment Plan Complete after you upgrade to Update 18C.
- Order Management uses the transportation plan to determine whether Transportation Management already sent the transportation plan to Shipping.

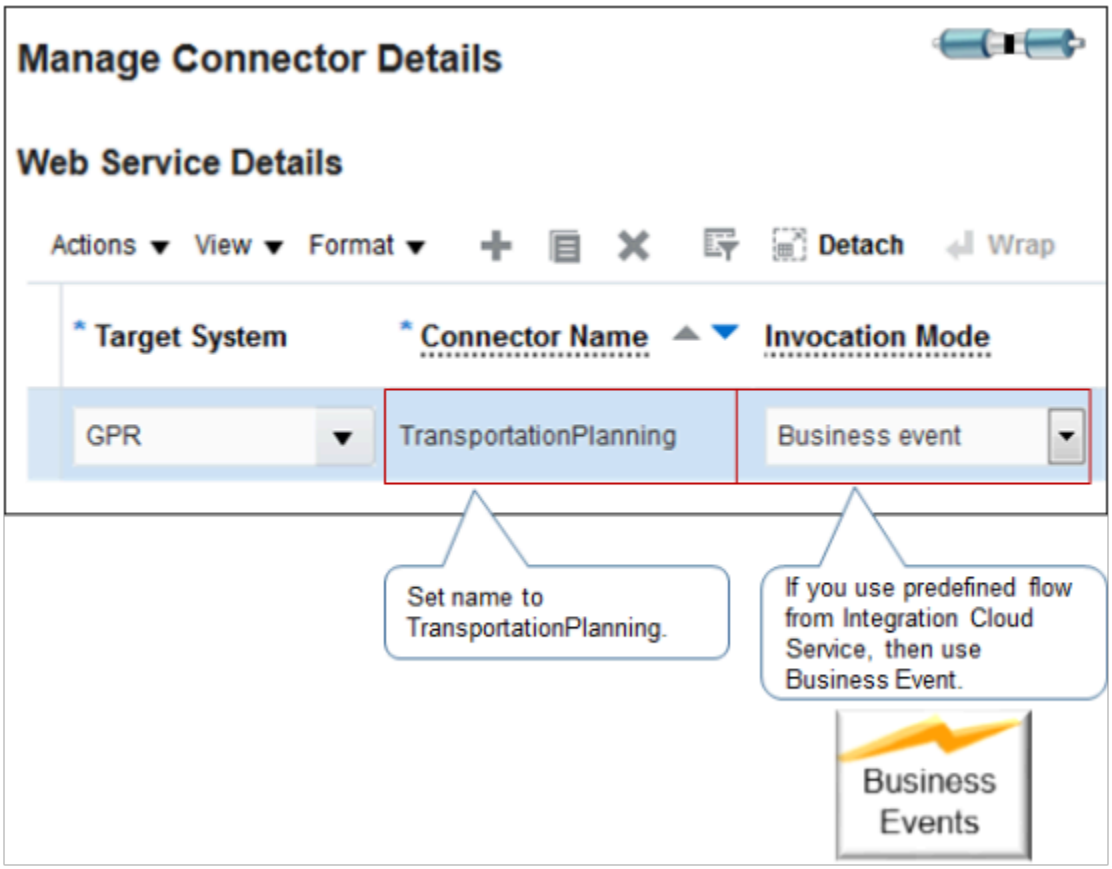
- If you don't use Oracle Inventory Management, then don't enable the Shipment Plan Complete constraint.

For details, see *Manage Processing Constraints*.

Set Up Integration

Set Up Connector

Set up a connector on the Manage Connector Details page.



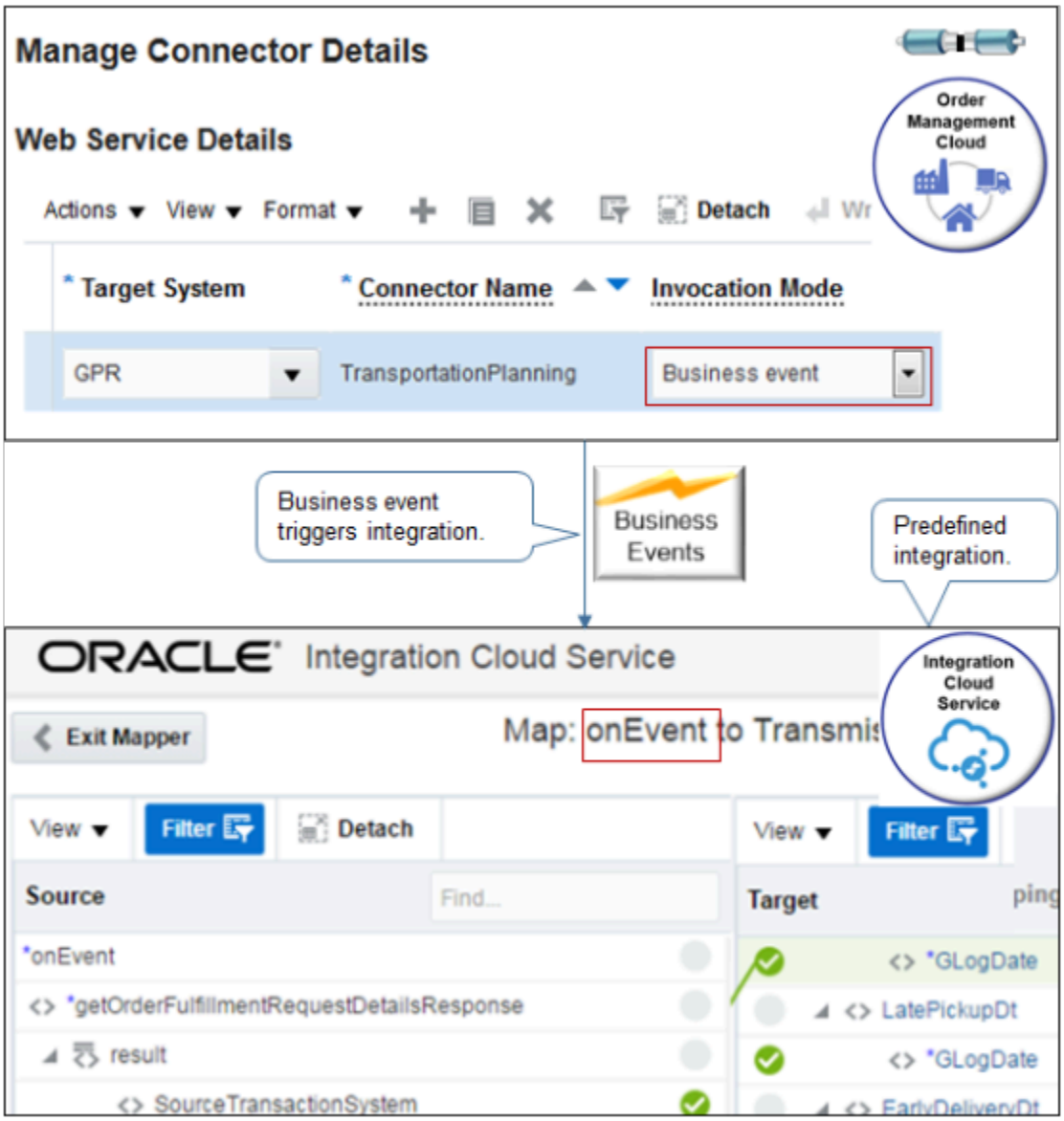
Note

Attribute	Description
Connector Name	Name the connector TransportationPlanning. Order Management uses this name at various locations in the flow. Using TransportationPlanning will help you identify the connector involved in the flow that you're setting up or during troubleshooting.
Invocation Mode	If you. • Use the predefined integration in Integration Cloud Service, then set Invocation Mode to Business Event. • Don't use the predefined integration, then set Invocation Mode to Synchronous or Asynchronous.

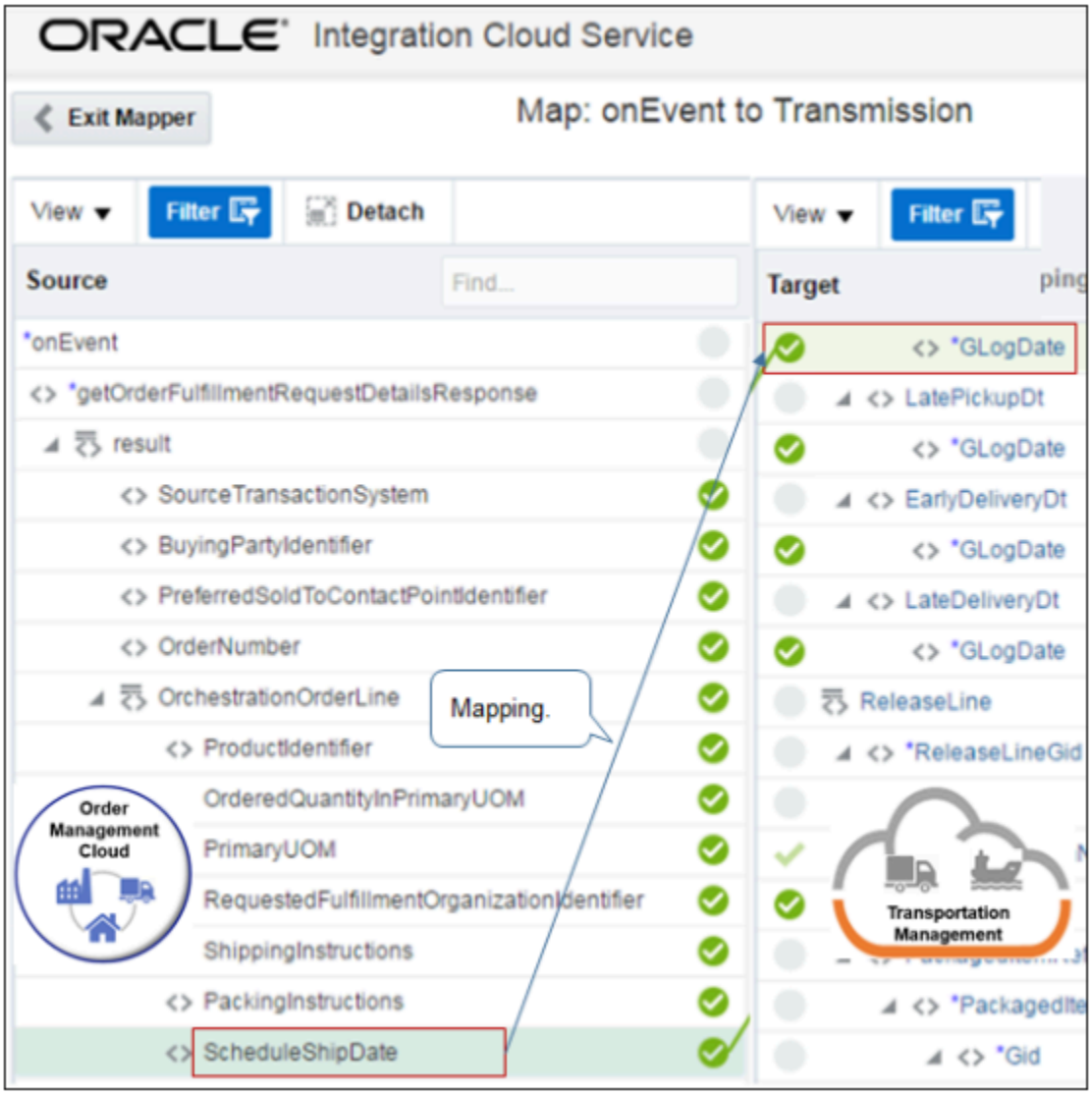
Attribute	Description
User Name	Make sure the user has the DOO_MANAGE_WEB_SERVICE_INTERFACE_TO_TRANSPORTATION_DATA_FOR_SALES_ORDER_PRIV privilege. For details about how to set up privileges, see *Security Reference for Order Management*.

Use Predefined Integration in Integration Cloud Service

Use a predefined integration in Integration Cloud Service as a starting point for your integration. Use the TransportationPlanning connector to connect through a business event.



For example:



Note

- A business event triggers the integration.
- One fulfillment line in Order Management maps to one order release plus one order release line in Transportation Management. For example, the integration maps the ScheduleShipDate attribute in Order Management to the GLogDate attribute in Transportation Management.
- Modify the predefined integration to meet your business requirements. For example, to group fulfillment lines in a shipment set or configured item, group them into a single order release in Transportation Management.
- Transportation Management sends a status update to Order Management for each order release line that maps to one fulfillment line.

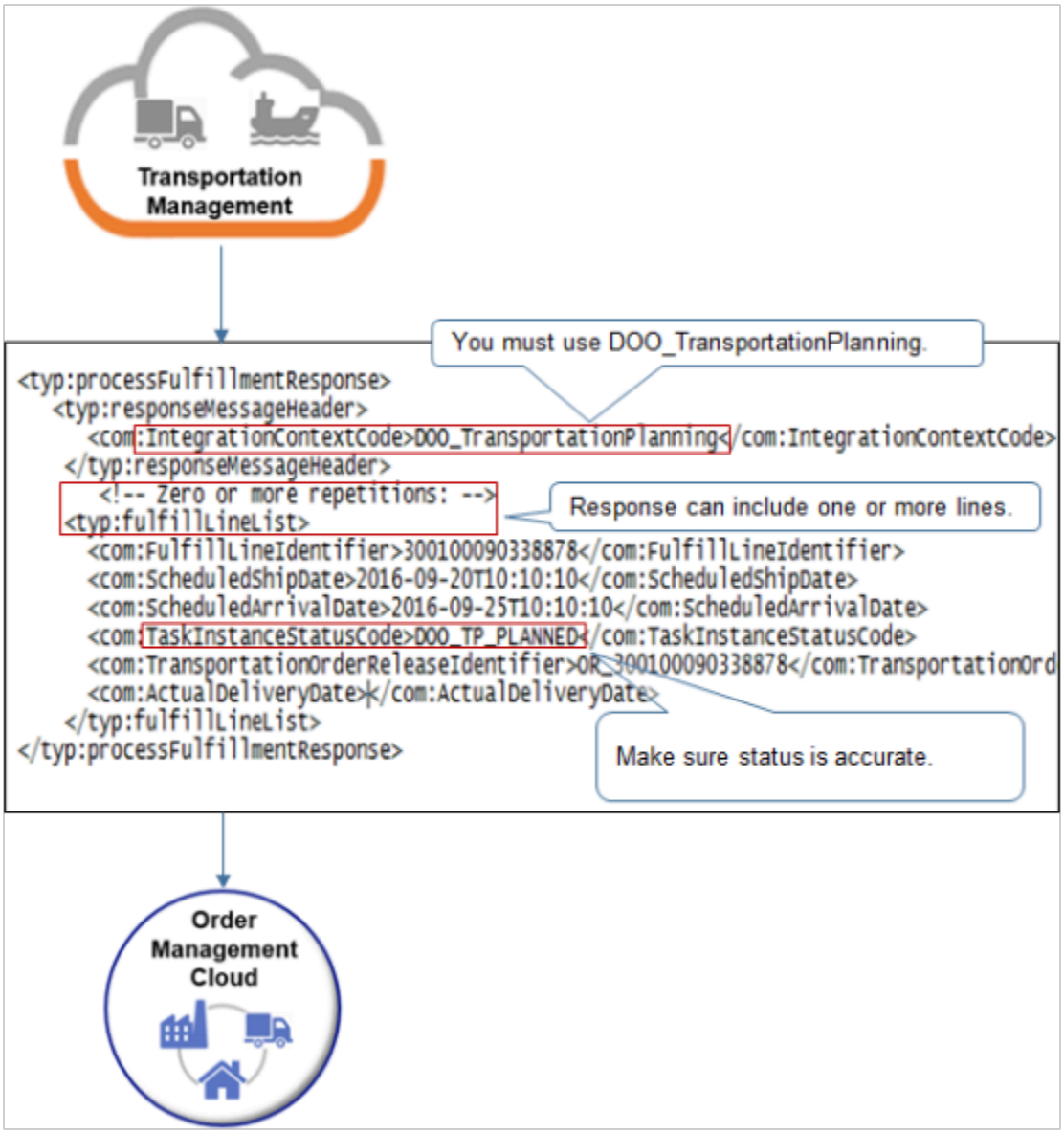
Examine predefined integrations that use Integration Cloud Service.

- Go to *Oracle Logistics Cloud to Oracle Fusion Cloud SCM Integration* and read the content.
- Click **Get App** to access *Integration with OTM or GTM using ICS (Doc ID 2209248.1)*.

- Import one of the example integrations into Integration Cloud Service. For example, import integration OTM_OM, Communicate Order Shipping Information to Order Management.
- Examine the integration in the Integration Cloud Service user interface.

Example Response Payload

Here's an example of the response that Transportation Management sends to Order Management.



Note

- The predefined integration in Integration Cloud Service uses this structure. If you modify it, then use this payload as an example of the data that your fulfillment system must send.
- The response can include one or more fulfillment lines.
- The IntegrationContextCode attribute must equal the DOO_TransportationPlanning task type.

- Make sure the TaskInstanceStatusCode attribute contains a value that accurately reflects the status, such as DOO_TP_PLANNED. For details about the statuses that you can use, see *Integrate Order Management with Transportation Management*.

- The screen print truncates the TransportationOrderReleaseIdentifier line. Here's the full line.

```
<com:TransportationOrderReleaseIdentifier>OR_300100090338878</com:TransportationOrderReleaseIdentifier>
```

Related Topics

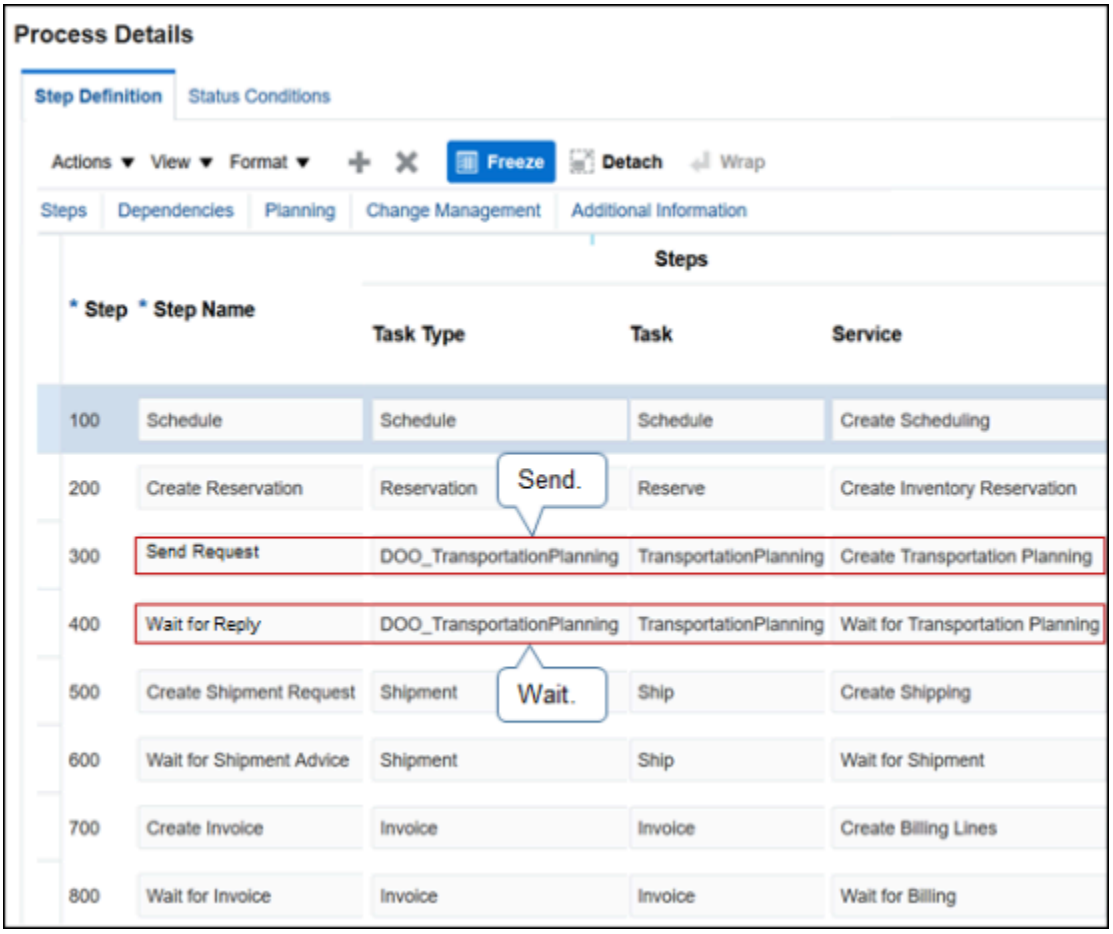
- Overview of Integrating Order Management with Transportation Management
- How Order Management Integrates with Transportation Management
- Use Integration Cloud Service with Order Management
- Overview of Orchestration Processes

Integrate Order Management with Transportation Management

Use Integration Cloud Service to create an integration between Order Management and Transportation Management.

You can also set up an orchestration process so it does Transportation Management tasks.

In this example, you create an orchestration process so it sends the request for transportation planning after the fulfillment system identifies a source and reserves inventory.



This topic uses example values. You might need different values, depending on your business requirements.

Try it.

1. Use Integration Cloud Service to create an integration between Order Management and Transportation Management.

   Make sure you set the business event trigger points that you need to start the integration process. For details, see *Use Integration Cloud Service with Order Management*.

2. Create the orchestration process.
    - Make sure you have the privileges that you need to administer Order Management.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Orchestration Process Definitions
    - On the Manage Orchestration Process Definitions page, search for ShipOrderGenericProcess.
    - In the search results, click **Actions > Duplicate**.

      You can integrate Order Management with Transportation Management only through a long-running task.
    - In the Edit Orchestration Process Definition dialog, set the values, then click **Save**.

Attribute	Value
Process Name	Enter any text. For this example, enter `ShipOrder_PlanTransportation`.
Process Display Name	Enter any text. For this example, enter `Ship Order and Plan Transportation`.
Process Class	Ship Order Class

3. Add status conditions.
    - Click **Status Conditions**.
    - In the Orchestration Process Status Values area, add status values.

Attribute	Value
Awaiting Transportation Planning Response	`"Transportation Planning" = "DOO_TP_AWAIT_ACCEPT"` Use this expression as an exit condition for the Wait step.
Transportation Planning Request Accepted	`"Transportation Planning" = "DOO_TP_REQ_ACCEPTED"`
Transportation Planning Request Rejected	`"Transportation Planning" = "DOO_TP_ REQ _REJECT"`
Transportation Planning Completed	`"Transportation Planning" = "DOO_TP_PLANNED"`

4. On the Step Definition tab, add a step at any location in the sequence of steps.

Attribute	Value
Step Name	Send Request to Transportation Management
Step Type	Service
Task Type	DOO_TransportationPlanning
Task	TransportationPlanning
Service	Create Transportation Planning
Update Service	Update Transportation Planning
Cancel Service	Cancel Transportation Planning
Use Transactional Item Attributes	Contains a check mark.
Use Flexfield Attributes	Contains a check mark.

This step sends the sales order to the fulfillment system for transportation planning. For this example, add this step immediately after the Create Reservation step.

5. Optional. Add this step immediately after the Send Request to Transportation Management step.

Attribute	Value
Step Name	Wait for Transportation Management
Step Type	Service
Task Type	DOO_TransportationPlanning
Task	TransportationPlanning
Service	Wait for Transportation Planning Validation

Attribute	Value
Exit Criteria	Add a check mark to each criteria. o Canceled o Transportation Planning Request Accepted This exit criteria causes the orchestration process to wait until the Transportation Planning Request Accepted status equals Y, or until Order Management cancels the request, such as through a time out.
Next Expected Task Status	Transportation Planning Request Accepted
Use Transactional Item Attributes	Contains a check mark.
Use Flexfield Attributes	Contains a check mark.

Note

- This step temporarily pauses the orchestration process so Transportation Management can validate the planning request before the orchestration process creates the shipment request.
- Set up your orchestration process so it processes the response while waiting on a pause step.
- If you don't add this step, then the orchestration process will continue to run and fulfill the sales order without waiting for validation from Transportation Management.

Related Topics

- Overview of Integrating Order Management with Transportation Management
- How Order Management Integrates with Transportation Management
- Overview of Orchestration Processes
- Use Integration Cloud Service with Order Management

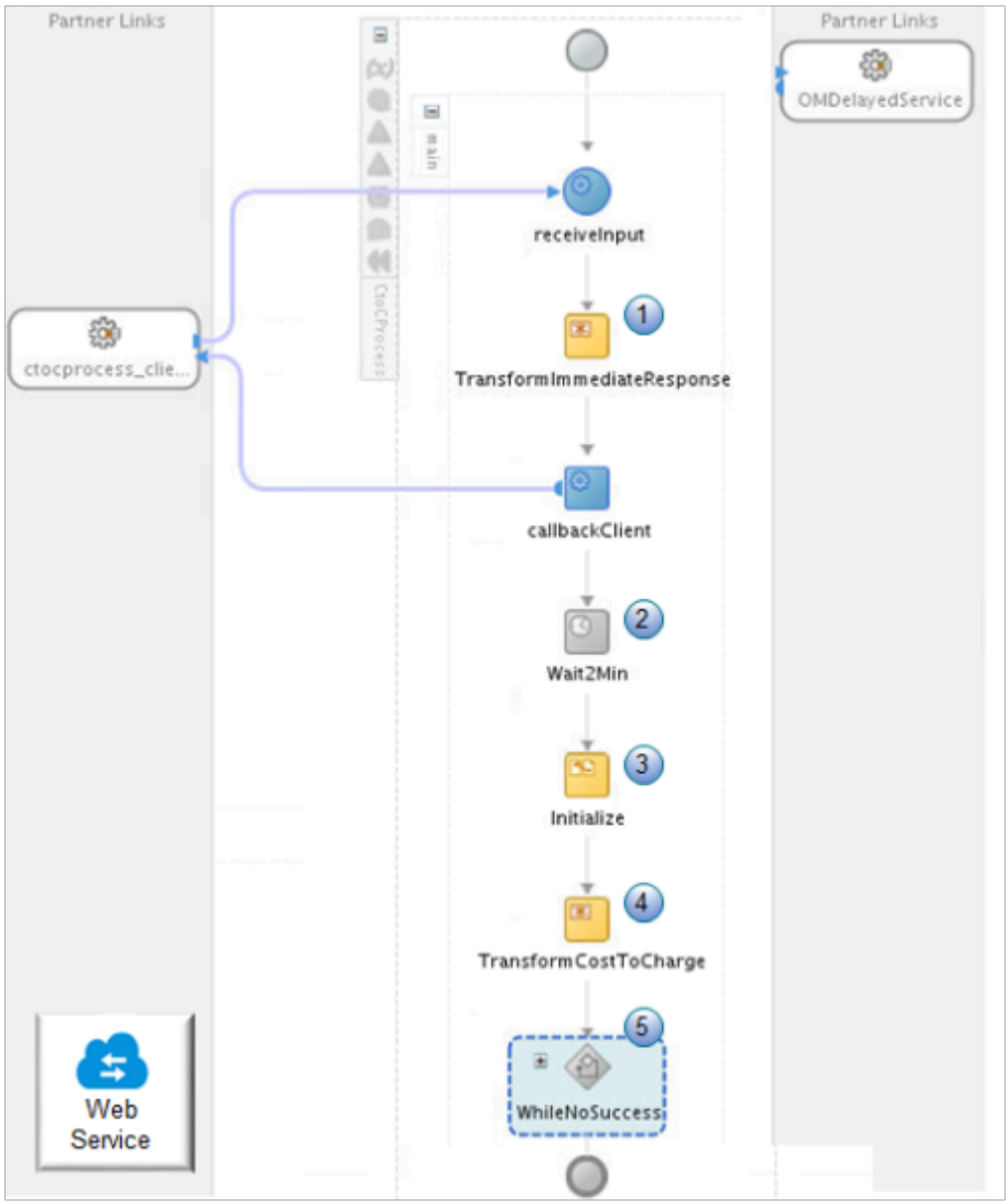
Shipping

Convert Shipment Costs to Freight Charges

Use your own task to convert shipment costs to freight charges.

1. Create your own task. For details, see *Create Your Own Task Type*.
2. Add a new step in your orchestration process that references your new task. Place it after the step that completes the shipment task and before the step that starts the invoice task. The published payload for your task will contain the shipping cost that you capture in the shipping system.

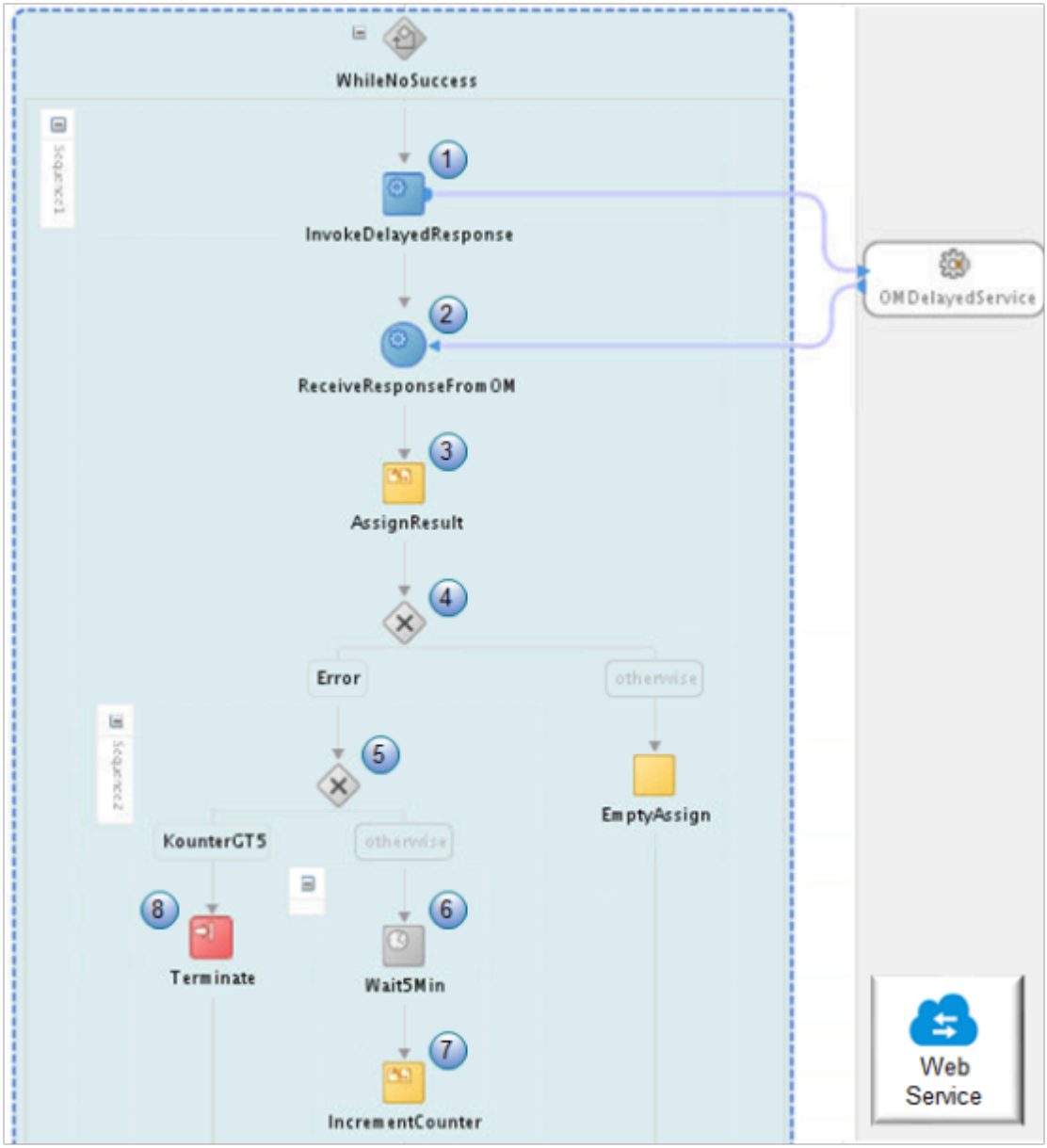
3. Set up your connector. Here's what it looks like in Oracle JDeveloper (Java Developer).



Note

1. Transform your input into an immediate response and send it to Order Management.
2. Wait for two minutes.
3. Initialize local variables, such as Result and Counter.
4. Transform your input into a delayed response. It converts cost to charge, including a mark up, if necessary.

5. Send a delayed response, then wait for a SUCCESS reply.
   Make sure you also implement error handling to handle an ERROR reply.
   Here's an expanded view of step 5, WhileNoSuccess.



6. Send the delayed response that contains charges back to Order Management. The diagram doesn't include this step.

The connector updates these charges in Order Management, then sends them to invoicing.

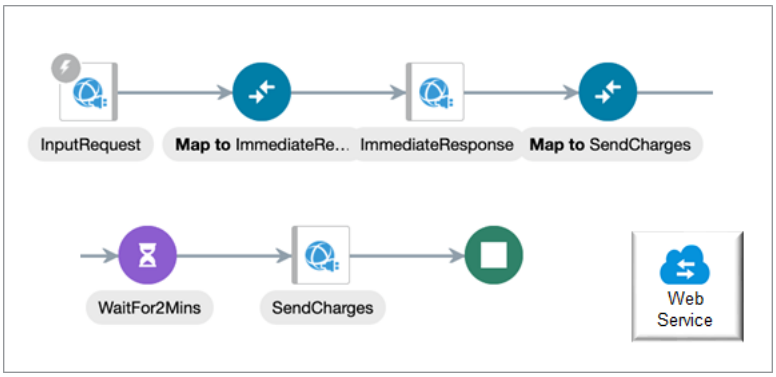
Note

1. Send charges in a delayed response to Order Management.
2. Receive a reply from Order Management. The reply contains SUCCESS or ERROR.
3. Save the results of the reply in a local variable.
4. This is the If condition that routes the flow depending on the value of the local variable, SUCCESS or ERROR.

5. This is the If condition that routes the flow depending on the Counter local variable that counts the number of attempts.
6. Depending on the error that you receive, wait five minutes, then call the service in Order Management again.
7. Increment the Counter variable.
8. If Counter is equal to or greater than 5, then end the flow.

   You can modify the value in Counter to meet your needs.

As an alternative, you can use Integration Cloud Service instead of JDeveloper. Here's what your connector will look like in Integration Cloud Service.



For details see, *Use Integration Cloud Service with Order Management*.

Guidelines

- Set up your task as a long-running task that uses a wait step in your orchestration process.
- Send the existing charges back through the task response in addition to the new charges that you're adding for freight.
- Make sure you use only one primary charge for each ChargeAppliesTo on each fulfillment line. For details, see *Manage Pricing Charge Definitions*.
- Make sure you set the Applies To attribute to SHIPPING for each new charge that you add.
- If you split a shipment line, and if you capture more than one shipping cost in one fulfillment line, then sum the costs and send a single freight charge through your task.
- If you must calculate tax on the freight charges, see *Integrate Order Management with Accounts Receivable*.
- This set up captures the shipping cost when you confirm the shipment.
- This set up adds a shipping charge only to the order line. It doesn't add the shipping charge to the sales order total. The total on the invoice won't match the sales order total, but Order Management does send these charges to Accounts Receivable.

  Assume your sales order has one order line, the line has a price of $10 each, a quantity of 10, and the line's total is $100. Assume you add a $9 freight charge, which results in a line total of $109 but an order total of $100. Order Management will send $109 to Accounts Receivable.
- You must test your set up in a test environment for scalability and performance before you deploy it to your production environment.
- You can use this set up only on the order line that you're orchestrating. You can't use it for an entire sales order.

Make sure the user who calls the delayed response service has these privileges:

- Manage Orchestration Order Template Interface Web Service

- Manage Orchestration Order Fulfillment Interface Web Service

Download Example Files

Some of the payloads that you use in this topic are too long or complex to display in this document, but you can download them as files.

1. Go to *Technical Reference for Order Management (Doc ID 2051639.1)*.
2. Download the Payloads and Files attachment.
3. Use the files in the attachment.

File	Description
CostToChargeMappings.xlsx	Example mapping that the flow uses for a delayed response.
CostToChargeTTLRequest.xml	Example payload that requests to convert costs to freight charges. The PriceAdjustmentTL tag contains the cost that the flow records when it confirms shipment.
CostToChargeV4.xls	Use this example transformation style sheet to convert CostToChargeTTLRequest.xml to CostToChargeDelayedResponse.xml.
convert_ship_cost_to_freight_charge_input_payload.txt	Example input payload.
convert_ship_cost_to_freight_charge_delayed_response_payoad.txt	Example payload for a delayed response.

This topic describes most but not all of the attributes that you need to map. For the complete list, see CostToChargeMappings_V5.xlsx.

Nomenclature

We use indentation to indicate the hierarchy that you must use in the payload. For example:

```
Headers
 SOAPHeaders
 ReplyTo
 Address
```

Here's how you read this hierarchy:

- The Address attribute is in the ReplyTo entity.
- The ReplyTo entity is in the SOAPHeaders entity.
- The SOAPHeaders entity is in the Headers entity.

We use **`bold font`** to indicate that if the attribute isn't available in the destination, then don't map it.

Immediate Response

We recommend that you set the Invocation Mode attribute on the connector to Asynchronous Service.

You must map these values between your request payload and your response payload when you make an asynchronous call for an immediate response:

Request Payload	Response Payload

`Headers`
`SOAPHeaders`
`ReplyTo`
`Address`
	`Headers`
	`SOAPHeaders`
	`ReplyTo`
	`Address`

`Headers`
`SOAPHeaders`
`MessageID`
	`Headers`
	`SOAPHeaders`
	`MessageID`

Here are the rest of the values that you need to map on the order header.

Request Payload	Response Payload
`headerTL` `HeaderId`	`headerTR` `HeaderId`
`SUCCESS`	`headerTR` `Status`

Fulfillment Line Entity

Map these values for each fulfillment line entity.

Request Payload	Response Payload
`headerTL` `FulfillLineTLVO1`	`headerTR` `FulfillLinesTR`
`FulfillLineId`	`FulfillLineId`
`headerTL` `SourceOrderSystem`	`SourceOrderSystem`
`AWAIT_RESPONSE` You can use some other status. This is typically the intermediate status for the task type.	`StatusCode`
`headerTL` `TaskTypeCode`	`TaskType`

Example Payload

Here's an example payload from the CostToChargeImmediateResponse.xml file.

```
<nstrgmpr:HeaderId>300100135839047</nstrgmpr:HeaderId>
<nstrgmpr:FulfillLinesTR>
<nstrgmpr:FulfillLineId>300100135839049</nstrgmpr:FulfillLineId>
<nstrgmpr:SourceOrderSystem>GPR</nstrgmpr:SourceOrderSystem>
<nstrgmpr:StatusCode>AWAIT_RESPONSE</nstrgmpr:StatusCode>
<nstrgmpr:TaskType>ConvertCostToCharge</nstrgmpr:TaskType>
</nstrgmpr:FulfillLinesTR>
</nstrgmpr:headerTR>
```

Delayed Response

You must map values between your request payload and your response payload for a delayed response.

Fulfillment Line Entity

Map these values for each fulfillment line entity.

Request Payload	Response Payload
`headerTL FulfillLineTLVO1`	`Fline`
`FulfillLineId`	`FulfillLineId`
`headerTL SourceOrderSystem`	`SourceOrderSystem`
`headerTL TaskTypeCode`	`TaskType`
Include the appropriate status. This is typically the exit criteria for the task type.	`Status`
`headerTL FulfillServiceProvider` You can also provide the name of a connector instead.	`CallbackConnectorName`

Charge Entity on the Fulfillment Line

Map these values for each charge entity on the fulfillment line.

Request Payload	Response Payload
`OrderChargeTL`	`FulfillLineOrderChargeSDOTR`
`AvgUnitSellingPrice`	`AvgUnitSellingPrice`

Request Payload	Response Payload
ChargeAppliesTo	ChargeAppliesTo
ChargeCurrencyCode	ChargeCurrencyCode
ChargeDefinitionCode	ChargeDefinitionCode
ChargeSubtypeCode	ChargeSubtypeCode
ChargeTypeCode	ChargeTypeCode
FreightReferenceLineId	FreightReferenceLineId
ParentEntityCode	ParentEntityCode
PricedQuantity	PricedQuantity
PricedQuantityUomCode	PricedQuantity unitCode
PricedQuantityUomCode	PricedQuantityUOMCode
PricePeriodicityCode	PricePeriodicityCode
PriceTypeCode	PriceTypeCode
PrimaryFlag	PrimaryFlag
ReferenceOrderChargeId	ReferenceOrderChargeId If Y then true, else false.
RollupFlag	RollupFlag If Y then true, else false.
SequenceNumber	SequenceNumber
SourceChargeId	SourceChargeId
CanAdjustFlag	CanAdjustFlag

If you set CanAdjustFlag to true, and if you copy the sales order or return it, then you can adjust this entity's values in the copy or return. Set it to false, then you can't adjust them and you must use the values from the original sales order.

Charge Component Entity on the Fulfillment Line

Map these values for each charge component entity on the fulfillment line.

Request Payload	Response Payload
OrderChargeComponentTL	FulfillLineOrderChargeComponentSDOTR
ChargeCurrencyCode	ChargeCurrencyCode
ChargeCurrencyDurationExtendedAmou	ChargeCurrencyDurationExtendedAmount
ChargeCurrencyExtAmount	ChargeCurrencyExtendedAmount
ChargeCurrencyCode	ChargeCurrencyExtendedAmount currencyCode
ChargeCurrencyUnitPrice	ChargeCurrencyUnitPrice
ChargeCurrencyCode	ChargeCurrencyUnitPrice currencyCode
Explanation	Explanation
ExplanationMessageName	ExplanationMessageName
HeaderCurrencyCode	HeaderCurrencyCode
HeaderCurrencyDurationExtendedAmo	HeaderCurrencyDurationExtendedAmount
HeaderCurrencyExtAmount	HeaderCurrencyExtendedAmount
HeaderCurrencyCode	HeaderCurrencyExtendedAmount currencyCode
HeaderCurrencyUnitPrice	HeaderCurrencyUnitPrice
HeaderCurrencyCode	HeaderCurrencyUnitPrice currencyCode
PercentOfComparisonElement	PercentOfComparisonElement
PriceElementCode	PriceElementCode
PriceElementUsageCode	PriceElementUsageCode
PricingSourceId	PricingSourceId
PricingSourceTypeCode	PricingSourceTypeCode
RollupFlag	RollupFlag If Y then true, else false.

Request Payload	Response Payload
SequenceNumber	SequenceNumber
SourceChargeId	SourceChargeId
SourceParentChargeCompId	SourceParentChargeCompId
SourceChargeComponentId	SourceChargeComponentId

Price Adjustment Entity on the Fulfillment Line

Map these values for each price adjustment entity on the fulfillment line.

Request Payload	Response Payload
PriceAdjustmentTL	FulfillLineOrderChargeSDOTR
SHIPPING	ChargeAppliesTo
headerTL TransactionalCurrencyCode	ChargeCurrencyCode
QP_SHIP_FREIGHT You can also use an appropriate Charge Definition Code instead.	ChargeDefinitionCode • Make sure you have only one charge for each Charge Definition Code. • Make sure you set the Applies To attribute for this charge definition to Shipping. • If the order line is a return line and you need to refund the freight charge, then make sure this Charge Definition Code is refundable. • To verify charge definition codes, go to the Setup and Maintenance work area, then open the Manage Pricing Charge Definitions page. Search for your charge definition, then make sure the combination of Charge Definition Code, Charge Type Code, Charge Sub Type Code and Price Type Code is valid. For details, see *Manage Pricing Charge Definitions*.
ORA_PRICE You can also use an appropriate ChargeCharge Sub Type Code instead.	ChargeSubtypeCode
ORA_SHIPPING_FREIGHT You can also use an appropriate Charge Type Code instead.	ChargeTypeCode
headerTL FulfillLineTLVO1 OrderedQty	PricedQuantity

Request Payload	Response Payload
`headerTL FulfillLineTLVO1 OrderedUom`	`PricedQuantity unitCode`
`headerTL FulfillLineTLVO1 OrderedUom`	`PricedQuantityUOMCode`
`ONE_TIME`	`PriceTypeCode`
true or false	`PrimaryFlag` You can have only one primary charge for each ChargeAppliesTo on a fulfillment line.
false	`RollupFlag`
Maximum plus position where • Maximum equals the FulfillLineTLVO1 divided by OrderChargeTL divided by SequenceNumber • Position equals FulfillLineTLVO1 divided by PriceAdjustmentTL Use this code to calculate the value: `ue-among-nodeset (../ nsmpr0:OrderChargeTL/ nsmpr0:SequenceNumber ) + position()`	`SequenceNumber`
`PriceAdjustmentId`	`SourceChargeId`
true	`CanAdjustFlag`

Note

- You repeat this entity in each FulfillLineOrderChargeSDOTR entity.
- You can use the code that's in the Request Payload column with Oracle Integration Cloud Service. If you use a different service, then you might need to use some other code.

FulfillLineOrderChargeComponentSDOTR Entity on the Fulfillment Line

Map these values for each FulfillLineOrderChargeComponentSDOTR entity on the fulfillment line.

Request Payload	Response Payload
`headerTL TransactionalCurrencyCode`	`ChargeCurrencyCode`

Request Payload	Response Payload

Amount multiplied by markup divided by OrderedQty:
`headerTL`
`FulfillLineTLVO1`
`OrderedQty`
Use this code to calculate the value:
`nsmpr0:Amount div ../`
`nsmpr0:OrderedQty`
	`ChargeCurrencyUnitPrice`

`headerTL` `TransactionalCurrencyCode`	`ChargeCurrencyUnitPrice` `currencyCode`

Amount multiplied by markup divided by OrderedQty:
`headerTL`
`FulfillLineTLVO1`
`OrderedQty`
Use this code to calculate the value:
`nsmpr0:Amount div ../`
`nsmpr0:OrderedQty`
	`HeaderCurrencyUnitPrice`

`headerTL` `TransactionalCurrencyCode`	`HeaderCurrencyUnitPrice` `currencyCode`
Amount multiplied by markup	`ChargeCurrencyExtendedAmount`
`headerTL` `TransactionalCurrencyCode`	`ChargeCurrencyExtendedAmount` `currencyCode`
Amount multiplied by markup	`HeaderCurrencyExtendedAmount`
`headerTL` `TransactionalCurrencyCode`	`HeaderCurrencyExtendedAmount` `currencyCode`
`headerTL` `TransactionalCurrencyCode`	`HeaderCurrencyCode`
`QP_NET_PRICE`	`PriceElementCode`
`PriceAdjustmentId`	`SourceChargeId`
`NET_PRICE`	`PriceElementUsageCode`
`AdjustmentName`	`Explanation`
`false`	`RollupFlag`
`1`	`SequenceNumber`

Request Payload	Response Payload
You are adding one charge component, so you use a literal value of 1.	
`1` You are adding one charge component, so you use a literal value of 1.	`SourceChargeComponentId`

`markup` is conditional. If the value that you need to charge in this attribute equals the shipping cost, then don't include the markup in the expression.

Map these values for each duplicate FulfillLineOrderChargeComponentSDOTR entity on the fulfillment line.

Request Payload	Response Payload
`headerTL` `TransactionalCurrencyCode`	`ChargeCurrencyCode`

Amount multiplied by markup divided by OrderedQty:
`headerTL`
`FulfillLineTLVO1`
`OrderedQty`
Use this code to calculate the value:
`nsmpr0:Amount div ../`
`nsmpr0:OrderedQty`
	`ChargeCurrencyUnitPrice`

`headerTL` `TransactionalCurrencyCode`	`ChargeCurrencyUnitPrice` `currencyCode`

Amount multiplied by markup divided by OrderedQty:
`headerTL`
`FulfillLineTLVO1`
`OrderedQty`
Use this code to calculate the value:
`nsmpr0:Amount div ../`
`nsmpr0:OrderedQty`
	`HeaderCurrencyUnitPrice`

`headerTL` `TransactionalCurrencyCode`	`HeaderCurrencyUnitPrice` `currencyCode`
Amount multiplied by markup	`ChargeCurrencyExtendedAmount`
`headerTL` `TransactionalCurrencyCode`	`ChargeCurrencyExtendedAmount` `currencyCode`
Amount multiplied by markup	`HeaderCurrencyExtendedAmount`

Request Payload	Response Payload
headerTL TransactionalCurrencyCode	HeaderCurrencyExtendedAmount currencyCode
headerTL TransactionalCurrencyCode	HeaderCurrencyCode
QP_LIST_PRICE	PriceElementCode
PriceAdjustmentId	SourceChargeId
LIST_PRICE	PriceElementUsageCode
false	RollupFlag
2 You are adding two charge components, so you use a literal value of 2.	SequenceNumber
2 You are adding two charge components, so you use a literal value of 2.	SourceChargeComponentId

Example Response Payload

Here's an example payload for a delayed response. The CostToChargeDelayedResponse.xml file contains this response. Use the FulfillLineOrderChargeSDOTR tag to send back the shipping charges.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:FulfillmentRequest xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/taskLayer/fulfillOrder/
DooTaskFulfillOrderResponseInterfaceComposite">
 <ns1:FLine xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/common/process/model/">
 <ns2:FulfillLineId>300100135839049</ns2:FulfillLineId>
 <ns2:SourceOrderSystem>GPR</ns2:SourceOrderSystem>
 <ns2:Status>COMPLETED</ns2:Status>
 <ns2:TaskType>ConvertCostToCharge</ns2:TaskType>
 <ns2:FulfillLineOrderChargeSDOTR>
 <ns2:ChargeDefinitionCode>QP_SHIP_FREIGHT</ns2:ChargeDefinitionCode>
 <ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
 <ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
 <ns2:PricedQuantity>3</ns2:PricedQuantity>
 <ns2:PricedQuantityUOMCode>Ea</ns2:PricedQuantityUOMCode>
 <ns2:PrimaryFlag>true</ns2:PrimaryFlag>
 <!-- This attribute specifies that the charge is a shipping charge: -->
 <ns2:ChargeAppliesTo>SHIPPING</ns2:ChargeAppliesTo>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceChargeIdentifier>300100171244129</ns2:SourceChargeIdentifier>
 <ns2:ChargeTypeCode>ORA_SHIPPING_FREIGHT</ns2:ChargeTypeCode>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:SequenceNumber>2</ns2:SequenceNumber>
 <ns2:GsaUnitPrice/>
 <ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
```

```
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>15</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<!-- This attribute specifies the unit net price of the charge in the charge's currency:
It can be same or different than the order header currency. -->
<ns2:ChargeCurrencyUnitPrice>5</ns2:ChargeCurrencyUnitPrice>
<!-- This attribute specifies the per unit net price of the charge in the order header's currency.
It can be the same or different than the charge's currency: -->
<ns2:HeaderCurrencyUnitPrice>5</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<!-- This attribute identifies the charge for this charge component in the source system:-->
<ns2:SourceChargeId>300100171244129</ns2:SourceChargeId>
<!-- This attribute identifies the charge component in the source system: -->
<ns2:SourceChargeComponentId>1</ns2:SourceChargeComponentId>
<!-- This attribute specifies the total line amount for the charge component in the charge's currency.
You must specify the quantity multiplied by the unit price in the charge's currency: -->
<ns2:ChargeCurrencyExtendedAmount>15</ns2:ChargeCurrencyExtendedAmount>
<ns2:Explanation>UPS Freight Costs</ns2:Explanation>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>15</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<!-- This attribute specifies the unit net price of the of the charge in the currency in which charge was
calculated.
It can be same or different than the order header currency. -->
<ns2:ChargeCurrencyUnitPrice>5</ns2:ChargeCurrencyUnitPrice>
<!-- This attribute specifies the per unit net price of the of the charge in the currency of the order
header.
It can be same or different than the charge currency. -->
<ns2:HeaderCurrencyUnitPrice>5</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<!-- This attribute identifies the charge for this charge component in the source system: -->
<ns2:SourceChargeId>300100171244129</ns2:SourceChargeId>
<!-- This attribute identifies the charge component in the source system: -->
<ns2:SourceChargeComponentId>2</ns2:SourceChargeComponentId>
<!-- This attribute specifies the total line amount for the charge component in the charge's currency.
You must specify the quantity multiplied by the unit price in the charge's currency:-->
<ns2:ChargeCurrencyExtendedAmount>15</ns2:ChargeCurrencyExtendedAmount>
</ns2:FulfillLineOrderChargeComponentSDOTR>
</ns2:FulfillLineOrderChargeSDOTR>
<!-- Here are the existing charges. Don't modify these values:>
<ns2:FulfillLineOrderChargeSDOTR>
<ns2:ChargeAppliesTo>PRICE</ns2:ChargeAppliesTo>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ParentEntityCode>LINE</ns2:ParentEntityCode>
<ns2:PricePeriodicityCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>3</ns2:PricedQuantity>
<ns2:PricedQuantityUomCode>Ea</ns2:PricedQuantityUomCode>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ReferenceOrderChargeId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:AvgUnitSellingPrice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:FreightReferenceLineId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:CanAdjustFlag>true</ns2:CanAdjustFlag>
```

```
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>7500</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>2500</ns2:ChargeCurrencyUnitPrice>
<ns2:Explanation>Base List Price Applied from Corporate Segment Price List</ns2:Explanation>
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>7500</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>2500</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>1</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_BASE_LIST_PRICE</ns2:PriceElementCode>
<ns2:PriceElementUsageCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceId>300100071623860</ns2:PricingSourceId>
<ns2:PricingSourceTypeCode>PRICE_LIST_CHARGE</ns2:PricingSourceTypeCode>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1000</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226042</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>7500</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>2500</ns2:ChargeCurrencyUnitPrice>
<ns2:Explanation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>7500</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>2500</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>1</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:PricingSourceId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceTypeCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1001</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226043</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>-30</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>-10</ns2:ChargeCurrencyUnitPrice>
<ns2:Explanation>Manual Discount of 10USD for reason code Price match</ns2:Explanation>
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>-30</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>-10</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>-0.004</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_CUSTOM_ADJUSTMENT</ns2:PriceElementCode>
<ns2:PriceElementUsageCode>PRICE_ADJUSTMENT</ns2:PriceElementUsageCode>
<ns2:PricingSourceId>300100171226062</ns2:PricingSourceId>
<ns2:PricingSourceTypeCode>MANUAL_ADJUSTMENT</ns2:PricingSourceTypeCode>
<ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1002</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226064</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>-33</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>-11</ns2:ChargeCurrencyUnitPrice>
```

```
 <ns2:Explanation>Manual Discount of 11USD for reason code Error correction</ns2:Explanation>
 <ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>-33</ns2:HeaderCurrencyExtendedAmount>
 <ns2:HeaderCurrencyUnitPrice>-11</ns2:HeaderCurrencyUnitPrice>
 <ns2:ModifiedFlag>N</ns2:ModifiedFlag>
 <ns2:PercentOfComparisonElement>-0.0044</ns2:PercentOfComparisonElement>
 <ns2:PriceElementCode>QP_CUSTOM_ADJUSTMENT</ns2:PriceElementCode>
 <ns2:PriceElementUsageCode>PRICE_ADJUSTMENT</ns2:PriceElementUsageCode>
 <ns2:PricingSourceId>300100171226080</ns2:PricingSourceId>
 <ns2:PricingSourceTypeCode>MANUAL_ADJUSTMENT</ns2:PricingSourceTypeCode>
 <ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SequenceNumber>1003</ns2:SequenceNumber>
 <ns2:SourceChargeComponentId>300100171226082</ns2:SourceChargeComponentId>
 <ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
 <ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 </ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:ChargeCurrencyExtendedAmount>-7.5</ns2:ChargeCurrencyExtendedAmount>
 <ns2:ChargeCurrencyUnitPrice>-2.5</ns2:ChargeCurrencyUnitPrice>
 <ns2:Explanation>Manual Discount of 0.1% on price element List Price for reason code Sales negotiation</
ns2:Explanation>
 <ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>-7.5</ns2:HeaderCurrencyExtendedAmount>
 <ns2:HeaderCurrencyUnitPrice>-2.5</ns2:HeaderCurrencyUnitPrice>
 <ns2:PercentOfComparisonElement>-0.001</ns2:PercentOfComparisonElement>
 <ns2:PriceElementCode>QP_CUSTOM_ADJUSTMENT</ns2:PriceElementCode>
 <ns2:PriceElementUsageCode>PRICE_ADJUSTMENT</ns2:PriceElementUsageCode>
 <ns2:PricingSourceId>300100171226101</ns2:PricingSourceId>
 <ns2:PricingSourceTypeCode>MANUAL_ADJUSTMENT</ns2:PricingSourceTypeCode>
 <ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SequenceNumber>1004</ns2:SequenceNumber>
 <ns2:SourceChargeComponentId>300100171226103</ns2:SourceChargeComponentId>
 <ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
 <ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 </ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:ChargeCurrencyExtendedAmount>7429.5</ns2:ChargeCurrencyExtendedAmount>
 <ns2:ChargeCurrencyUnitPrice>2476.5</ns2:ChargeCurrencyUnitPrice>
 <ns2:Explanation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>7429.5</ns2:HeaderCurrencyExtendedAmount>
 <ns2:HeaderCurrencyUnitPrice>2476.5</ns2:HeaderCurrencyUnitPrice>
 <ns2:PercentOfComparisonElement>0.9906</ns2:PercentOfComparisonElement>
 <ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
 <ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
 <ns2:PricingSourceId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:PricingSourceTypeCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SequenceNumber>1005</ns2:SequenceNumber>
 <ns2:SourceChargeComponentId>300100171226044</ns2:SourceChargeComponentId>
 <ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
 <ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
 </ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:FulfillLineOrderChargeComponentSDOTR>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:ChargeCurrencyExtendedAmount>1485.9</ns2:ChargeCurrencyExtendedAmount>
 <ns2:ChargeCurrencyUnitPrice>495.3</ns2:ChargeCurrencyUnitPrice>
 <ns2:Explanation>Exclusive Tax ( 20%)</ns2:Explanation>
```

```
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>1485.9</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>495.3</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>0.19812</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_EXCLUSIVE_TAX</ns2:PriceElementCode>
<ns2:PriceElementUsageCode>EXCLUSIVE_TAX</ns2:PriceElementUsageCode>
<ns2:PricingSourceId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceTypeCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1015</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226200</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>8915.4</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>2971.8</ns2:ChargeCurrencyUnitPrice>
<ns2:Explanation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>8915.4</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>2971.8</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>1.18872</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_NET_PRICE_PLUS_TAX</ns2:PriceElementCode>
<ns2:PriceElementUsageCode>NET_PRICE_PLUS_TAX</ns2:PriceElementUsageCode>
<ns2:PricingSourceId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceTypeCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1016</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226201</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:FulfillLineOrderChargeComponentSDOTR>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeCurrencyExtendedAmount>7429.5</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeCurrencyUnitPrice>2476.5</ns2:ChargeCurrencyUnitPrice>
<ns2:Explanation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ExplanationMessageName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>7429.5</ns2:HeaderCurrencyExtendedAmount>
<ns2:HeaderCurrencyUnitPrice>2476.5</ns2:HeaderCurrencyUnitPrice>
<ns2:PercentOfComparisonElement>0.9906</ns2:PercentOfComparisonElement>
<ns2:PriceElementCode>QP_MARGIN</ns2:PriceElementCode>
<ns2:PriceElementUsageCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:PricingSourceTypeCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:ReferenceOrderChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SequenceNumber>1017</ns2:SequenceNumber>
<ns2:SourceChargeComponentId>300100171226045</ns2:SourceChargeComponentId>
<ns2:SourceChargeId>300100171226040</ns2:SourceChargeId>
<ns2:SourceParentChargeCompId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</ns2:FulfillLineOrderChargeComponentSDOTR>
</ns2:FulfillLineOrderChargeSDOTR>
</ns1:FLine>
</ns1:FulfillmentRequest>
</soap:Body>
</soap:Envelope>
```

Update Attributes on Split Order Lines for Partial Shipments

Update attributes on a split order line for a partial shipment.

Realize these benefits:

- Update attributes on an order line after you ship part of that line.
- Avoid having to cancel split lines and manually recreate new ones when all you want to do is update attributes on the split line.
- Reduce the number of steps and the time you need to process change on an order line after you ship part of that line.
- Improve efficiency and accuracy of change processing on the line after you ship part of the line.
- Improve tracking and reporting for lines that you fulfill in more than one shipment.
- Increase customer satisfaction.

Consider a scenario.

- You partially ship an order line because some items in the warehouse were damaged.
- You update the Requested Date attribute or the warehouse attribute on the split line and submit your change.
- Oracle Order Management sends your change to Oracle Global Order Promising, and Promising reschedules the line it so it ships on a different date or from a different warehouse.
- Order Management also sends an update to Oracle Inventory Management, and Inventory Management updates the reservation and shipment details.
- Oracle Shipping ships the split line according to the updated schedule date or from the updated warehouse.

Shipping can ship part of a fulfillment line to help meet customer demand. If only some of the quantity on an order line is currently available on the requested date, then Shipping ships the quantity that it can ship, and Order Management creates a split order line for the remaining quantity. We now have a new fulfillment line in Order Management that has the remaining quantity that Shipping still needs to ship.

You might need to update attributes on the split line in Order Management after Shipping ships part of the order line. For example, you know the Denver warehouse has enough supply to fulfill the line that hasn't shipped, so you update the warehouse attribute on that line to Denver, submit your change, and Order Management will send these details to your downstream fulfillment system so it can fulfill the request.

Use this feature so you can:

- Update attributes on a split order line that Order Management creates when Shipping ships part of an order line in a standard or a back-to-back flow.
- Update attributes on a split line in the Order Management work area, through REST API, or in some other way, such as through Backlog Management in Oracle Supply Planning, Global Order Promising, or an update to a work order, purchase order, or transfer order in a back-to-back flow.
- Orchestrate and send change on the split line to your downstream fulfillment system. For example, to Global Order Promising to reschedule the line, to Inventory Management to update reservation and shipment details, to Global Trade Management to screen for trade compliance, to Transportation Management to plan transportation, and so on.
- Examine reservation and shipment details for the split lines in Inventory Management after you update an attribute on the line in Order Management.

- Send the fulfillment line for the split line in a shipping service to some other application, such as Transportation Management, Global Trade Management, and so on.

Update the value of an attribute on a split line in Order Management, and then use that value to:

- Reprice and recalculate tax.
- Reauthorize the credit card transaction when your customer uses a credit card to pay for the item, or when you update the ordered quantity, price, credit card, and so on.
- Fulfill, recognize revenue, and bill the split lines.
- Cost the transaction and recognize the cost of goods sold for the split lines in Oracle Cost Management.
- Orchestrate financial details and process the transaction in Oracle Supply Chain Financial Orchestration for the split lines.

For more, including helpful screen prints, see *Demonstration of the Update Attributes on Split Order Lines for Partial Shipments Feature*.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Revise a Sales Order

You can update one or more attributes on a split line. You can update line attributes just like you do when you update attributes on a line that isn't split and that's awaiting shipping.

You can update an attribute on the order header, and Order Management will cascade your update to all the split lines that you haven't overridden. You can also use the Override Order Line action to update an attribute on the split line.

As an option, you can use the Update Lines action to update attributes on all the lines that you select at the same time. If you opt into this feature, then you can use this action for the split lines and for the order lines that you backorder in Shipping.

You can also use an order management extension or posttransformation rule to update an attribute on a split line.

Here's an example of what you can do on the Create Order Revision page.

- Make changes to more than one line. For details, see *Update More Than One Order Line*.
- Revise the requested date. Order Management will cascade the date to each split order line.
- Update the quantity.
- Manually adjust the price.
- Use an override action to update the warehouse.

You can do an action just like you do on a line that isn't split and that's awaiting shipping. For example:

- Edit Line Details
- Manage Attachments
- Edit Additional Information
- Edit Tax Determinants
- Manage Incentives
- Cancel Line

- Override Order Line
- Manage Sales Credit

What You Can't Do

You can’t update an attribute on a split line if:

- It’s a line that Order Management created while partially fulfilling a drop shipment.
- It’s a remnant line of a configured item. For more about remnants, see *Fulfill Configured Items*.
- It’s a line that Order Management created when Oracle Receiving partially received a return line.
- It’s on a sales order that you submitted before you opted into the Update Attributes on Split Order Lines for Partial Shipments feature. You can update attributes on a split line only on a sales order that you submit after you opt into the feature.

Note

- You can’t update the Unit of Measure, Shipment Set, Supplier and Supplier Site attributes on a split line.
- You can’t update the configuration of a configured item on a split line.
- If you update an attribute on a split line through REST API, or through some way, such as through Backlog Management, Global Order Promising, or Supply Planning, but it’s a split line that Order Management doesn't allow you to update, then you will receive an error. However, if it’s an update from a supply order in a back-to-back flow, then Order Management will ignore the update. It will update the work order, purchase order, or transfer order and the supply order for the split line, but Order Management won't update the split line.

Use Fulfillment Views

You can use actions in the fulfillment views to update split lines just like you do when you use actions on a line that isn't split and that's awaiting shipping.

If you use an unschedule or an unreserve action for one or more split lines, and if the fulfillment tolerance on at least one of the lines that you select isn’t 0, then Order Management will ask you to confirm the action.

If you use an action in a fulfillment view to update the split lines, and if your selection includes:

- Only the split lines that Order Management can’t update, then Order Management will display the same error it displays when you don't use this feature.
- Split lines that Order Management can update, then Order Management updates them, and doesn't update the split lines that it can't update.

How Order Management Processes Change on Split Lines

Order Management processes change on a split line similar to how it processes change on a line that isn't split and that's awaiting shipping. You can use different compensation patterns according to the attributes that you update or according to the fulfillment task that Order Management is compensating.

If you update an attribute on a split line, and if the orchestration process uses that attribute to identify change for a fulfillment task, then the orchestration process sends the updated attribute value on the split line to that fulfillment system. For example, if you update attributes such as the ordered quantity, shipping method, requested date, scheduled ship date and so on a split line, then Order Management compensates the shipping task and sends the updated attribute value to Shipping. If this is the first update on the split line, then:

- The orchestration process compensates all of its fulfillment tasks, then sends the split line to the fulfillment system. This helps to make sure that the split line is available in the fulfillment system after the first update.

- If your orchestration process uses a trade compliance task, transportation planning task, template task, or fulfillment task to integrate with an application that isn't an Oracle application, then the orchestration process also sends the parent fulfillment line of the split line to that application, such as Oracle Global Trade Management, Oracle Transportation Management, and so on. That application then updates the quantity of the parent fulfillment line so it's the same as the ordered quantity in Order Management. This logic helps to make sure the fulfillment application has the same ordered quantity that Order Management has for the parent fulfillment line of the split order line.

If you substitute an item or update the warehouse on a split line, then Order Management will send a request to cancel the current request to the fulfillment system, creates a new request for the updated line, then send the new request to the fulfillment system. For example, if your orchestration process has schedule, supply, and shipping tasks, then:

- Global Order Promising will cancel the old demand and create a new one.
- Supply Chain Orchestration will cancel the old supply order and create a new one.
- Inventory Management will cancel the old reservation and shipment line and create a new reservation and shipment line.

Note

- Order Management applies this same logic when you unschedule or unreserve a split line, and then schedule or reserve the split line.
- If you enable the Use Flexfield Attributes option on your orchestration process, and if you update an extensible flexfield on a split line, then the orchestration process compensates it's fulfillment tasks. If you use a service mapping to send the extensible flexfield to Shipping, then Order Management sends the updated values in the extensible flexfield to Shipping.
- You can cancel a split line or fulfill it without updating any attribute just like you do when you don't use this feature. However, if you cancel now, then your cancellation request cancels the current request in all the fulfillment systems that your orchestration process references.
- Order Management doesn't compensate the parent line of a split line when you cancel that split line for a partial shipment.

For details about compensation, see *Overview of Managing Change That Occurs During Order Fulfillment*

Guidelines

You must enable the feature:

1. Go to the Setup and Maintenance work area.
2. Select the Order Management offering, then click **Change Feature Opt In**.
3. On the Opt In page, click the **pencil** in the row that has Order Management in the Name column.
4. On the Edit Features page, add a check mark to the Update Attributes on Split Order Lines for Partial Shipments feature.

More

- You can't opt out of the Update Attributes on Split Order Lines for Partial Shipments feature after you opt in and submit a sales order. At this point, you're committed to using the feature.
- Use this feature with sales orders that you revise in the Order Management work area, that you import through REST API, through file-based data import (FBDI) that uses REST API, or through an update from some other way, such as through Backlog Management, Global Order Promising, or Supply Planning.
- Use this feature in a standard or a back-to-back flow.

- You can't use a web service from Application Development Framework (ADF) or a file-based data import (FBDI) that uses an ADF web service to update an attribute on a split line. You can only use an ADF web service to cancel a split line like just you do when you don't use this feature.
- You can update an attribute on a split line that has a configured item or a kit just like you do when you update an attribute on a line that isn't split and that's awaiting shipping, but you can't modify the configuration of a configured item on a split line.
- If you update an attribute on a split line, then Oracle Pricing reprices the line just like it does when you update an attribute on a line that isn't split and that's awaiting shipping. A tier adjustment, manual price adjustment, sales agreement, or incentive might affect pricing. The tier adjustment on the original line might not apply on the split line. Make sure you have the correct set up in Pricing Administration, and make sure you set up incentives correctly in Channel Revenue Management.
- If you ship lines in a shipment set, and if you partially ship or don't ship at least one of these lines, then Order Management removes the lines that you haven't shipped from the set. You can now update attributes on the removed lines just like you do on a line that isn't in a shipment set in a standard or a back-to-back flow.
- You can use the Order Management Created Split Line for Partial Shipment validation rule set to constrain some updates on a split line that Order Management creates when Shipping ships part of the order line. For example, you opt into this feature so you can update attributes on split lines, but you don't want to allow your users to update the FOB (free on board) attribute on a split line. You can set up a processing constraint that doesn't allow the user to update the FOB attribute on a fulfillment line, and then use this validation rule set in the condition. You can also use these validation rule sets to constrain updates on a split line or on a line that isn't split according to one of these line statuses in Inventory Management:
  - Fulfillment Line is Backordered
  - Fulfillment Line is Released to Warehouse
  - Fulfillment Line is Picked or Staged

Drop Shipments and Transfers

- If you need to drop ship a split line in a standard or in a back-to-back flow, then you must unschedule the split line, and then schedule it. As an option you can set the supplier and the supplier site before you schedule the split line. You can set the supplier or the supplier site on a split line only after you unschedule it.
- If a line is part of an internal material transfer in Order Management, and if Shipping already shipped part of that line, then you can't change the transfer order in Inventory Management. You can only cancel it. However, you can update a fulfillment attribute on the split line, such as the schedule date, shipping method, and so on. You can update it in a fulfillment view in the Order Management work area in some other way, such as through Backlog Management, Global Order Promising, and so on.

Back-to-Back

If you enable the Defer Online Processing of Inventory Updates shipping parameter for a warehouse, and if you partially ship an order line from this warehouse in a back-to-back flow, then Supply Chain Orchestration will close the supply order after Shipping ships the first partial shipment. So, you can't update an attribute on the split line at this point. If you must update an attribute on the split line, then:

1. Unschedule the split line.
2. Update your attribute on the split line, such as the Scheduled Ship Date attribute or the Order Quantity attribute.
3. Schedule the split line.

   Supply Chain Orchestration will create a new supply order for the split line with your updated attribute values. You can substitute an item or update a warehouse attribute on the split line without unscheduling it.

If you modify the warehouse on a split line or on a line that isn't split in a back-to-back flow, then Supply Chain Orchestration will cancel the previous supply and create the optimal supply for the new warehouse. Global Order Promising uses your sourcing rules to determine the optimal new supply. This behavior is same on the split line or on the line that isn't split.

For more details about how you can process split order lines in a back-to-back flow, see the Improve Supply Tracking for Sales Orders in Your Back to Back Flows feature in the Supply Chain Orchestration functional area.

Fulfillment Tolerances and Substitutions

- If you substitute the item on a split line, or unschedule or unreserve it, and if the fulfillment tolerance on the line isn't 0, then Shipping will apply the fulfillment tolerance on the split order line, but it won't consider the cumulative quantity that you already shipped across all of the order line's fulfillment lines. Instead, Shipping will consider only the quantity on the fulfillment line that Shipping is shipping.
- If you need to substitute the item on a split line or split the split line again according to a recommendation from Global Order Promising, then you must unschedule the split line, and then schedule it.

Warehouse Updates

- You can update the selling profit center attribute on a split line only if you also update the warehouse attribute on that line.
- If the new warehouse on a split line and the old warehouse on the original order line each use a different primary unit of measure for the item, and if the fulfillment tolerance on the split line isn't 0, then you can't update the warehouse on the split line. Order Management will reject this change even if Global Order Promising recommends it. In this situation, you must use the old warehouse on the split line, or your new warehouse must use the primary unit of measure for the item that the original warehouse uses.

Inventory Management

Inventory Management updates the reservation for a split line according to the change request that it receives from Order Management.

If you update an attribute on a split line, then the reservation displays a reference to the fulfillment line for the split line. The attribute that you update must be an attribute that identifies change for the reservation task in Order Management.

Here are some examples of how Inventory Management handles the update that you make.

What You Do on the Split Line in Order Management	What Inventory Management Does
Reduce the ordered quantity.	Reduces the reserved quantity on the reservation to the ordered quantity.
Increase the ordered quantity.	Increases the reserved quantity on the reservation to the ordered quantity. If Inventory Management already staged a shipment line for the split line, then it creates a new reservation for the quantity that exceeds the original quantity.
Update the ordered quantity with no existing reservation.	If there is no reservation on the split line, Inventory Management creates a new reservation. This might happen when Shipping backorders the remaining quantity of a fulfillment line that it partially ships.
Update the warehouse.	Deletes the existing reservation for the split line in the original warehouse, and then creates a new reservation in the new warehouse.
Substitute the item.	Deletes the reservation for the original item, and then creates a new reservation for the new item.

What You Do on the Split Line in Order Management	What Inventory Management Does
Do an unschedule action or an unreserve action.	Deletes the existing reservation for the split line. If you then do a schedule or reserve action, then Inventory Management creates a new reservation. If you also update the scheduled ship date when you do this action, then Inventory Management updates the Due Date attribute on the Manage Reservations and Picks page.

Shipping

Oracle Shipping updates the shipment line for a split order line according to the change request that it receives from Order Management.

Here are some examples of how Shipping handles updates that you make on an attribute that identifies change for the shipping task in Order Management.

What You Do on the Split Line in Order Management	What Shipping Does
Update an attribute.	If the attribute that you update identifies change for the shipping task, then Shipping also updates the shipment line. For example, if you update the requested date in Order Management, then Shipping updates the requested date on the shipment line. The same logic applies for other attributes that identify change, such as the scheduled ship date, shipping method, FOB, and so on.
Increase the ordered quantity.	Creates a new shipment line for the quantity that exceeds the original quantity, and sets the line's status to Ready to Release.
Reduce the ordered quantity.	Reduces the quantity on the shipment line that's in the Staged, Released to Warehouse, or Backordered status. Shipping doesn't reduce the picked quantity for shipment lines that are in the Released to Warehouse status.
Update the warehouse, substitute an item, or do an unschedule action or an unreserve action.	Cancels the existing shipment line for the split line, creates a new shipment line, and sets the shipment line's status to Ready to Release. If you: • Update the warehouse, then Shipping creates a new shipment line with the new warehouse. • Substitute an item, then Shipping creates a new shipment line with the new item. • Unschedule and then schedule the split line, or unreserve and then reserve the split line, then Shipping creates a new shipment line with a new scheduled ship date.

If you update an attribute on a split line, then Order Management populates the Original Source Order Fulfillment Line attribute on the shipment lines of the split order line. It populates the attribute with a reference to the fulfillment line of the original order line. The attribute that you update must be an attribute that identifies change for any of the tasks in the orchestration process.

Shipping doesn't populate the Original Source Order Fulfillment Line attribute on the new shipment line that it creates. Instead, Source Order Fulfillment Line will contain a reference to fulfillment line that fulfils the split order line.

Cost Management

Cost Management uses the changes that it receives from Order Management and Shipping to cost the transactions.

If you update the warehouse on a split line in Order Management, and if that warehouse is in a different business unit than the original warehouse, then Cost Management changes the transaction flow to an Internal Drop Shipment.

Supply Chain Financial Orchestration

Supply Chain Financial Orchestration uses the attribute updates from the split line that Order Management creates to process the intercompany transaction.

- Assume shipment 2548623 is for the original order line that Shipping partially shipped.
- The shipping organization and the selling organization are in the same business unit, so Financial Orchestration doesn't process this shipment.
- Shipping creates shipment 2633638 for the split line.
- You update the shipping warehouse on the split line in Order Management.
- Financial Orchestration will process shipment 2633638 because the shipping organization and the selling organization for 2633638 are in different business units.

The Monitor Financial Orchestration Execution page displays the quantity shipped for a split line in the ordered quantity attribute.

Supply Chain Orchestration

You can track supply for a split line in a back-to-back flow and continue to modify supply until Shipping ships the entire quantity for the line.

- If you ship part of a split line in a back-to-back flow, then Supply Chain Orchestration makes sure that the reservation remains the same for the quantity that hasn't shipped.
- If you modify the quantity or the scheduled ship date on a split line, then Supply Chain Orchestration updates your incoming reserved supply and the reservation for that supply.
- If you modify the warehouse or substitute the item on a split line, then Supply Chain Orchestration will cancel the previous supply and create the optimal supply for the new warehouse or the new item. Global Order Promising uses your sourcing rule to determine the optimal new supply.
- If the supply for your remaining, unshipped quantity gets disrupted, then Supply Chain Orchestration updates the scheduled ship date on the split line, and then searches for another source of supply.
- If the original supply gets canceled, then Supply Chain Orchestration creates more supply. If Supply Chain Orchestration can't find more supply, then it sends a notification to Order Management that the Order Entry Specialist must manually take action, such as modifying an expected ship date.
- If you backorder your unshipped quantity, then Supply Chain Orchestration will make sure that the previous supply is no longer reserved for your backordered line. If you reschedule your backordered line, Supply Chain Orchestration will create new supply.

More

- *Implementing Manufacturing and Supply Chain Materials Management*
- *Set Up Cost Accounting*
- *Using Supply Chain Cost Management*
- *Using Supply Chain Orchestration*

- *Using Functional Setup Manager*

Global Trade Management

Integrate Order Management with Global Trade Management

Create an integration between Oracle Order Management and Oracle Global Trade Management.

You will create an ERP Cloud connection to access a Global Trade Management event. Use the connection as the source for the integration, then use Integration Cloud Service to set up an event that calls the Global Trade Management service.

This topic provides a summary of the set up. This integration is similar to the integration that you create with transportation management or trade compliance. For details, see *Overview of Integrating Order Management with Transportation Management* and *Overview of Setting Up Trade Compliance*.

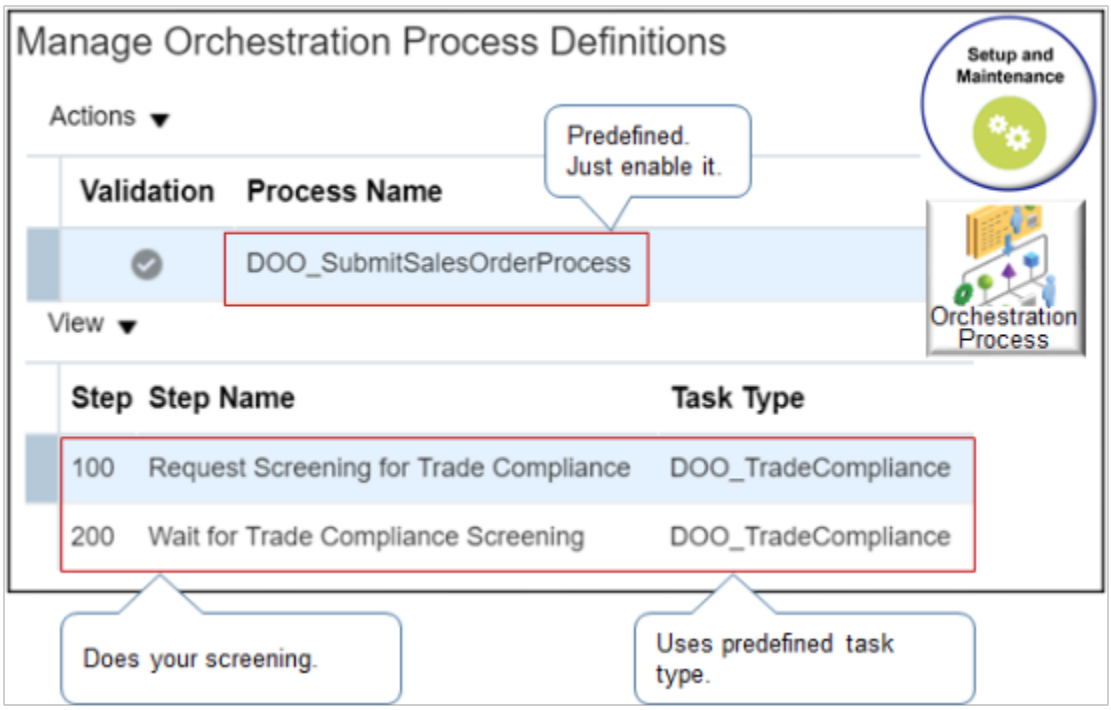
Summary of the Setup

1. Set up Order Management.
2. Create connection for Order Management.
3. Create connection for Global Trade Management.
4. Map source to target.
5. Communicate attributes from Order Management to Global Trade Management.
6. Communicate attributes from Global Trade Management to Order Management.
7. Add business identifiers and activate integration.

Set up Order Management

1. Set the Check for Trade Compliance When User Submits Sales Order parameter to Yes. For details, see *Manage Order Management Parameters*.

2. Set up trade compliance screening so it happens when the user submits a sales order in Order Management, or include the DOO_TradeCompliance task in an orchestration process.
    - Use a predefined orchestration process or create your own.
    - Use the predefined DOO_SubmitSalesOrderProcess orchestration process to validate the sales order during the submit. It already references the predefined DOO_TradeCompliance task type, which does screening for you. For details about orchestration processes, see *Set Up Orchestration Processes*.



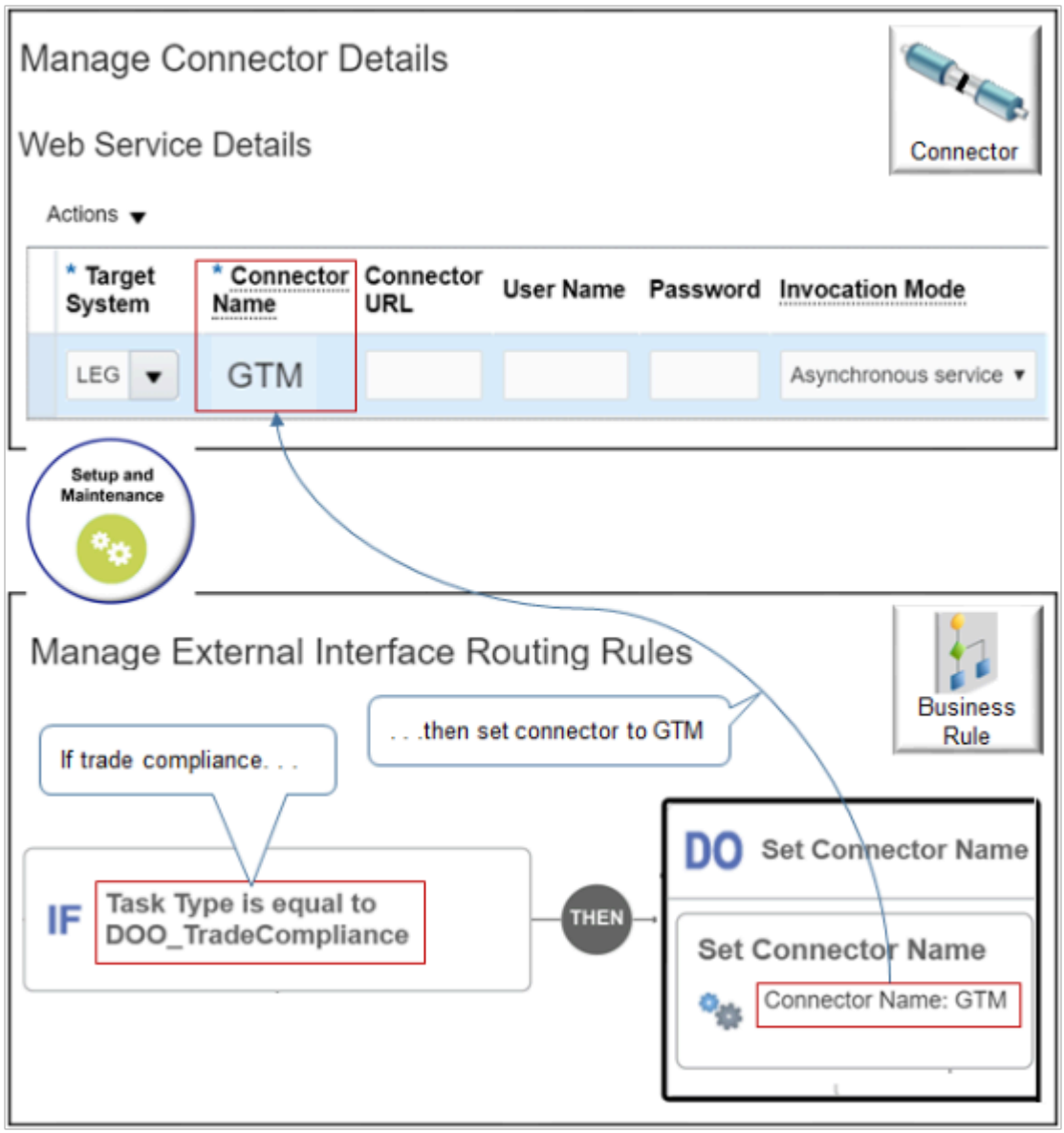
    - If you don't use DOO_SubmitSalesOrderProcess, then add steps that call the DOO_TradeCompliance task type to some other predefined orchestration process or one that you create.

Make sure you include these steps.

Step	Task Type	Service
100	DOO_TradeCompliance	Request Screening for Trade Compliance
200	DOO_TradeCompliance	Wait for Trade Compliance Screening

It isn't necessary to use steps 100 and 200. You can place these steps anywhere in the sequence of steps. But you must request screening first, and then wait.

3. Route the request.



Note

- Use the Manage Connector Details page to add a connector. If you.
  - **Add a routing rule**. Set Invocation Mode to Asynchronous or Synchronous.
  - **Don't add a routing rule**. Set Invocation Mode to Event.

  Learn how to add a connector. For details, see *Manage Connector Details Between Order Management and Your Fulfillment System*.
- Create a routing rule that routes sales orders that you send to Global Trade Management according to the DOO_TradeCompliance task type.
- Use Visual Information Builder to create a routing rule.

```
If Task Type is equal to DOO_TradeCompliance, then Set Connector Name to GTM.
```

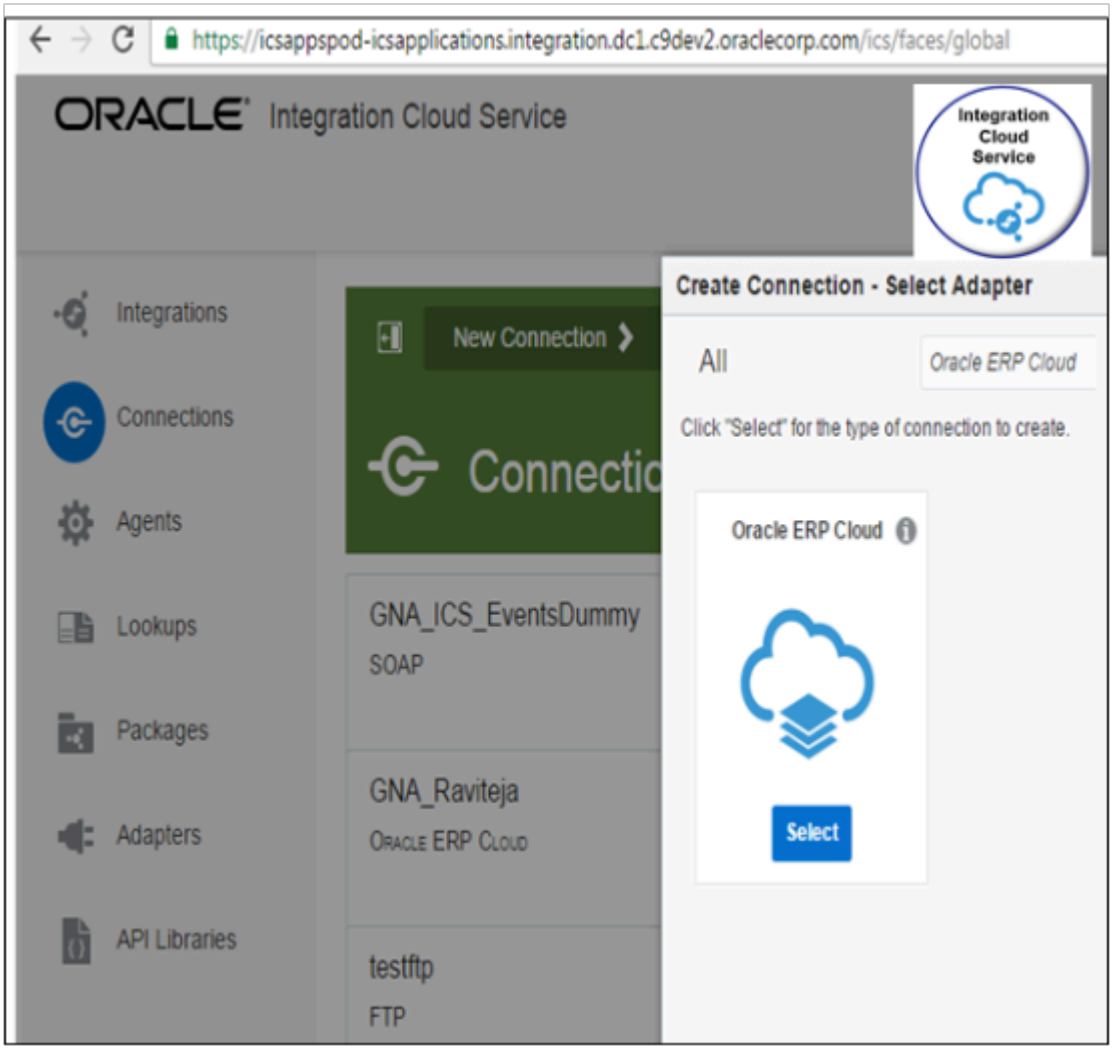
Here's your logic if you use Oracle Business Rules instead of Visual Information Builder.

```
If header is a Order header
 header.Task Type equals ignore case "DOO_TradeCompliance"
Then
 assert new Result (resultObj:"GTM",resultObjKey:"SERVICE_NAME")
```

For details, see *Overview of Using Business Rules With Order Management*.

Create Connection for Order Management

1. Sign into Integration Cloud Service.
2. Click **Connections**, search for Oracle ERP Cloud, then click **Select**.



3. Enter a name. For this example, enter `OMtoGTM`.

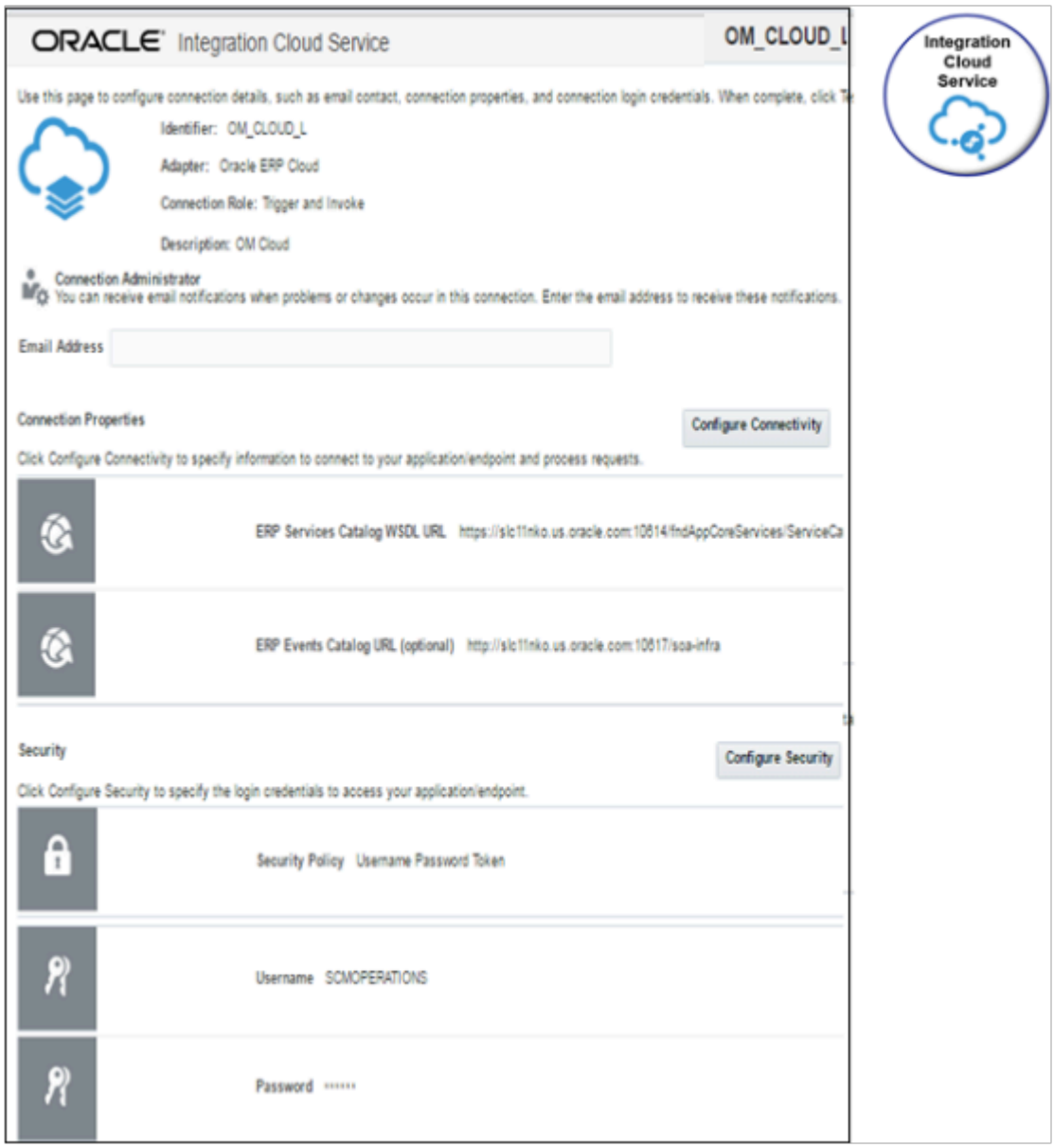
4. Enter the WSDL for ERP Service Catalog. For example:

   ```
   <secure_protocol>://<host>:<secure_port>/fndAppCoreServices/ServiceCatalogService?WSDL
   ```

5. Enter the URL for the ERP Event Catalog. For example:

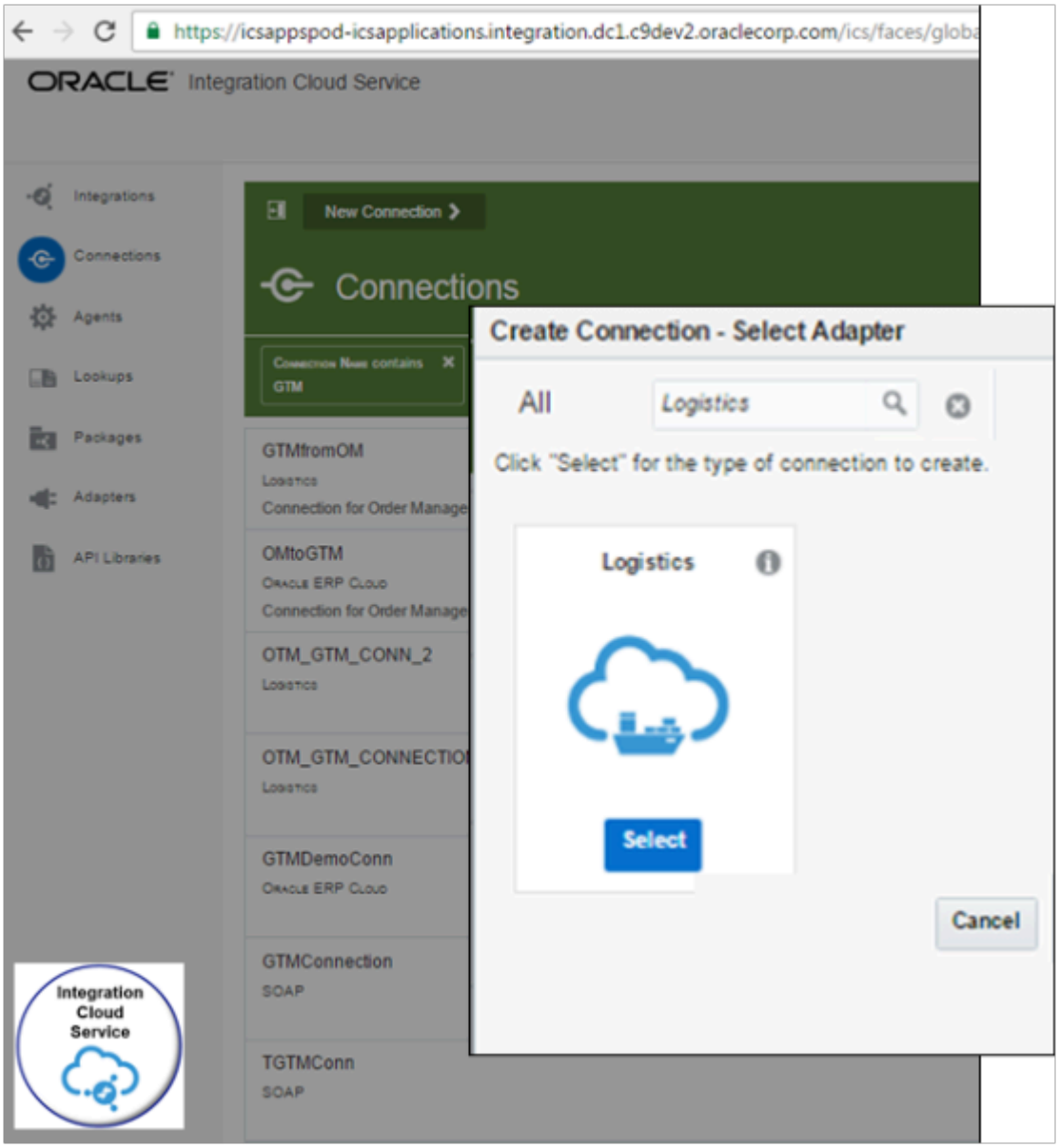
   ```
   <Protocol>://<host>:<port>/soa-infra
   ```

6. Enter the user name and password that you use to sign into Order Management with administrative privileges.
7. Test and save the connection.



Create Connection for Global Trade Management

1. Click **Connections**.
2. Search for Logistics, then click **Select**.



3. Enter a name. For this example, enter `GTMfromOM`.
4. Enter the URL to the WSDL for the GTM service. For example:

   ```
   http://<server:port>/GC3Services/TransmissionService/call?wsdl
   ```

5. Enter the user name and password that you use to access the WSDL.

   The user that you enter must have this privilege.

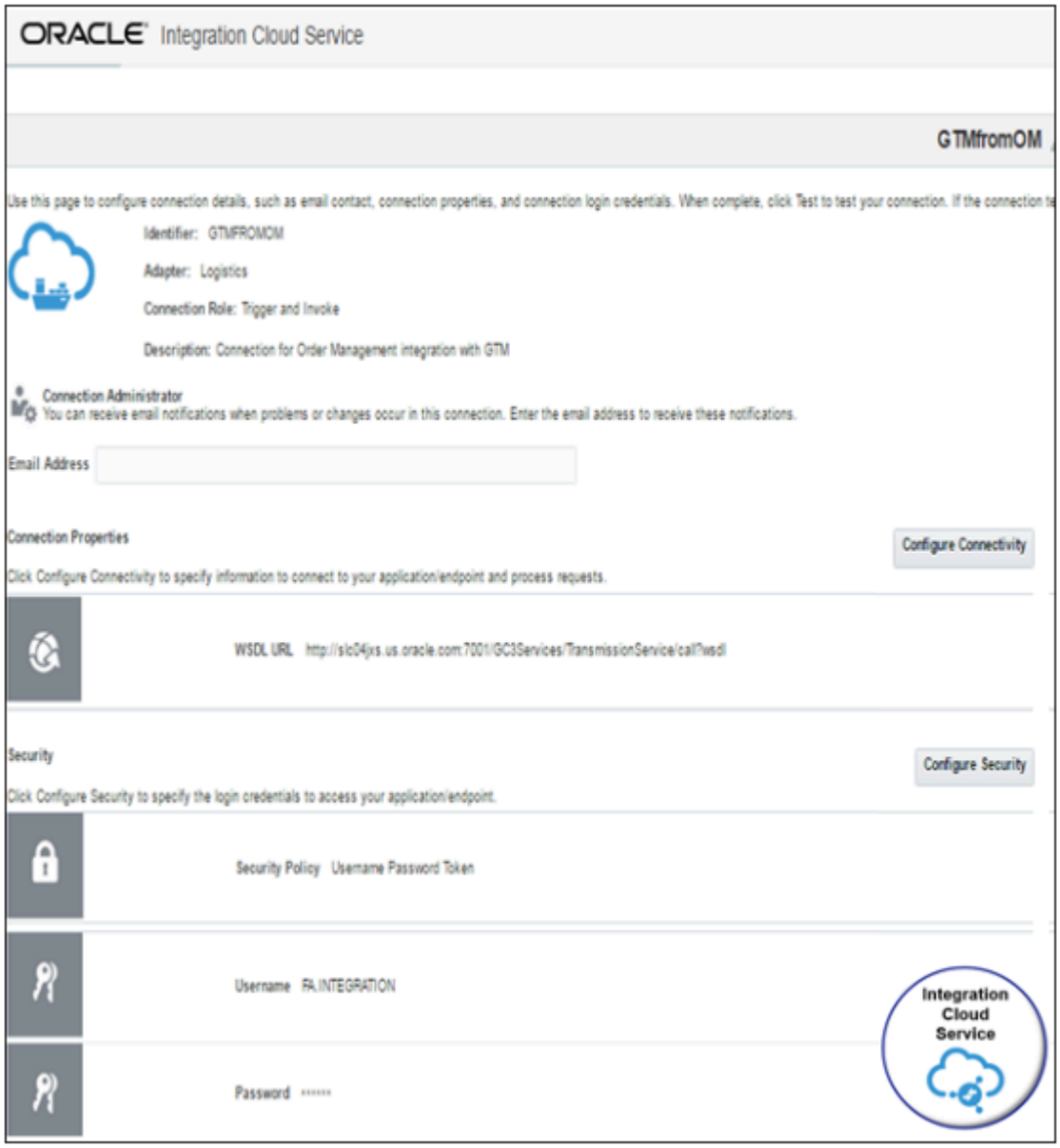
   - Manage Orchestration Order Trade Compliance Interface Web Service

   If it doesn't, you might encounter an error, during setup or in your runtime environment. The error will be something that starts with.

   ```
   The caller is not authorized to call this service
   ```

For details, see *Privileges That You Need to Implement Order Management*.

6. Test and save the connection. You should see this page.



Map Source to Target

1. On the right corner, search for the OMtoGTM you just created, then drag and drop it onto the source. The source is on the left side of the integration.

2. In the dialog that displays, enter details.
    - For the Name, identify the end point, such as OM_Source.
    - On the Request tab, set the values.

Attribute	Value
Type	BusinessEvent
Name	Sales Order Trade Compliance Screening

    - on the Successful Response tab of the Response tab, set the values.

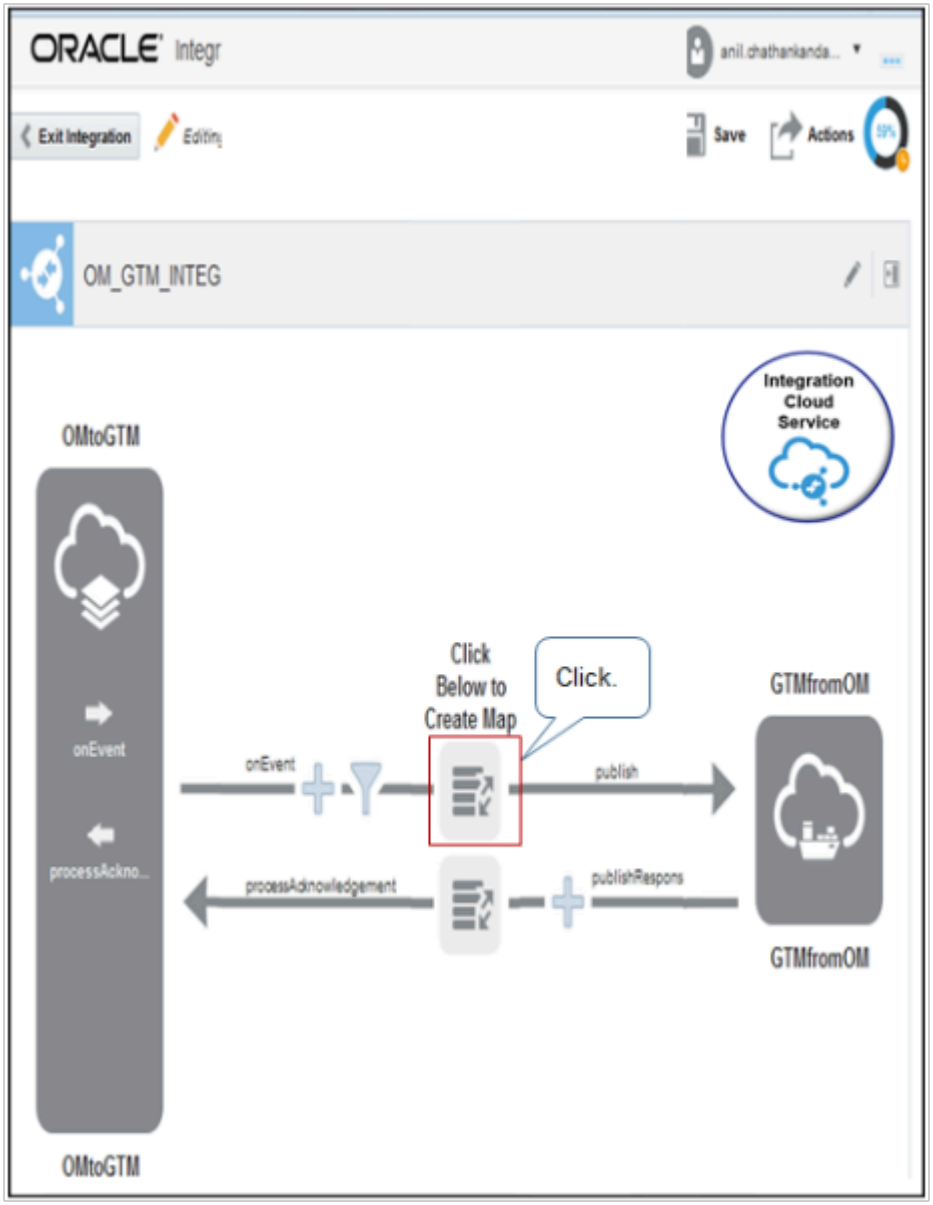
Attribute	Value
Type	Delayed
Business Object	Orchestration Order: OrderFulfillmentResponseService
Operation	processAcknowledgement

3. Search for the GTMfromOM connection you created earlier, then drag and drop it onto the target. The target is on right side of the integration.
4. In the dialog that displays, enter details.

Attribute	Value
Name	Identify the end point, such as GTM_Destination.
Operations attribute on the Operations tab	Publish
Business Object attribute on the Operations tab	GtmTransaction

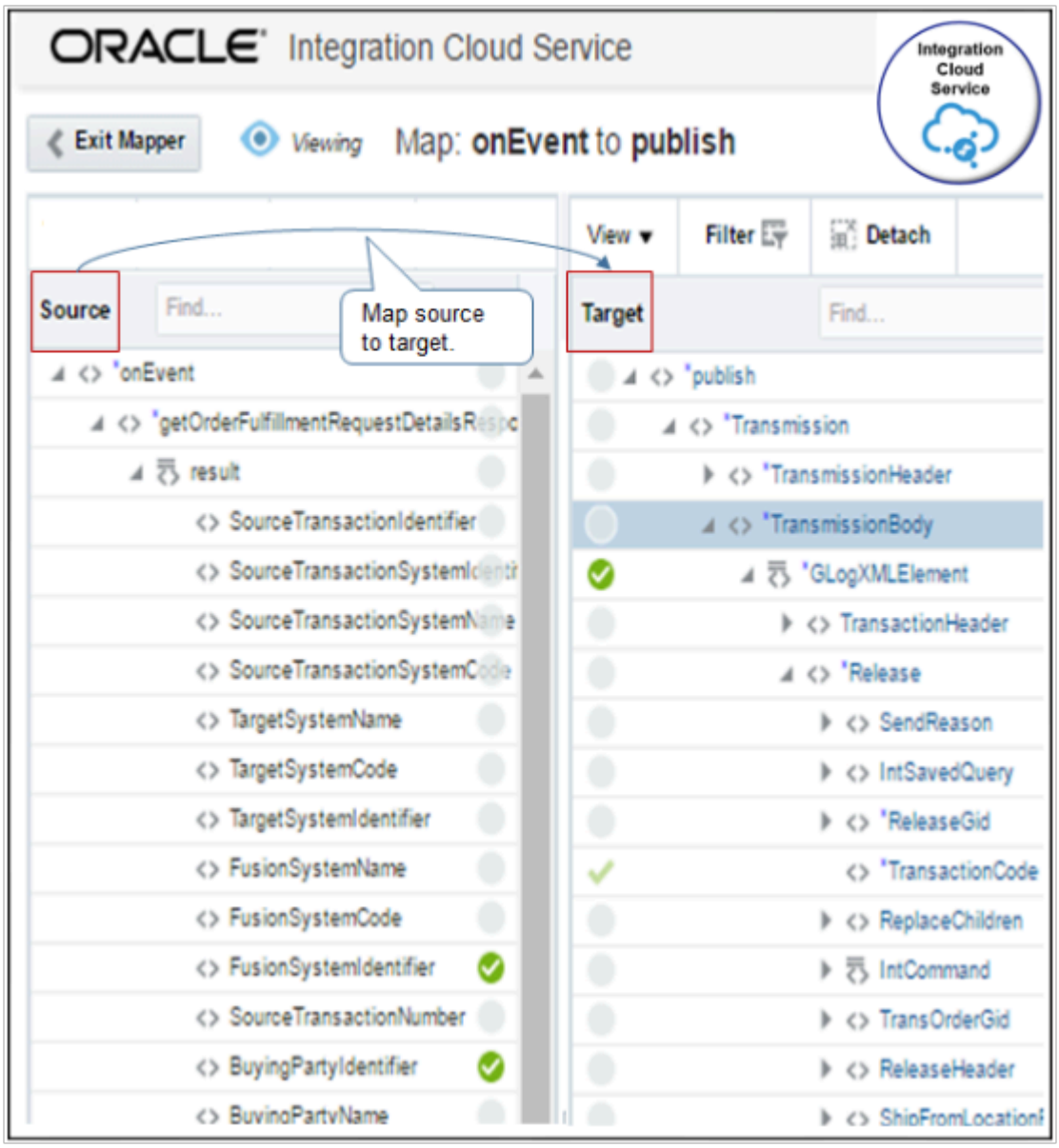
Communicate Attributes from Order Management to Global Trade Management

1. Under the text Click Below to Create Map, click the **icon** on the top line. Its the onEvent to publish line that points from OMtoGTM to GTMfromOM.



2. Finish the mapping, then save and close the mapper.

   For example:



   Note

   - Get a list of the attributes that you can map. For details, see *Operations and Attributes You Can Use with the Receive Order Request Service*.

Communicate Attributes from Global Trade Management to Order Management

1. Under the text Click Below to Create Map, click the **icon** on the lower line. Its the publishResponse to processAcknowledgement line that points from GTMfromOM to OMtoGTM.

2. Finish the mapping, then save and close the mapper.

   Use this example payload to help determine the mapping that you need to create. Global Trade Management sends this payload as a response to the call from Order Management.

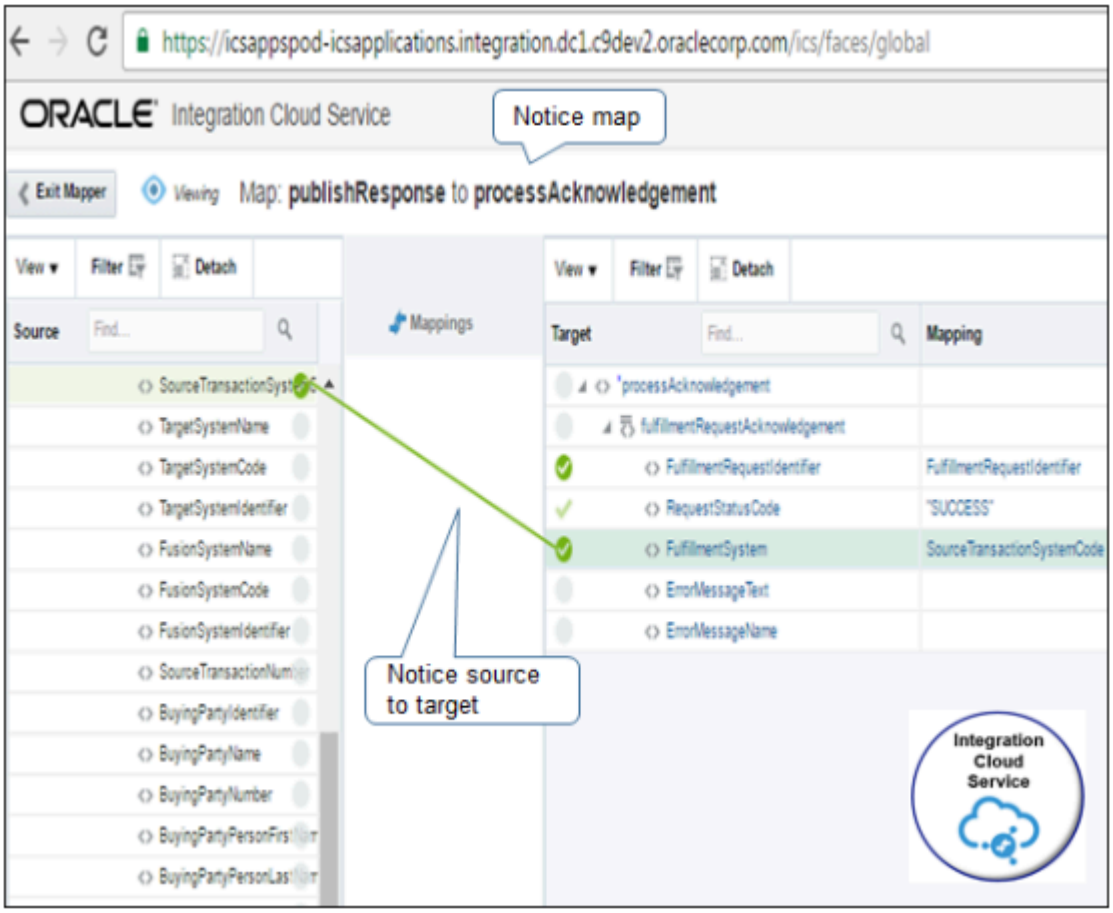
```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/scm/doo/taskLayer/commonService/types/" xmlns:com="http://xmlns.oracle.com/apps/
scm/doo/taskLayer/commonService/" xmlns:mod="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <soapenv:Body>
 <typ:processFulfillmentResponse>
 <typ:responseMessageHeader>
 <com:IntegrationContextCode>DOO_TradeCompliance</com:IntegrationContextCode>
 <com:FulfillmentSystemResponseIdentifier/>
 </typ:responseMessageHeader>
 <typ:fulfillLineList>
 <com:OrderNumber/>
 <com:FulfillLineIdentifier>300100175497843</com:FulfillLineIdentifier>
 <com:TradeComplianceScreeningResultCode>ORA_PASSED</com:TradeComplianceScreeningResultCode>
 <com:TradeComplianceScreeningDate>2018-11-23T12:12:12</com:TradeComplianceScreeningDate>
 <com:FulfillmentDetail>
 <com:TradeComplianceTypeCode>ORA_SANCTIONED_TERR</com:TradeComplianceTypeCode>
 <com:TradeControlCode/>
 <com:TradeComplianceComments/>
 <com:TradeComplianceScreeningResultCode>ORA_PASSED</com:TradeComplianceScreeningResultCode>
 </com:FulfillmentDetail>
 <com:FulfillmentDetail>
 <com:TradeComplianceTypeCode>ORA_RESTRICTED_PARTY</com:TradeComplianceTypeCode>
 <com:TradeControlCode/>
 <com:TradeComplianceComments/>
 <com:TradeComplianceScreeningResultCode>ORA_PASSED</com:TradeComplianceScreeningResultCode>
 </com:FulfillmentDetail>
 <com:FulfillmentDetail>
 <com:TradeComplianceTypeCode>ORA_TRADE_CONTROL</com:TradeComplianceTypeCode>
 <com:TradeControlCode/>
 <com:TradeComplianceComments/>
 <com:TradeComplianceScreeningResultCode>ORA_PASSED</com:TradeComplianceScreeningResultCode>
 </com:FulfillmentDetail>
 </typ:fulfillLineList>
 </typ:processFulfillmentResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

   Note

   - This mapping provides the input to the processAcknowledgement operation of service OrderFulfillmentResponseService. Here are the attributes that it sends.
     - TaskInstanceId
     - SourceTransactionSystemCode
     - RequestStatusCode with a value of SUCCESS

- The integration uses this return mapping only when it calls processAcknowledgement, and only when Global Trade Management sends a successful response.

For example:



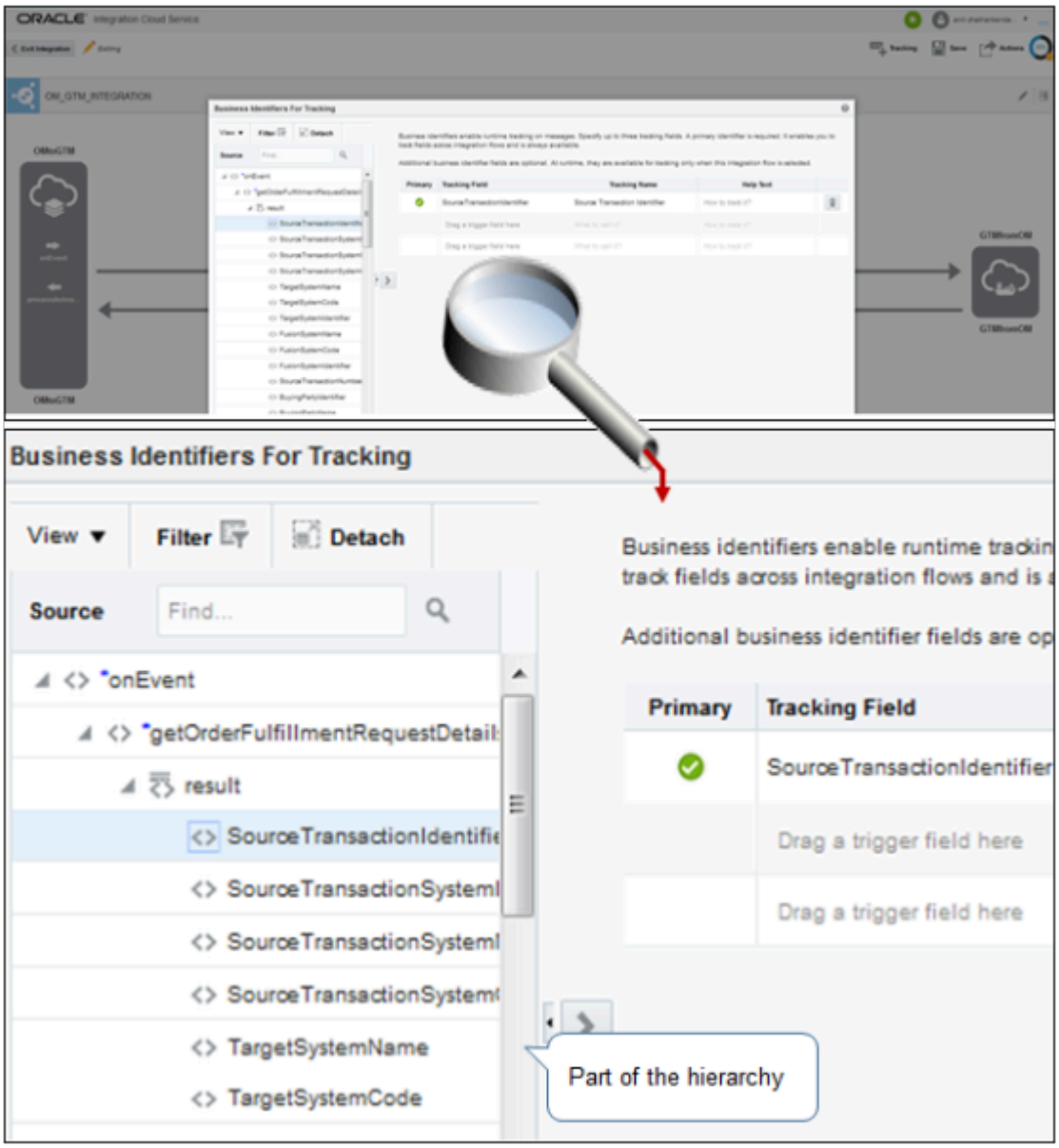
Add Business Identifiers and Activate Integration

1. Add business identifiers for tracking.

   On the menu at the top right of the edit integration page, click **Tracking**, drag a payload field from the Source, and drop it onto the Tracking Field section.

   Integration Cloud Service uses business identifiers to track messages at run time. You can use them to help identify and monitor instances of the integration.

   For example:



   The screen print is hard to read and displays only part of the hierarchy. Here's the Source hierarchy that it displays.

```
onEvent
 getOrderRequestFulfillmentDetails
 result
 SourceTransactionIdentifier
 SourceTransactionSystemIdentifier
 TargetSystemName
 TargetSystemCode
 TargetSystemIdentifier
 FusionSystemName
 FusionSystemCode
 FusionSystemIdentifier
 SourceTransactionNumber
 BuyingPartyIdentifier
 BuyingPartyName
 BuyingPartyNumber
```

2. Save and activate.
    - Click **Test** to test your integration, then click **Save**.
    - Activate the integration. As an option, in the Confirmation dialog, enable the Enable Trace option to log payloads in the runtime logs.

Related Topics

- Overview of Integrating Order Management with Transportation Management
- Overview of Orchestration Processes
- Use Integration Cloud Service with Order Management
- Overview of Setting Up Trade Compliance
- Privileges That You Need to Implement Order Management

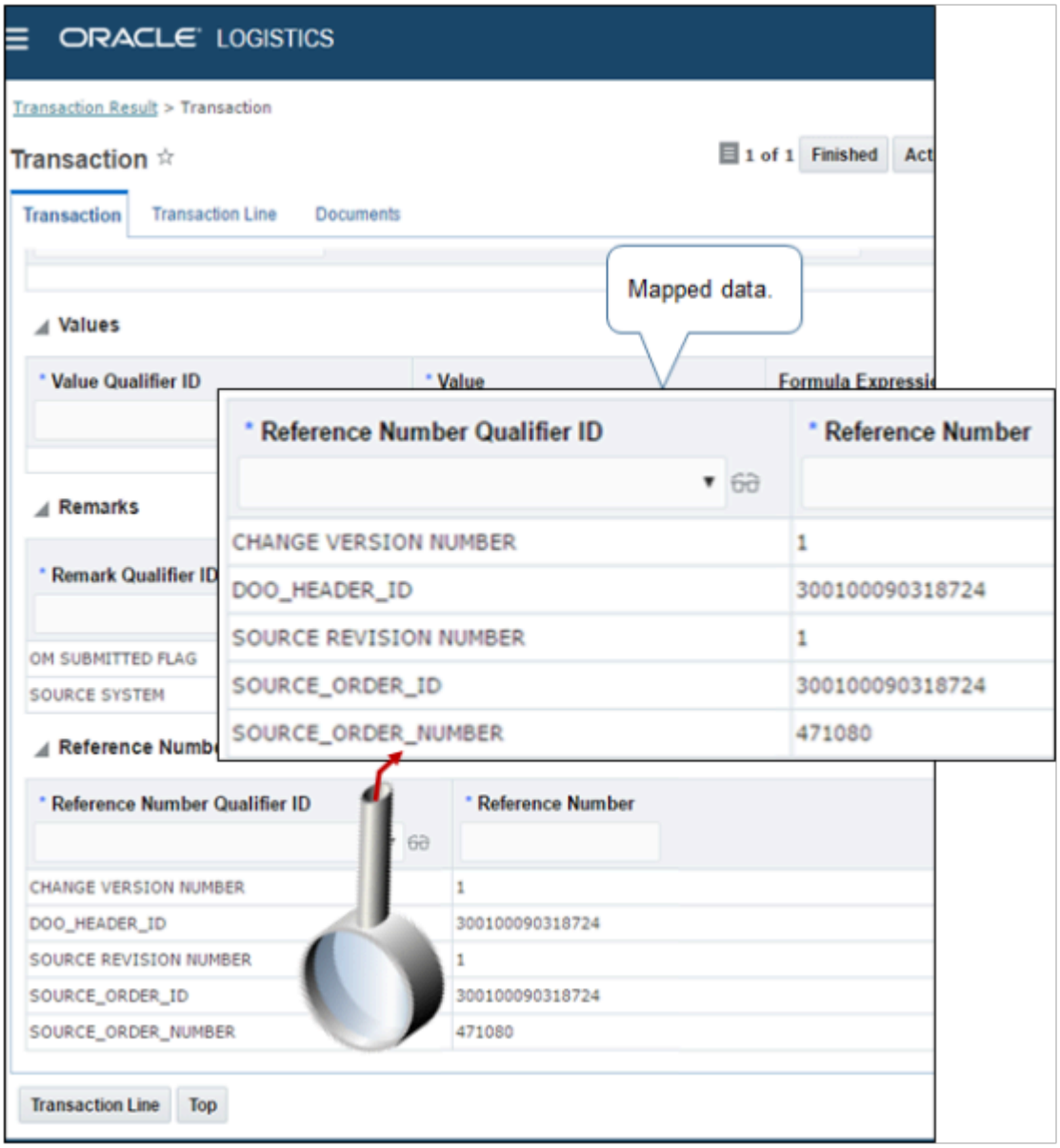
Get Details About Transactions with Global Trade Management

Get details for transactions that happen in an integration between Oracle Global Trade Management and Oracle Order Management.

Get Transaction Details in Global Trade Management

1. Sign into Global Trade Management.
2. On the Oracle Logistics page, navigate to **Trade Compliance Management > Trade Transaction Management > Trade Transaction**.
3. On the Transaction Finder page, enter the TransactionId of the sales order, then click **Search**.

4. On the Transaction page, examine the mapped data.

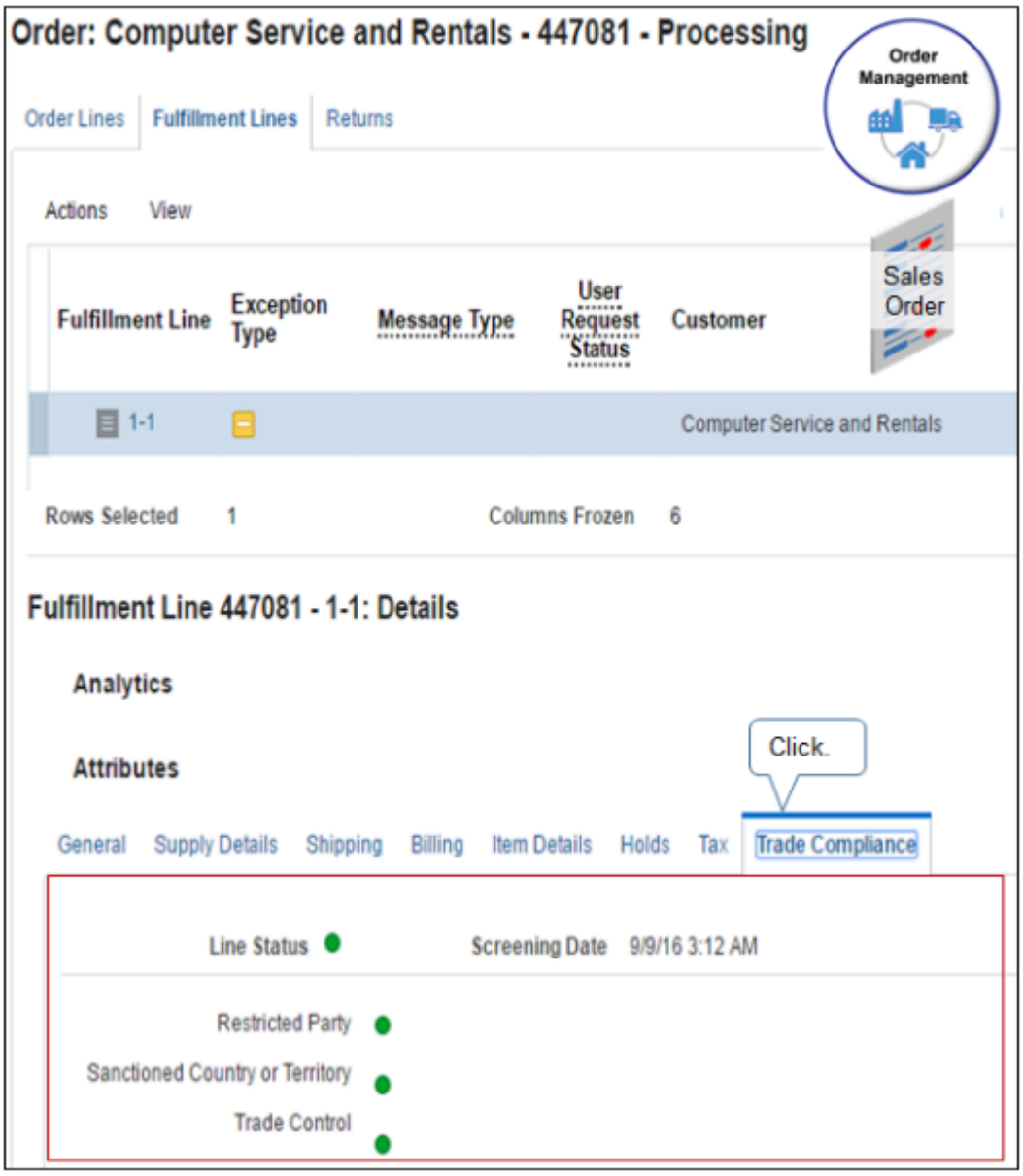


Here's how the flow from Order Management to Global Trade Management works.

1. Submit an order through an orchestration process that includes the GTM step.
2. The orchestration process reaches the create step, then uses a GTM event. Integration Cloud Service subscribes to the event.
3. The flow creates a new instance of Integration Cloud Service. The instance calls the mapping that maps output from OrderFulfillmentRequestService to the input to Global Trade Management, then calls the service in Global Trade Management.
4. The flow successfully establishes the integration, then calls the processAcknowledgement operation of service OrderFulfillmentResponseService. processAcknowledgement changes the fulfillment line status to DOO_GTM_AWAIT.

Get Transaction Details in Order Management

1. Sign into Order Management.
2. Open the sales order in a fulfillment view, then examine the Trade Compliance Status attribute on the order header.
3. Use the Trade Compliance tab in the fulfillment line detail area.



Here's how the flow from Order Management to Global Trade Management works.

Lookup	Description
ORA_DOO_VALIDATION_RESULT	Contains predefined values for TradeComplianceScreeningResultCode.

Lookup	Description
	• Failed • Under Review • Passed Each response from trade compliance screening contains the screening result for the fulfillment line and details for each screening type for the fulfillment detail. It stores these details in the TradeComplianceScreeningResultCode attribute.
ORA_DOO_TRADE_COMPLIANCE_TYPE	Contains predefined values for TradeComplianceTypeCode. • Restricted Party • Sanctioned Country or Territory • Trade Control Each fulfillment detail in the response corresponds to a screening type that the TradeComplianceTypeCode attribute specifies in the fulfillment detail.
ORA_DOO_TRADE_CONTROL_CODE	Contains values for the TradeControlCode attribute in the fulfillment detail. Applies only when TradeComplianceTypeCode is ORA_TRADE_CONTROL.
ORA_TRADE_CONTROL	You must define values for this lookup.

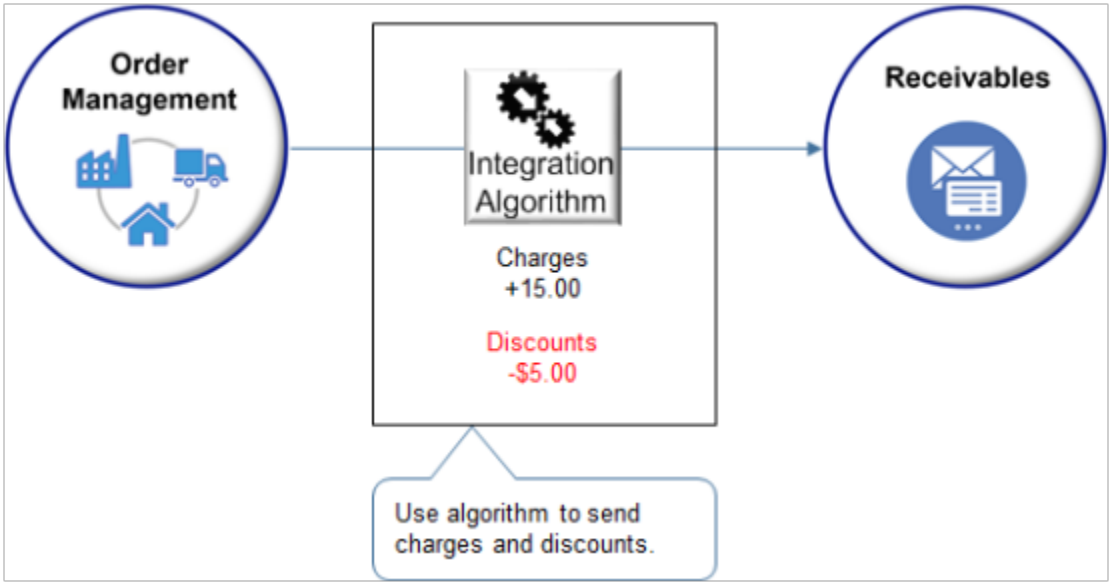
Receivables

Overview

Overview of Integrating Order Management with Accounts Receivable

Get an overview of how you can specify the way Order Management sends charges and charge lines to Oracle Accounts Receivable.

Set up the predefined Integration Algorithm for Sales Order Charges so it sends the values to Accounts Receivable that receivables needs to calculate various charges, such as how to calculate tax on the shipping charge of a shipping line.



Accounts Receivable uses only a single, one-time charge for your item. It combines all other charges into a single freight charge on the invoice header, and it doesn't tax any of these charges. If you need to tax them, then you must include each charge as a separate item in Order Management, and send each charge to Accounts Receivable as a separate line. As an alternative, you can use Integration Algorithm for Sales Order Charges and a service mapping. This chapter describes how to use the algorithm and the mapping.

For example, tax rules on freight might vary according to region or to details on the sales order. Determining tax manually might delay order fulfillment, introduce unnecessary tax, or result in an invoice error. Instead, you can:

- Specify one or more shipping charges for an order line to automate calculating tax, such as sending freight charges to Receivables to calculate the tax amount for freight.
- Calculate freight tax in Oracle Pricing, send it to Order Management, then display it on an order line in Order Management.
- Use the integration algorithm to help achieve tax compliance, minimize sales order exceptions, and improve accuracy for your billing.

Shipping Charges

In some prior updates, Order Management sent secondary charges, including freight charges, to Receivables. Receivables added the secondary charges together, then displayed them as a single freight charge on the invoice header. Receivables couldn't apply a tax on this single charge because it might need to tax individual charges at different tax rates, and some of these charges might not be taxable. Order Management can now send the shipping charge on a separate charge line.

- This feature adds a new charge line for each shipping charge for each item. For example, if the sales order contains three items, and if each item includes a shipping charge, then the invoice will contain six lines. Order Management will ship three lines, and the other three lines will contain freight charges.
- The charge line will include the item number and the description that you set up in the Product Information Management work area. To relate each freight charge to the shippable line, you can override the Description attribute and TranslatedDescription attribute so they reference the correct shippable item.
- If you override an attribute in the algorithm, then you must include the attribute in the Sources tab and the Services tab of the entity you're modifying in the service mapping.
- As an option, you can also modify discount lines.

- Make sure you set up the item in Product Information Management.

Shipping Example

Assume your fulfillment line has two shipping charges.

- $30 charge for shipping and handling
- $10 charge for insurance

Order Management will send each of these shipping charges separately to Receivables with the Line Type attribute set to Freight.

The Import AutoInvoice scheduled process combines these two shipping charges together with a value of $40 in the Freight attribute on the invoice header, and Receivables doesn't calculate tax on the shipping charge. For details, see *Update Intercompany Receivables Invoice Import Details*.

To capture shipping charges as revenue and calculate tax on them:

- You must update the service mapping from LINE_TYPE = FREIGHT to LINE_TYPE = LINE.
- If you want to have each shipping charge as a separate item, then you must also update the Inventory Item Id for each item.
- Import AutoInvoice will create a separate invoice line for each shipping charge in Receivables, and Receivables will calculate the amount and tax for each charge on two separate invoice lines.

Net Amount

Order Management sends only the total amount for each shipping charge to Receivables. It doesn't send discounts regardless of how you set the Send Discount Details to Billing Systems parameter. For details, see *Manage Order Management Parameters*.

Assume your fulfillment line has a quantity of 5, a $10 shipping charge, and a $2 discount. The total amount is $50 (5 multiplied by $10), and the discount is $10 (5 multiplied by $2). Order Management will send a $40 shipping charge to Receivables ($50 minus $10).

Examples of Integrating with Accounts Receivable

For an example, see *Use Integration Algorithms to Implement Complex Logic*.

For details about managing order numbers that you import when you integrate with Receivables, see *Keep Your Order Number When You Import*.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Use Integration Algorithms to Implement Complex Logic
- Pricing Algorithms
- Service Mapping
- Create a Sandbox So You Can Edit Service Mappings

Integrate Order Management with Accounts Receivable

Specify how Order Management sends charges and charge lines to Oracle Accounts Receivable.

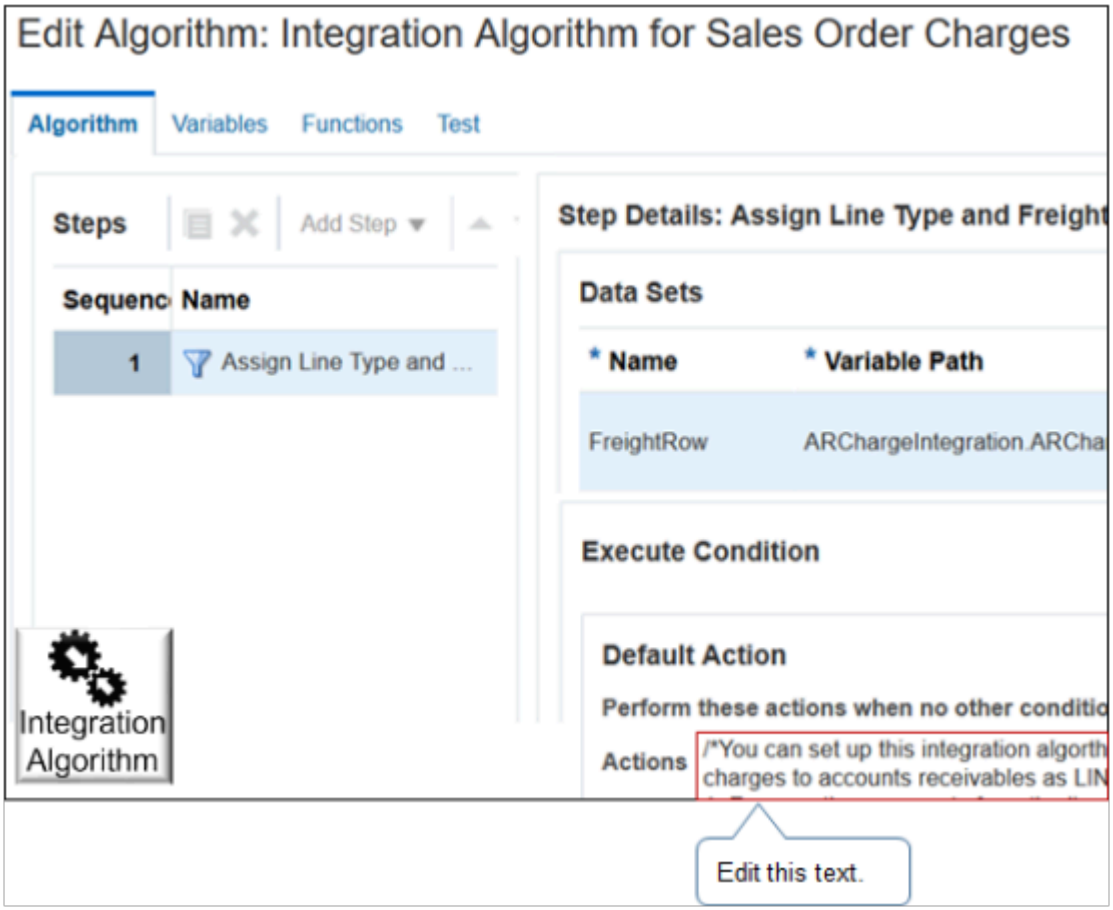
This topic uses example values. You might need different values, depending on your business requirements.

Summary of the Set Up

1. Edit integration algorithm.
2. Manage service mapping.
3. Enable feature.

Edit Integration Algorithm

1. Make sure you have the privileges that you need to administer Order Management.
2. Go to the Pricing Administration work area.
3. Click **Tasks**, then, under Order Management Configuration, click **Manage Algorithms**.
4. On the Manage Algorithms page, click **Integration Algorithm for Sales Order Charges**.
5. On the Edit Algorithm page, in the Step Details area, in the Execute Condition area, in the Default Action area, locate the code in the Default Actions window.



6. Click **Edit** immediately to the right of the Actions window.
   Notice the text that Actions contains.

```
/*You can set up this integration algorthim so it uses accounts receivable to calculate tax on the
 shipping charge. To do this, you must interface taxes and charges to accounts receivable as LINE and
 not as FREIGHT. Do this:
1. Remove the comments from the lines below.
2. Publish your integration algorithm.
3. Use the Manage Service Mappings page to map your integration algorithm to service InvoiceService.
```

```
If an integration algorithm already maps to InvoiceService, then do not map your your integration
 algorithm to InvoiceService. Instead, incorporate steps from your integration algorithm into the
 integration algorithm that already maps to InvoiceService.
*/
//FreightRow.LineType = ChargeRow.ChargeApplyTo == "SHIPPING" ? "LINE" : FreightRow.LineType
//FreightRow.InventoryItemId = ChargeRow.ChargeApplyTo == "SHIPPING" ? '<Replace with Freight Inventory
 Item Id>' : null
```

7. Remove the commented instructional text, remove the two forward slashes (//) from this code, then add the line and Id details for your shipping line:

   ```
   //FreightRow.LineType = ChargeRow.ChargeApplyTo == "SHIPPING" ? "LINE" : FreightRow.LineType
   //FreightRow.InventoryItemId = ChargeRow.ChargeApplyTo == "SHIPPING" ? '<Replace with Freight Inventory
    Item Id>' : null
   ```

   where

   - **LINE**. Specifies the shipping line where you're requesting Receivables to calculate tax on the shipping charge.
   - **<Replace with Freight Inventory Item Id>**. Specifies the Item Id of the freight inventory item that Receivables will use to calculate tax.

   You can also add a description.

   For example, here's some code that specifies to use item 123456789.

   ```
   FreightRow.LineType = ChargeRow.ChargeApplyTo == "SHIPPING" ? "LINE" : FreightRow.LineType
   FreightRow.InventoryItemId = ChargeRow.ChargeApplyTo == "SHIPPING" ? '123456789' : null
   ```

   As an option, you can specify other attributes. For details, see *Include Price, Discounts, and Shipping Charges in Your Payloads*.
8. Click **Save and Close**.
9. On the Manage Algorithms page, click **Actions > Publish**.

Manage Service Mapping

You use the Manage Service Mappings page to map your integration algorithm to the InvoiceService service.

If you already set up an integration algorithm that maps to InvoiceService, then don't map your integration algorithm to InvoiceService. Instead, incorporate steps from your integration algorithm into the integration algorithm that already maps to InvoiceService. For details about how to map your integration algorithm to InvoiceService, see *Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications*.

Manage service mapping.

1. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
2. In the Tasks pane, under Order Management Configuration, click **Manage Service Mappings**.
3. On the Manage Service Mappings page, click **FulfillmentIntegration**.
4. Set up the source for the interface line.
   - Click **Sources**.
   - Click the **row** that contains InvoiceSource in the Source column.
   - In the Entity Mappings list, click the **row** that contains ARChargeInterfaceLine in the Entity column.
   - In the Attribute Mappings list, add these attributes.
     - InventoryItemId

        - LineType

5. Set up the service for the interface line.
    - Click **Services**.
    - Click the **row** that contains InvoiceService in the Service column.
    - In the Entities list, click the row that contains ARChargeInterfaceLine in the Entity column.
    - In the Attribute list, add these attributes.
        - InventoryItemId
        - LineType
6. Set up the source for the charge.
    - Click **Sources**.
    - Click the **row** that contains InvoiceSource in the Source column.
    - In the Entity Mappings list, click the **row** that contains Charge in the Entity column.
    - In the Attribute Mappings list, add an attribute.

Attribute	View Object Attribute
ChargeApplyTo	ChargeAppliesTo

7. Set up the service for the charge.
    - Click **Services**.
    - Click the **row** that contains InvoiceSource in the Service column.
    - In the Entities list, click the **row** that contains Charge in the Entity column.
    - In the Attribute list, add an attribute.

Attribute	Value
Attribute	ChargeApplyTo

    - Click **Save and Close**.

Enable Feature

Enable the Invoice option in the Enable Custom Payloads for Downstream Integration feature. For details, see *Get Started with Integrating Order Management with Other Oracle Applications*.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Get Started with Integrating Order Management with Other Oracle Applications
- Pricing Algorithms
- Service Mapping
- Create a Sandbox So You Can Edit Service Mappings

Integrate Order Management with Accounts Receivable When You Use Financial Orchestration

If the receiving business unit on the order header is different than the business unit for the selling profit center on the order line, then Receiving sends an event to Supply Chain Financial Orchestration, and Financial Orchestration might display an error.

```
The business event could not be processed because no valid financial orchestration flow is found for the
 source document information (FOS-3465118)
```

If you use Financial Orchestration, then you can use this topic to prevent the error.

For background details, see *Overview of Integrating Order Management with Accounts Receivable*.

Summary of the Set Up

1. Create an integration algorithm.
2. Map your entity to the source.
3. Map entities to the service.
4. Test your set up.

This topic uses example values. You might need to use different values for your implementation.

Create an Integration Algorithm

1. 1. Go to the Pricing Administration work area, Click Tasks, and then, under Order Management Configuration, click Manage Algorithms. For details about how, see *Create Your Service Mapping*.
2. On the Create Algorithm page, set the values, then click **Save**.

Attribute	Value
Name	Integrate Order Management with Receiving Custom
Description	Use this integration algorithm to send the business unit that we use for the selling profit center as the receiving business unit.

3. Click **Add Step > Condition Action**.
4. In the Date Sets area, add two data sets.

Name	Variable Path	Primary	Cardinality	Date Set Join
Fline	PCVR.FulfillLine	Contains a check mark	Leave empty	Leave empty
ReceiptAdvice	PCVR.ReceiptAdvice	Doesn't contain a check mark	Zero or one	[HeaderId: {Fline.HeaderId}]

Name	Variable Path	Primary	Cardinality	Date Set Join

5. In the Execute Condition area, click **Add Condition > Default Action**.
6. In the Edit Actions window, enter this code.

```
ReceiptAdvice.BUId = Fline != null ? Fline.SellingProfitCenterBUId : null;
```

7. Click **Save**.
8. Test and publish.

Map Your Entity to the Source

1. Enter a sandbox. See *Create a Sandbox So You Can Edit Service Mappings*.
2. Click **Tasks**, and then, under Order Management Configuration, click **Manage Service Mappings**, then click **FulfillmentIntegration**.
3. Click **Sources**.
4. Click **ReceiptSource**.
5. In ReceiptSource details area, select the ReceiptAdvice entity.
6. In ReceiptAdvice details, click **Action > Add Row**.
7. Select the BUId attribute.
8. In the ReceiptSource details area, select the **FulfillLine** entity.
9. In FulfillLine details, click **Action > Add Row**.
10. Select the SellingProfitCenterBUId attribute.

Map Algorithm and Attribute to the Service

1. Click the **Services** tab.
2. In the row that contains ReceiptService in the service column, set the values, then click **Save**.

Attribute	Value
Implementation Type	Algorithm
Implementation	Integrate Order Management with Receiving Custom This is the name of the algorithm that you created earlier in this procedure.

3. In ReceiptServices details, select the FulfillLine entity.
4. Click **Actions > Add Row**.
5. Select **SellingProfitCenterBUId**.
6. In ReceiptServices details, select the **ReceiptAdvice** entity.
7. Click **Actions > Add Row**.
8. Select the **BUId** attribute.

Test Your Set Up

Create a referenced return.

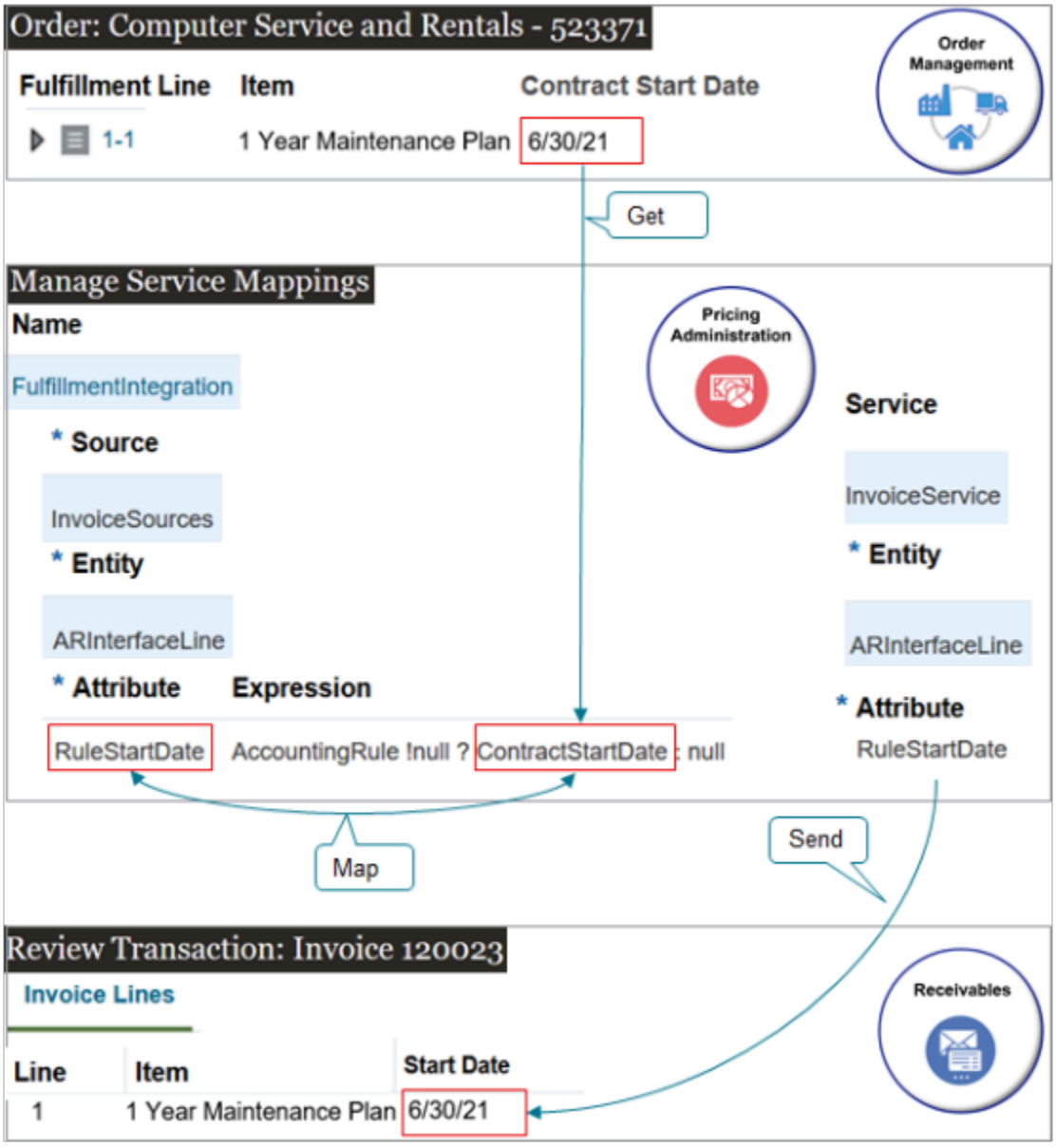
1. Go to the Order Management work area and create a sales order.

2. Confirm that Order Management shipped the order.
3. Create a referenced return for the order that you just created.
4. Set the business unit on the order header to Vision Operations.
5. Set the business unit for the selling profit center on the order line to Singapore Distribution Center.
6. Put away the referenced return.
7. Go to the Scheduled Processes work area, then run the *Send Receipt Confirmation* scheduled process.
8. Confirm that Financial Orchestration doesn't receive the event and doesn't display the error.

Integrate Attribute Values

Integrate a Predefined Attribute Between Order Management and Accounts Receivable

In this example you use a service mapping to map the Contract Start Date from the fulfillment line of a sales order, to the Start Date on the invoice line of an invoice in Accounts Receivable.



Note

- You use the predefined FulfillmentIntegration service mapping. Its already set up to do most of the work for you.
- You use predefined objects in FulfillmentIntegration to create the mapping, such as the InvoiceSources source, the ARInterfaceLine entity, and the InvoiceService service.
- You don't have to significantly modify the predefined objects. All you need to do is add and map the attribute.
- At runtime, the service mapping gets the value of the Contract State Date from the fulfillment line, maps it to the RuleStartDate, then sends the value to Receivables.
- Receivables displays the value in the Start Date attribute on the invoice line.

Summary of the Setup

1. Edit service mapping.
2. Test your set up.

To view a multimedia demonstration that's similar to this set up, go to *Order Management Enhancements*. The demonstration starts at 45:15.

Edit Service Mapping

Edit a service mapping so Order Management can send attribute details to Oracle Receivables.

1. Make sure you have the privileges that you need to administer Order Management. These privileges allow you to access the service mapping that you use to set up the integration.
2. Go to the Pricing Administration work area.
3. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
4. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
5. Modify the source.

   Specify how the payload that you send to Accounts Receivable will get the Contract Start Date from the fulfillment line on the sales order. You will write the logic to make sure you populate the RuleStartDate with the correct value.

   - Click **Sources**.
   - In the Sources list, click the **row** that has the value.

Attribute	Value
Source	InvoiceSources

   - On the Entity Mappings tab, notice that most entities use the FulfillLineVO view object to get their attribute values.
   - In the InvoiceSources Details area, on the Entity Mappings tab, click the **row** that has the value.

Attribute	Value
Entity	ARInterfaceLine

- In the ARInterfaceLine Details area, on the Attribute Mappings tab, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Attribute	RuleStartDate This value identifies the attribute that you will as the target attribute.
Expression	`AccountingRule !null ? ContractStartDate : null` here's what this code means. If the AccountingRule. - Contains a value, then use the value in ContractStartDate to populate RuleStartDate - Doesn't contain a value, then leave ContractStartDate empty You're getting the value for RuleStartDate from the FulfillLineVO view object, so you examine an attribute in the FulfillLineVO view object to see if it has a value.

6. Map your attributes to the service that Order Management uses to communicate with Oracle Receivables.
   - Click **Services**.
   - In the Services list, click the **row** that has the value.

Attribute	Value
Service	InvoiceService

   - In the InvoiceService Details area, click the **row** that has the value.

Attribute	Value
Entity	ARInterfaceLine

   - At the bottom of the page, in the ARInterfaceLine area, click **Actions > Add Row** to add the RuleStartDate attribute so Order Management can communicate it through the InvoiceService.

Attribute	Value
Attribute	RuleStartDate

   Now your payload will include the RuleStartDate attribute.
   - Click **Save and Close**.

Test Your Setup

1. Create a sales order.
    - Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

    - Add an order line.

Attribute	Value
Item	AS54888
Quantity	1

    - Add another order line.

Attribute	Value
Item	1 Year Maintenance Plan
Quantity	1

      Assume this item has a Contract Start Date of June 30, 2021.
    - Click Submit.

      Assume your order number is 523371, and that the orchestration process schedules and ships the order lines.

2. Verify that Order Management correctly sends your data to Oracle Receivables.
    - Make sure you have the privileges that you need manage Accounts Receivable.

      For details, see *Privileges That You Need to Implement Order Management*.
    - Go to the Billing work area, then click **Tasks > Manage Transactions**.

      For details, see *Set Receivables Transaction for Reverse Billing*.
    - On the Manage Transactions page, search for the transaction.

Attribute	Value
Business Unit	Vision Operations
Transaction Source	Distributed Order Orchestration
Reference	523371

    - On the Review Transaction page, in the Invoice Details area, verify the value for the invoice line that has the warranty.

Attribute	Value
Start Date	6/30/21

Related Topics

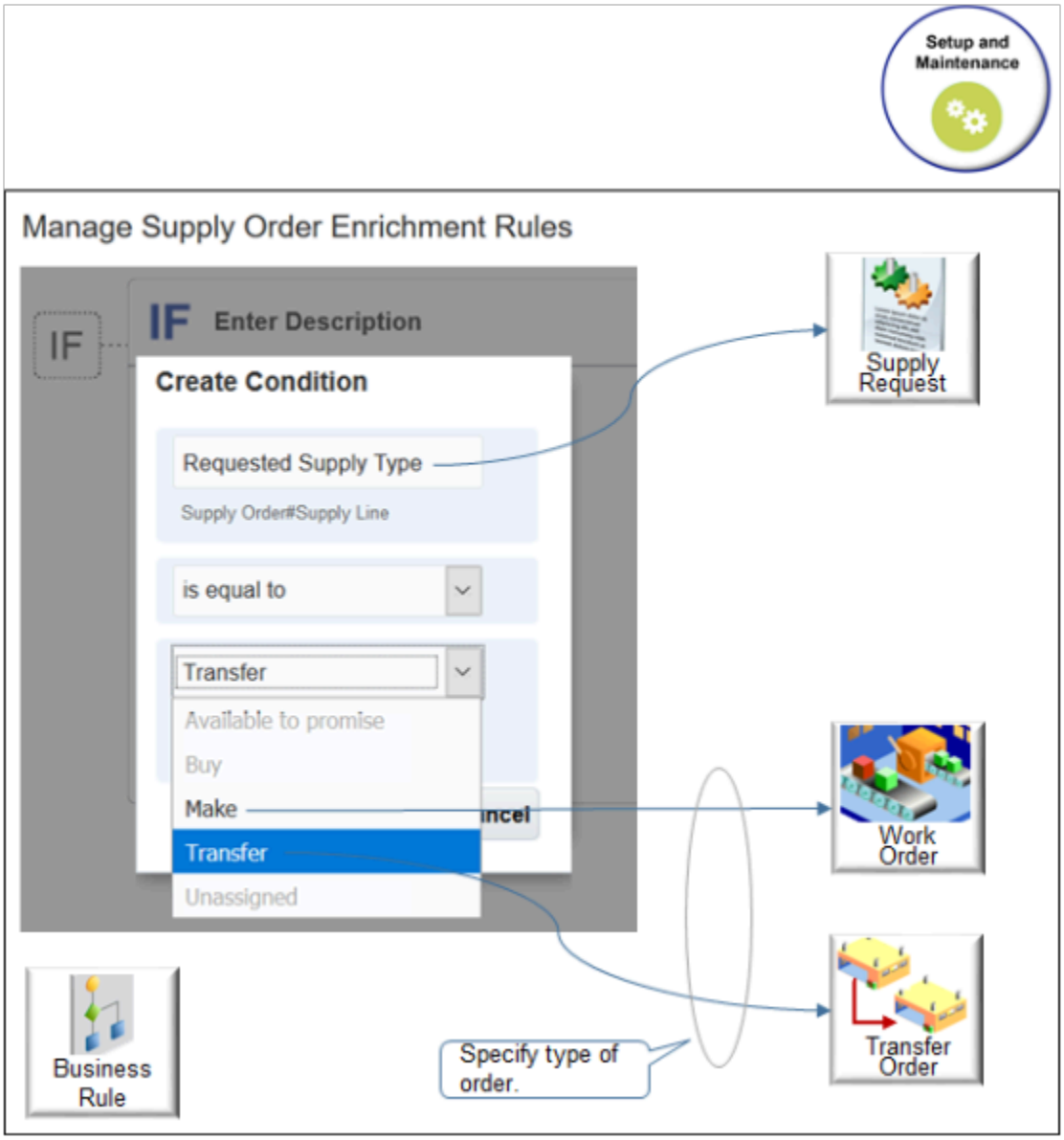
- Overview of Integrating Order Management with Other Oracle Applications
- Privileges That You Need to Implement Order Management
- Service Mapping
- Create a Sandbox So You Can Edit Service Mappings
- Requirements for Completing a Receivables Transaction

Integrate Your Own Header Attributes Between Order Management and Accounts Receivable

In this example, you use an extensible flexfield on the order header in Order Management to capture the primary salesperson.

You then send that value through the ResourceSalesPersonId attribute on an interface line to Accounts Receivable so receivables can display it on an invoice.

Here's how it works.



Note

1. You create an extensible flexfield in the Setup and Maintenance work area.
2. The Order Entry Specialist uses your extensible flexfield to set the Primary Salesperson on the order header in the Order Management work area.
3. You use the Pricing Administration work area to create a service mapping. The mapping uses the view object and the view attribute from the flexfield to map it to Accounts Receivable.
4. You use the Pricing Administration work area to create an integration algorithm that tells Order Management to include the salesperson from the flexfield when it sends the payload to Accounts Receivable.
5. Your accounts receivable manager uses the Billing work area to open the invoice for the sales order and view the salesperson in the Salesperson attribute.

Summary of the Setup

1. Prepare your setup.
2. Create extensible flexfield.
3. Identify your view object.

4. Edit service mapping.
5. Modify integration algorithm.
6. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Prepare Your Setup

1. Enable the Invoice option in the Enable Custom Payloads for Downstream Integration feature.

   For details, see *Get Started with Integrating Order Management with Other Oracle Applications*.
2. Create a sand box, and activate the Service Mappings tool and the Flexfields tool in the sand box.

   For details, see *Create a Sandbox So You Can Edit Service Mappings*.

Create Extensible Flexfield

Create the extensible flexfield that your Order Entry Specialist will use to set the primary salesperson on the order header.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Extensible Flexfields
2. On the Manage Order Extensible Flexfields page, search for the value.

Attribute	Value
Name	Header Information This is a predefined extensible flexfield that you can use on the order header.

3. In the search results, click **Actions > Edit**.
4. Create the context.
   - On the Edit Extensible Flexfield page, click **Manage Contexts**.
   - On the Manage Contexts page, click **Action > Create**.
   - On the Create Context page, set the values, then click **Save and Close**.

Attribute	Value
Display Name	Primary Salesperson
Code	Primary Salesperson
API Name	primarySalesperson

Attribute	Value
Enabled	Enabled.
Behavior	Single Row

- In the Context Usages area, click **Actions > Create**, then set the values.

Attribute	Value
Name	Additional Header Information

- Click **Save**.
- In the Context Sensitive Segments area, click **Actions > Create**.
- On the Create Segment page, set values, then click **Save and Close**.

Attribute	Value
Name	Primary Salesperson
Code	Primary Salesperson
API Name	primarySalesperson
Enabled	Enabled.
Data Type	Character
Table Column	Pick one that's available.
Value Set	30 Characters
Prompt	Primary Salesperson
Display Type	List of Values

- Click Save and Close on the current page and subsequent pages until you're on the Edit Extensible Flexfield page.

- In the Category area, click the **row** that has Additional Header Information in the Display Name attribute.
- In the Additional Header Information Details area, on the Associated Context tab, click **Actions > Select and Add**.
- Add the Primary Salesperson context.
- On the Edit Extensible Flexfield page, click **Save and Close**.
- On the Manage Order Extensible Flexfields page, select the **line** in the search results that contains Header Information in the Name attribute, then click **Deploy Flexfield**.
- In the Confirmation dialog, verify that the deployment finished successfully, then click **OK**.

Identify Your View Object

Get the name of the view object and attribute that you will need in your service mapping.

1. Download and open the extensible flexfield.
   - Click **Actions > Download Flexfield Archive**, wait for processing to finish, then, in the Confirmation dialog, click **Download**.
   - In the dialog that displays, save the file to your preferred location.
   - Use your browser's download feature to navigate to the archive.

     For example, if you're using Firefox, click the **down arrow** on the top banner, then navigate to the 10008_DOO_HEADERS_ADD_INFO file, such as `C:\Users\your_name\Downloads\10008_DOO_HEADERS_ADD_INFO.zip`.
   - Unzip 10008_DOO_HEADERS_ADD_INFO.zip.
   - Navigate to folder `C:\Users\user_name \Downloads\ 10008_DOO_HEADERS_ADD_INFO.zip\oracle\apps\scm\doo\processOrder\flex\headerContextsB\view`.
   - Use an XML editor to open the `HeaderEffBPrimary__Sales__RepresentativeprivateVO.xml` file.
2. Identify the name of the view object.
   - Find the Name attribute of the view object, then note its value. It should be on about the 4th line of code.

     ```
     <ViewObject
      xmlns="http://xmlns.oracle.com/bc4j"
      Name="HeaderEffBPrimary__Sales__RepresentativeprivateVO"
     ```

     In this example, the value is `HeaderEffBPrimary__Sales__RepresentativeprivateVO`.
3. Find the Name attribute of the view attribute, then note its value.

   ```
   ViewAttribute Name="primarySalesperson"
   ```

   In this example, the value is `primarySalesperson`.
   The flexfield archive has a number of attributes, so make sure you identify the correct one.

Edit Service Mapping

Edit the service mapping that Order Management uses to integrate with Accounts Receivable.

1. Go to the Pricing Administration work area.
2. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
3. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
4. Add an entity for the extensible flexfield.

- On the Edit Service Mappings page, on the Entities tab, click **Actions > Add Row**, then set the values.

Attribute	Value
Entity	HeaderEFF_Custom You must include the `_Custom` suffix with any new entity that you create. You must include the `EFF_Custom` suffix with any new entity that you create for an extensible flexfield.
Description	Extensible flexfield that stores the salesperson on the order header.

- In the details area, add your attributes, then click **Save**.

Attribute	Type
HeaderId_Custom	Long
SalesPersonId_Custom	Long

Enable the Allow Null attribute. Leave all other attributes at the current values.

You must use the same Type for each attribute that you specify for the corresponding attribute in the descriptive flexfield. If you don't use the same data type, then your service mapping might not fail when compiling but will fail in your runtime environment. The runtime failure might be difficult to troubleshoot because you might not encounter any errors but your set up might not work as you expected.

5. Modify the source.
   - Click **Sources**.
   - In the Sources list, click the **row** that has the value.

Attribute	Value
Source	InvoiceSources

Attribute	Value

- In the InvoiceSources Details area, click **View > Columns**, then enable the Joined Entity and Joined Entity Attribute.
- On the Entity Mappings tab, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	HeaderEFF_Custom
Type	View Object
View Object	HeaderEffBPrimary__Sales__RepresentativeprivateVO This value is the name of the view object that you noted earlier in this procedure when you downloaded the flexfield archive.
Query Type	Join
Query Attribute	HeaderId The invoice service primarily references the fulfillment line, so you query for the HeaderId from the fulfillment line. You use the Joined Entity to create a relationship between the HeaderId query attribute and the fulfillment line, and you use the Joined Entity Attribute to specify how to identify the joined entity.
Joined Entity	FulfilLine
Joined Entity Attribute	FulfilLineId

- On the Attribute Mappings tab, add your custom attributes, then click **Save**.

Attribute	View Object Attribute
HeaderId_Custom	HeaderId
SalesPersonId_Custom	primarySalesperson This value is the name of the ViewAttribute that you noted earlier in this procedure when you downloaded the flexfield archive.

Attribute	View Object Attribute

- On the Entity Mappings tab, click the **row** that has ARInterfaceLine in the Entity column.

  Order Management uses the ARInterfaceLine entity to communicate the attribute value in the payload that it sends to Accounts Receivable. You will add this attribute to the source. Later, you will also add it to the service.
- In the ARInterfaceLine Details area, add the attribute, then click **Save**.

Attribute	Value
Attribute	ResourceSalesrepId

Leave the View Object Attribute and the Expression attribute empty. The integration algorithm that you set up in this topic will use the extensible flexfield on the order header to determine these values.

6. Modify the service that Order Management uses to communicate with Accounts Receivable.
   - Click **Services**.
   - In the Services list, click the **row** that has the value.

Attribute	Value
Service	InvoiceService

   - In the InvoiceService Details area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	HeaderEFF_Custom
Alias	Leave empty
Read	Enabled
Write	Enabled

Leave the other attributes at their default values.

- At the bottom of the page, in the HeaderEFF_Custom Entities area, add your custom attributes so Order Management can communicate them through the InvoiceService.

Attribute	Value
HeaderId_Custom	Leave empty
SalesPersonId_Custom	Leave empty

Enable the Read attribute and Write attribute for each of the attributes that you add.

- In the InvoiceService Details area, click the **row** that has ARInterfaceLine in the Entity column.
- In the ARInterfaceLine Entities area, add the attribute.

Attribute	Value
Attribute	ResourceSalesrepId

- Click **Save and Close**.

Modify Integration Algorithm

To implement the logic you need, you will modify the integration algorithm from another help topic.

1. Create an integration algorithm. For details, see *Use Integration Algorithms to Implement Complex Logic*.
2. Add a new data set.

Attribute	Value
Name	HeaderEFF
Variable Path	ARIntegration.HeaderEFF_Custom
Primary	Not enabled.
Cardinality	Zero or one
Data Set Join	[HeaderId_Custom:(Fline.HeaderId)]

3. Add code to the conditional action. Add it as a new line at the end of the code.
   ```
   ARLine.ResourceSalesrepId=HeaderEFF.SalesPersonId_Custom
   ```
   This code says to get the value for the sales person from the SalesPersonId_Custom attribute on the HeaderEFF flexfield, then store it in the ResourceSalesrepId attribute in the ARLine data set.
4. Copy the code that you added to the conditional action, and paste it as a new line the end of the code in the default action.
   The algorithm you're modifying has a conditional action for order lines that contain a coverage item and a default action for lines that contain a covered item. You need to populate the ARLine data set regardless of whether the item on the line is a coverage item or a covered item, so you add the same code to the conditional action and the default action.
   You use this data set to make sure that you have the values you need for the extensible flexfield on the order header.
5. Publish the algorithm.

Test Your Setup

1. Create a sales order.
   - Go to the Order Management work area, create a sales order, then add an order line.
   - On the order header, click **Actions > Edit Additional Information**.
     The work area displays the Additional Information dialog. This dialog displays extensible flexfields that you create on the order header, by default.
   - Verify that the Additional Information dialog displays the Sales Representatives area, which is the flexfield context, and that this area displays the Primary Sales Rep attribute.
   - Set the Primary Sales Rep to Diane Cho.
   - Click Submit.
     Assume your order number is 54758.
2. Verify that Order Management correctly sends your data to Accounts Receivable.
   - Make sure you have the privileges that you need to manage Accounts Receivable. For details, see *Privileges That You Need to Implement Order Management*.
   - Go to the Billing work area, then click **Tasks > Manage Transactions**.
     For details, go to *Using Receivables Credit to Cash*, then search for Requirements for Completing a Receivables Transaction.
   - On the Manage Transactions page, search for the transaction.

Attribute	Value
Business Unit	Vision Operations
Transaction Source	Distributed Order Orchestration
Reference	54758

Attribute	Value

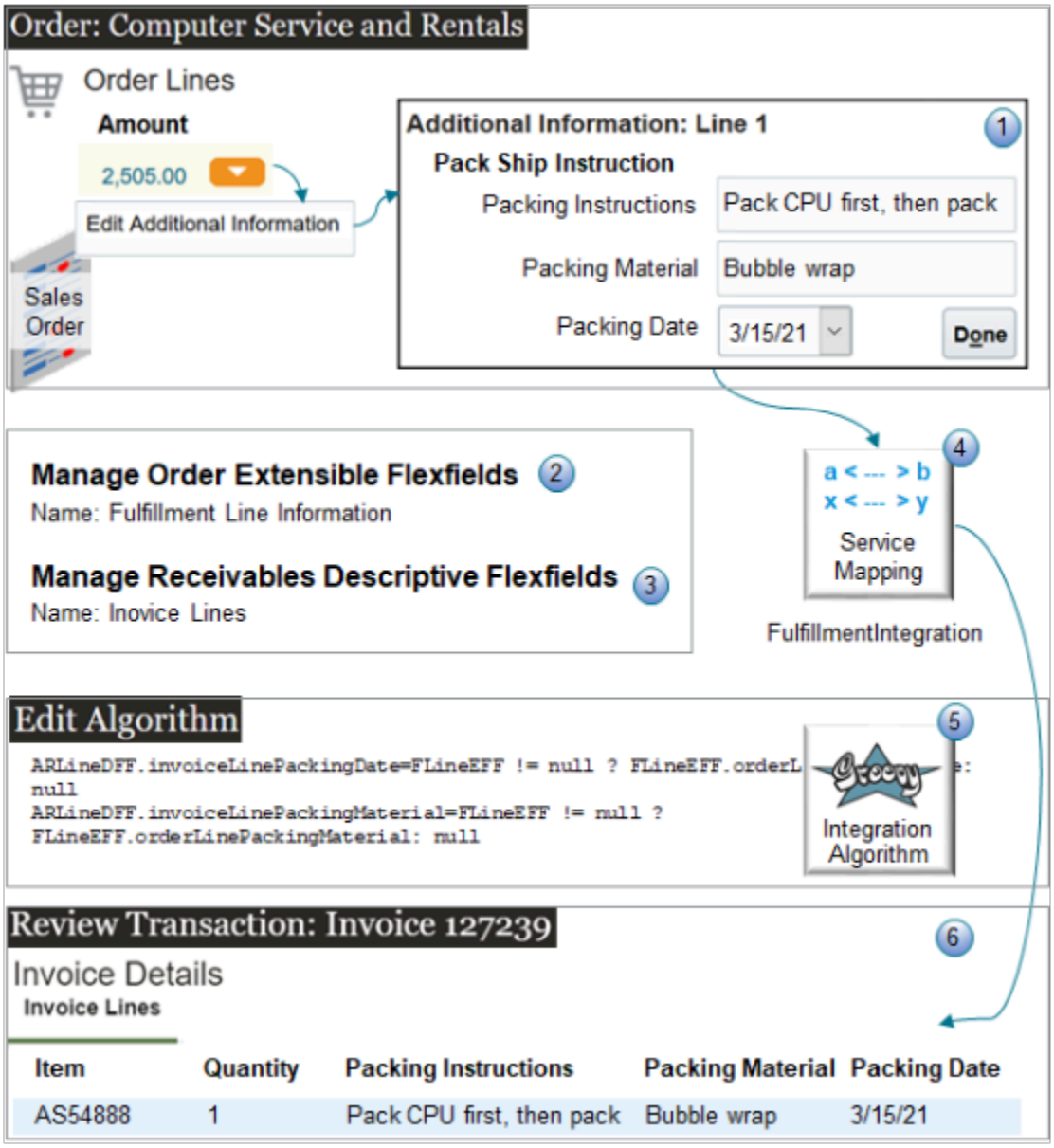
- On the Review Transaction page, in the General Information area, verify that the Salesperson attribute contains Diane Cho, which is the same value that you set on the order header in Order Management.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Privileges That You Need to Implement Order Management
- Use Integration Algorithms to Implement Complex Logic
- Create a Sandbox So You Can Edit Service Mappings
- Requirements for Completing a Receivables Transaction

Integrate Your Own Order Line Attributes between Order Management and Oracle Receivables

Use an extensible flexfield in Order Management to capture your own order line details, then send them to a descriptive flexfield so you can see these details on invoices in Oracle Receivables.



Note

1. Assume you need to add your own shipping details when you create an order line for a new sales order in Order Management. The Order Entry Specialist clicks the down arrow in the Amount column on the order line, clicks Edit Additional Information, then uses it to add the details.
    - Packing instructions
    - Packing material
    - Packing date
2. You will add these attributes to the Fulfillment Line Information extensible flexfield.
3. You edit the Invoice Lines descriptive flexfield so you can display shipping details on the invoice line.
4. You modify the FulfillmentIntegration service mapping to map the extensible flexfield between Order Management and the descriptive flexfield in Accounts Receivable.
5. You create an integration algorithm that specifies the data to communicate.

6. You use the Billing work area to examine the invoice that contains the shipping details.

Summary of the Setup

1. Prepare your setup.
2. Edit extensible flexfield.
3. Edit service mapping for extensible flexfield.
4. Edit descriptive flexfield.
5. Edit service mapping for descriptive flexfield.
6. Create integration algorithm.
7. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Prepare Your Setup

1. Enable the Invoice option in the Enable Custom Payloads for Downstream Integration feature.

   For details, see *Get Started with Integrating Order Management with Other Oracle Applications*.
2. Create a sand box, and activate the Service Mappings tool and the Flexfields tool in the sand box.

   For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Create a spreadsheet that you can use as a memory jogger.

Memory Jogger

Row	Attribute	Value
1	View object	
2	View object attribute for packing instructions on the order line	
3	View object attribute for packing material on the order line	
4	View object attribute for packing date on the order line	
5	SdoNameSpace	
6	View object attribute for packing instructions on the invoice line	
7	View object attribute for packing material on the invoice line	
8	View object attribute for packing date on the invoice line	

You will use this spreadsheet to help you remember some of the values that you need during the procedure. Add values while you do the setup, such as when you create the flexfields. We'll provide you reminders at the appropriate points.

Memory Jogger for This Example

Here's the spreadsheet with the values that you will use in this example.

Row	Attribute	Value
1	View object	FulfillLineEffBShipment__InstructionsprivateVO
2	View object attribute for packing instructions on the order line	orderLinePackingInstructions
3	View object attribute for packing material on the order line	orderLinePackingMaterial
4	View object attribute for packing date on the order line	orderLinePackingDate
5	SdoNameSpace	http://xmlns.oracle.com/apps/flex/financials/receivables/transactions/autoInvoices/TransactionLineDff/ For details, see *Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications*.
6	View object attribute for packing instructions on the invoice line	invoiceLinePackingInstructions
7	View object attribute for packing material on the invoice line	invoiceLinePackingMaterial
8	View object attribute for packing date on the invoice line	invoiceLinePackingDate

Edit Extensible Flexfield

Create the extensible flexfields that your Order Entry Specialist will use to enter shipping details on the order line.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields

2. On the Manage Order Extensible Flexfields page, search for the value.

Attribute	Value
Name	Fulfillment Line Information This is a predefined extensible flexfield that you can use on the order header.

3. In the search results, click **Actions > Edit**.
4. Create the context.
    - On the Edit Extensible Flexfield page, click **Manage Contexts**.
    - On the Manage Contexts page, click **Action > Create**.
    - On the Create Context page, set the values.

Attribute	Value
Name	Pack Ship Instructions
Code	Pack Ship Instructions
API Name	PackShipInstructions
Enabled	Contains a check mark.
Behavior	Single Row

    - In the Context Usages area, click **Actions > Create**, then set the value.

Attribute	Value
Name	Additional Fulfillment Line Information

    - Click **Save**.

5. Create the segment for the packing instructions.
    - In the Context Sensitive Segments area, click **Actions > Create**.
    - On the Create Segment page, set values, then click **Save and Close**.

Attribute	Value
Name	Order Line Packing Instructions
Code	Order Line Packing Instructions
API Name	orderLinePackingInstructions
Data Type	Character
Enabled	Contains a check mark.
Table Column	Select any value that's available.
Value Set	30 Characters
Prompt	Packing Instructions
Display Type	Text Box

6. Create the segment for the packing material.
    - In the Context Sensitive Segments area, click **Actions > Create**.
    - On the Create Segment page, set the values, then click **Save and Close**.

Attribute	Value
Name	Order Line Packing Material
Code	Order Line Packing Material
API Name	orderLinePackingMaterial
Data Type	Character

Attribute	Value
Enabled	Contains a check mark.
Table Column	Select any value that's available.
Value Set	30 Characters
Prompt	Packing Material
Display Type	Text Box

7. Create the segment for the packing date.
    - In the Context Sensitive Segments area, click **Actions > Create**.
    - On the Create Segment page, set the values, then click **Save and Close**.

Attribute	Value
Name	Order Line Packing Date
Code	Order Line Packing Date
API Name	orderLinePackingDate
Enabled	Contains a check mark.
Data Type	Date
Table Column	Select any value that's available.
Value Set	Standard Date
Prompt	Packing Date
Display Type	Date/Time

8. Click Save and Close on the current page and subsequent pages until you're on the Edit Extensible Flexfield page.

9. Deploy your flexfield.
    - Click **Deploy Flexfield**.
    - Sign out of Oracle Applications, then sign back in.
    - Navigate back to the Manage Order Extensible Flexfields page, then search the Name attribute for Fulfillment Line Information.

Identify Your View Object and View Attributes

1. Download and open the extensible flexfield.
    - Click **Actions > Download Flexfield Archive**, wait for processing to finish, then, in the Confirmation dialog, click **Download**.
    - In the dialog that displays, save the file to your preferred location.
    - Use your browser's download feature to navigate to the archive.

      For example, if you're using Firefox, click the **down arrow** on the top banner, then navigate to the 10008_DOO_FULFILL_LINES_ADD_INFO file, such as C:\Users\user_name \Downloads \10008_DOO_FULFILL_LINES_ADD_INFO.zip.
    - Unzip 10008_DOO_FULFILL_LINES_ADD_INFO.zip.
    - Navigate to the folder.

      ```
      C:\Users\user_name \Downloads\10008_DOO_FULFILL_LINES_ADD_INFO\oracle\ oracle\apps\scm\doo
      \processOrder\flex\fulfillLineContextsB\view
      ```

    - Use an XML editor to open the FulfillLineEffBShipment__InstructionsprivateVO.xml file.
2. Identify the name of the view object.
    - Find the Name attribute of the view object, then note its value. It should be on about the 3rd or 4th line of code.

      ```
      <ViewObject
       xmlns="http://xmlns.oracle.com/bc4j"
       Name="FulfillLineEffBShipment__InstructionsprivateVO"
      ```

    - In this example, the value is FulfillLineEffBShipment__InstructionsprivateVO. Add this value to row 1 of the memory jogger.
3. Identify the names of your custom attributes.

Search the File for the Value	Edit Your Memory Jogger
<ViewAttribute Name="orderLinePackingInstructions"	Add orderLinePackingInstructions to row 2 of your memory jogger.
<ViewAttribute Name="orderLinePackingMaterial"	Add orderLinePackingMaterial to row 3 of your memory jogger.
<ViewAttribute Name=" orderLinePackingDate"	Add orderLinePackingDate to row 4 of your memory jogger.

You will use these values in your service mapping.

Edit Service Mapping for Extensible Flexfield

Edit a service mapping so Order Management can send attribute details to Oracle Receivables.

1. Go to the Pricing Administration work area.

   Use order administrator privileges so you can access the integration algorithm and service mappings that you use in the Pricing Administration work area to set up the integration.
2. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
3. Enter your sand box.
4. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
5. Add an entity.
   - On the Edit Service Mappings page, on the Entities tab, click **Actions > Add Row**, then set the values.

Attribute	Value
Entity	PackingInstructions_Custom
Description	Context for an extensible flexfield on the order line. It contains details about packing instructions.

   - In the details area, add your attributes, then click **Save**.

Attribute	Type	Primary Key
FulfillLineId_Custom	Long	Contains a check mark.
OrderLinePackingInstructions_Custom	String	Doesn't contain a check mark.
OrderLinePackingMaterial_Custom	String	Doesn't contain a check mark.
OrderLinePackingDate_Custom	Date time	Doesn't contain a check mark.

Note

- You must use the same Type for each attribute that you specify for the corresponding attribute in the descriptive flexfield. If you don't use the same data type, then your service mapping might not fail when compiling but will fail in your runtime environment. The runtime failure might be difficult to troubleshoot because you might not encounter any errors but your set up might not work as you expected.
- Make sure Allow Null contains a check mark for each attribute that you add.
- Leave other attributes empty.

6. Modify the source.
    - Click **Sources**.
    - In the Sources list, click the **row** that has the value.

Attribute	Value
Source	InvoiceSources

    - In the InvoiceSources Details area, click **View > Columns**, then enable Joined Entity, and Joined Entity Attribute.
    - On the Entity Mappings tab, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	PackingInstructions_Custom Scroll all the way to the end of the list. Pricing Administration adds your custom attributes at the end of the list.
Type	View Object
View Object	FulfillLineEffBShipment__InstructionsprivateVO Get the value from your memory jogger. Recall that this is the name of the view object that you noted earlier in this procedure when you downloaded the flexfield archive.
Query Type	Join
Query Attribute	FulfillLineId
Joined Entity	FulfillLine If you can't pick the value, select another line in the Entity Mappings list, then go back to your PackingInstructions_Custom line.
Joined Entity Attribute	FulfillLineId

- On the Attribute Mappings tab, add your custom attributes, then click **Save**.

Attribute	View Object Attribute
FulfillLineId_Custom	FulfillLineId
OrderLinePackingInstructions_ Custom	OrderLinePackingInstructions This is the value from your memory jogger.
OrderLinePackingMaterial_Custom	OrderLinePackingMaterial This is the value from your memory jogger.
OrderLinePackingDate_Custom	OrderLinePackingDate This is the value from your memory jogger.

7. Map your custom attributes to the service that Order Management uses to communicate with Oracle Receivables.
   - Click **Services**.
   - In the Services list, click the **row** that has the value.

Attribute	Value
Service	InvoiceService

   - In the InvoiceService Details area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	PackingInstructions_Custom
Alias	Leave empty
Read	Enabled
Write	Enabled

   Leave the other attributes at their default values.

- At the bottom of the page, in the PackingInstructions_Custom Entities area, add your custom attributes so Order Management can communicate them through the InvoiceService.

Attribute	Alias
FulfillLineId_Custom	FulfillLineId
OrderLinePackingInstructions_Custom	OrderLinePackingInstructions
OrderLinePackingMaterial_Custom	OrderLinePackingMaterial
OrderLinePackingDate_Custom	OrderLinePackingDate

  Enable the Read attribute and Write attribute for each of the attributes that you add.
- Click **Save and Close**, then click **Done**.

Edit Descriptive Flexfield

Create the descriptive flexfields that Oracle Receivables will use to store the data that it receives from Order Management.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Financials
   - Functional Area: Receivables
   - Task: Manage Receivables Descriptive Flexfields
2. On the Manage Receivables Descriptive Flexfields page, search for the value.

Attribute	Value
Name	Invoice Lines

3. In the search results, click **Actions > Edit**.
4. Create the Global Segment for the packing instructions.
   - In the Global Segments area, click **Actions > Create**.
   - On the Create Segment page, set values.

Attribute	Value
Name	Invoice Line Packing Instructions

Attribute	Value
Code	Invoice Line Packing Instructions
API Name	invoiceLinePackingInstructions
Enabled	Contains a check mark.
Data Type	Character
Table Column	Select any one that's available.
Value Set	120 Characters
Prompt	Packing Instructions
Display Type	Text Box

- Click **Save and Close**.

5. Create the Global Segment for the packing material.
   - In the Global Segments area, click **Actions > Create**.
   - On the Create Segment page, set values.

Attribute	Value
Name	Invoice Line Packing Material
Code	Invoice Line Packing Material
API Name	invoiceLinePackingMaterial
Enabled	Contains a check mark.
Data Type	Character
Table Column	Select any one that's available.

Attribute	Value
Value Set	30 Characters
Prompt	Packing Material
Display Type	Text Box

- Click **Save and Close**.

6. Create the Global Segment for the packing date.
   - In the Global Segments area, click **Actions > Create**.
   - On the Create Segment page, set values.

Attribute	Value
Name	Invoice Line Packing Date
Code	Invoice Line Packing Date
API Name	invoiceLinePackingDate
Enabled	Contains a check mark.
Data Type	Date
Table Column	Select any one that's available.
Value Set	Standard Date
Prompt	Packing Date
Display Type	Date/Time

   - Click **Save and Close**.
7. On the Edit Descriptive Flexfield page, click **Save and Close**.

8. Deploy your flexfield.
    - On the Manage Receivables Descriptive Flexfields page, click **Deploy Flexfield**.
    - Sign out of Oracle Applications, then sign back in.
    - Navigate back to the Manage Receivables Descriptive Flexfields page, then search the Name attribute for Invoice Lines.

Identify Your Name Space and Attributes

1. Download the descriptive flexfield.
    - Click **Actions > Download Flexfield Archive**, wait for processing to finish, then, in the Confirmation dialog, click **Download**.
    - In the dialog that displays, save the file to your preferred location.
    - Use your browser's download feature to navigate to the archive.

      For example, if you're using Firefox, click the **down arrow** on the top banner, then navigate to the navigate to the 222_RA_CUSTOMER_TRX_LINES file, such as C:\Users\user_name\Downloads\222_RA_CUSTOMER_TRX_LINES.zip.
2. Unzip 222_RA_CUSTOMER_TRX_LINES.zip.
3. Navigate to the folder.

   ```
   C:\Users\user_name\Downloads\222_RA_CUSTOMER_TRX_LINES.zip\oracle\apps\flex\financials\receivables
   \transactions\autoInvoices\TransactionLineDff\view
   ```
4. Use an XML editor to open the TransactionLineFLEXVO.xml file.
5. Search for the SdoNameSpace attribute, then copy the value of this attribute to row 5 of the memory jogger.

   For this example, assume SdoNameSpace in your XML contains this value.

   ```
   SdoNameSpace="http://xmlns.oracle.com/apps/flex/financials/receivables/transactions/autoInvoices/
   TransactionLineDff/".
   ```
6. Identify the names of your custom attributes.

Search the File for the Value	Edit Your Memory Jogger
`<ViewAttribute Name="invoiceLinePackingInstru`	Add invoiceLinePackingInstructions to row 6 of your memory jogger.
`<ViewAttribute Name="invoiceLinePackingMateri`	Add invoiceLinePackingMaterial to row 7 of your memory jogger.
`<ViewAttribute Name="invoiceLinePackingDate"`	Add invoiceLinePackingDate to row 8 of your memory jogger.

   You will use these values in your service mapping.

Edit Service Mapping for Descriptive Flexfield

Edit the service mapping so Oracle Receivables can receive the attribute details that Order Management sends.

1. Go to the Pricing Administration work area.

   Use order administrator privileges so you can access the integration algorithm and service mappings that you use in the Pricing Administration work area to set up the integration.
2. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
3. Enter your sand box.
4. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
5. Modify the entity.
   - On the Edit Service Mappings page, on the Entities tab, click the row that contains the value.

Attribute	Value
Entity	TransactionLineDff

   - In the details area, verify that you have the attributes you need. If any are missing, add them now.

Attribute	Type
FulfillLineId	Long
NameSpace	String
XsiType	String
__FLEX_Context	String

   Note
   - Make sure the Primary Key doesn't contain a check mark for any of the attributes you add.
   - Make sure Allow Null contains a check mark for each attribute that you add.
   - Make sure the Referenced Entity attribute for FulfillLineId contains ARInterfaceLine.
   - Leave other attributes empty.
   - In the details area, add your attributes, then click **Save**.

Attribute	Type
InvoiceLinePackingMaterial_Custom	String
InvoiceLinePackingDate_Custom	Date

Attribute	Type
InvoiceLinePackingInstructions_ Custom	String

Note

- Make sure the Primary Key doesn't contain a check mark for any of the attributes you add.
- Make sure Allow Null contains a check mark for each attribute that you add.
- Leave all other attributes empty.

6. Modify the source.
   - Click **Sources**.
   - In the Sources list, click the **row** that has the value.

Attribute	Value
Source	InvoiceSources

   - In the InvoiceSources Details area, on the Entity Mappings tab, click the row that has the value.

Attribute	Value
Entity	TransactionLineDff

   - In the TransactionLineDff Details area, add your attributes.

Attribute	View Object Attribute	Expression
FulfillLineId	FulfillLineId	-
NameSpace	-	`"http://xmlns.oracle.com/apps/flex/financials/receivables/transactions/autoInvoices/TransactionLineDff/"` Note - Get this value from row 5 of your memory jogger. Recall that this is the value of the SdoNameSpace attribute that you copied from the TransactionLineFLEXVO.xml file.

Attribute	View Object Attribute	Expression
		- Enclose the value with double quotation marks.
XsiType	-	`"TransactionLineFLEX"` Enclose the value with double quotation marks.
__FLEX_Context	-	Enter the value of the context code from the XML file you downloaded for your flexfield archive. This example uses GlobalSegment1, so you can leave _FLEX_Context empty. If you didn't include a global segment in this example, you would enter `"ItemPriceForSupplier"`. Enclose the value with double quotation marks.
InvoiceLinePackingMaterial_Custom	-	-
InvoiceLinePackingDate_Custom	-	-
InvoiceLinePackingInstructions_Custom	InvoiceLinePackingInstructions	-

Note

- FulfillLineId should already exist, but if it doesn't, add it.
- The dash (-) means to leave the value empty.
- Capitalization is important. For example, use InvoiceLinePackingInstructions in the View Object Attribute, don't use invoiceLinePackingInstructions.
- Click **Save**.

7. Map your custom attributes to the service that Order Management uses to communicate with Oracle Receivables.
    - Click **Services**.
    - In the Services list, click the **row** that has the value.

Attribute	Value
Service	InvoiceService

Attribute	Value

- In the InvoiceService Details area, click the row that has the value.

Attribute	Value
Entity	TransactionLineDff

- At the bottom of the page, in the TransactionLineDff Entities area, add your custom attributes so Order Management can communicate them through the InvoiceService.

Attribute	Alias
FulfillLineId	FulfillLineId
NameSpace	-
XsiType	-
__FLEX_Context	-
InvoiceLinePackingMaterial_Custom	InvoiceLinePackingMaterial
InvoiceLinePackingDate_Custom	InvoiceLinePackingDate
InvoiceLinePackingInstructions_Custom	InvoiceLinePackingInstructions

Note

- FulfillLineId should already exist, but if it doesn't, add it.
- Enable the Read attribute and Write attribute for each of the attributes that you add except don't enable Write for FulfillLineId.
- Use the memory jogger when you set the Alias for your custom attributes. Its the same value as the API Name attribute from the segment that you created for the descriptive flexfield.

- Click **Save and Close**, then click **Done**.

Create Integration Algorithm

To implement the logic you need, you will modify the integration algorithm from another help topic.

1. Go to the Pricing Administration work area.

   Use order administrator privileges so you can access the integration algorithm and service mappings that you use in the Pricing Administration work area to set up the integration.
2. Click **Tasks**, and then, under Order Management Configuration, click **Manage Algorithms**.
3. On the Manage Algorithms page, click **Actions > Create**.
4. On the Create Algorithm page, set values.

Attribute	Value
Name	Integrate_OM_and_AR Custom You can use any text, except you must suffix the name with a space, then the word Custom.

5. Click **Save**.
6. Click **Variables**, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Name	ARIntegration
Data Type	Data Object
Required	Contains a check mark.
Input/Output	Input and Output
Internal Service Schema	FulfillmentIntegration.InvoiceService

7. Click **Algorithm**.
8. Click **Add Step > Conditional Action**.
9. In the Data Sets area, click **Add Row**, then set the values.

Attribute	Value
Name	FLine
Variable Path	ARIntegration.FulfillLine

Attribute	Value
Primary	Contains a check mark.

10. In the Execute Condition area, click **Add Condition > Default Action**.
11. In the Actions window, add the code, then click **Save**.

```
ARLineDFF.invoiceLinePackingDate=FLineEFF != null ? FLineEFF.orderLinePackingDate: null
ARLineDFF.invoiceLinePackingMaterial=FLineEFF != null ? FLineEFF.orderLinePackingMaterial: null
```

Here's what the code does.

Code	Description
`ARLineDFF.invoiceLinePackingDa = null ? FLineEFF.orderLinePackingDate: null`	If the value of the orderLinePackingDate attribute on the extensible flexfield. - **Contains a value**. Set the value of the invoiceLinePackingDate attribute on the descriptive flexfield to the value that orderLinePackingDate contains. - **Doesn't contain a value**. Set the value of invoiceLinePackingDate to empty.
`ARLineDFF.invoiceLinePackingMa = null ? FLineEFF.orderLinePackingMater null`	If the value of the orderLinePackingMaterial attribute on the extensible flexfield. - **Contains a value**. Set the value of the invoiceLinePackingMaterial attribute on the descriptive flexfield to the value that orderLinePackingMaterial contains. - **Doesn't contain a value**. Set the value of invoiceLinePackingMaterial to empty.

12. Test your algorithm, then click **Save and Close**.
13. Add your integration algorithm to the service mapping.
    - In the Pricing Administration work area, click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
    - On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
    - On the Edit Service Mapping page, click **Services**.
    - In the Services list, modify the row that contains the InvoiceService service.

Service	Implementation Type	Implementation
InvoiceService	Algorithm	Integrate_OM_and_AR Custom This the name of the integration algorithm that you just created.

    - Click **Save and Close**.

Test Your Setup

1. Create a sales order.

- Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

- Add an order line and set the values on the line.

Attribute	Value
Item	AS54888
Quantity	1

- Click the **down arrow** on the order line, then click **View Additional Information**.
- In the Additional Information dialog that displays, click **Ship Instructions**, then verify that the Ship Instructions area contains your Packing Material and Packing Date flexfield attributes.
- Add some test values in your flexfield attributes, then click **Done**.

Attribute	Value
Packing Instructions	Test value for packing instructions.
Packing Material	Test value for packing material.
Packing Date	Test value for packing date.

- Click Submit.

  Assume your order number is 55748.

2. Verify that Order Management correctly sends your data to Accounts Receivable.
    - Make sure you have the privileges that you need to manage Accounts Receivable. For details, see *Privileges That You Need to Implement Order Management*.
    - Go to the Billing work area, then click **Tasks > Manage Transactions**.
      For details, go to *Using Receivables Credit to Cash*, then search for Requirements for Completing a Receivables Transaction.
    - On the Manage Transactions page, search for the transaction.

Attribute	Value
Business Unit	Vision Operations
Transaction Source	Distributed Order Orchestration
Reference	55748

    - On the Review Transaction page, in the Invoice Details area, verify that the invoice contains attributes and attribute values for packing instructions, packing material, and packing date.

Another Example

Assume you set up two items in the Product Information Management work area:

- AS54888
- AS54888-1

These items are identical except the Description attribute is different for the AS54888-1.

You want to use the description for the AS54888 throughout order fulfillment except you want to use the description from AS54888-1 on the invoice in Accounts Receivable.

Here's your solution.

1. Create an extensible flexfield on the fulfillment line and use it to store the description for the AS54888-1 from the Product Information Management work area.
2. Create a descriptive flexfield on the invoice.
3. Use a service mapping to map the extensible flexfield to the descriptive flexfield.

Create an integration algorithm to integrate the extensible flexfield in Order Management with the descriptive flexfield in Oracle Receivables.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Privileges That You Need to Implement Order Management
- Use Integration Algorithms to Implement Complex Logic
- Create a Sandbox So You Can Edit Service Mappings
- Requirements for Completing a Receivables Transaction

Specify Transaction Types When You Integrate Order Management with Accounts Receivable

A sales order might include details you use to determine accounting for a financial transaction, such as an invoice memo, or a debit and credit memo.

You can set up a business rule that sets the value for the AR Transaction Type attribute according to details from the sales order, such as business unit, order type, line type, or item.

Assume you must create an invoice in Oracle Accounts Receivable with a different transaction type for each sales order. You can use AR Transaction Type to specify the accounting entries to use for the transaction.

Customer Name	Project Type	Sales Order Number	AR Transaction Type	Revenue Account
Customer 1	Support	Sales Order 1	Application Support	1111 Income from Application Support
Customer 2	Implementation	Sales Order 2	Oracle Implementation	1112 Income from Implementation

Note

- Accounts Receivable doesn't set the default value for Receivables Transaction.
- Here's the predefined behavior that Order Management uses to set the default value.

Are You Fulfilling or Returning Order Lines?	Transaction Type
Fulfilling	Invoice You can create a pretransformation rule or posttransformation rule to modify this behavior.
Returning	Credit Memo Don't modify this behavior. You must use the Credit Memo transaction type for each return line.

- If you specify AR Transaction Type in a pretransformation rule or posttransformation rule, then write your rule so it uses one of these combinations of attributes.

Attributes	Description
Line category and business unit	Use this combination when you must assign the business unit according to the transaction type. o Use the same AR Transaction Type, such as ARTT-Invoice1, for order lines you're fulfilling. o Use the same AR Transaction Type, such as ARTT-Credit1, for return order lines.
Use one of these combinations.	Use this combination when you must assign the business unit according to transaction types.

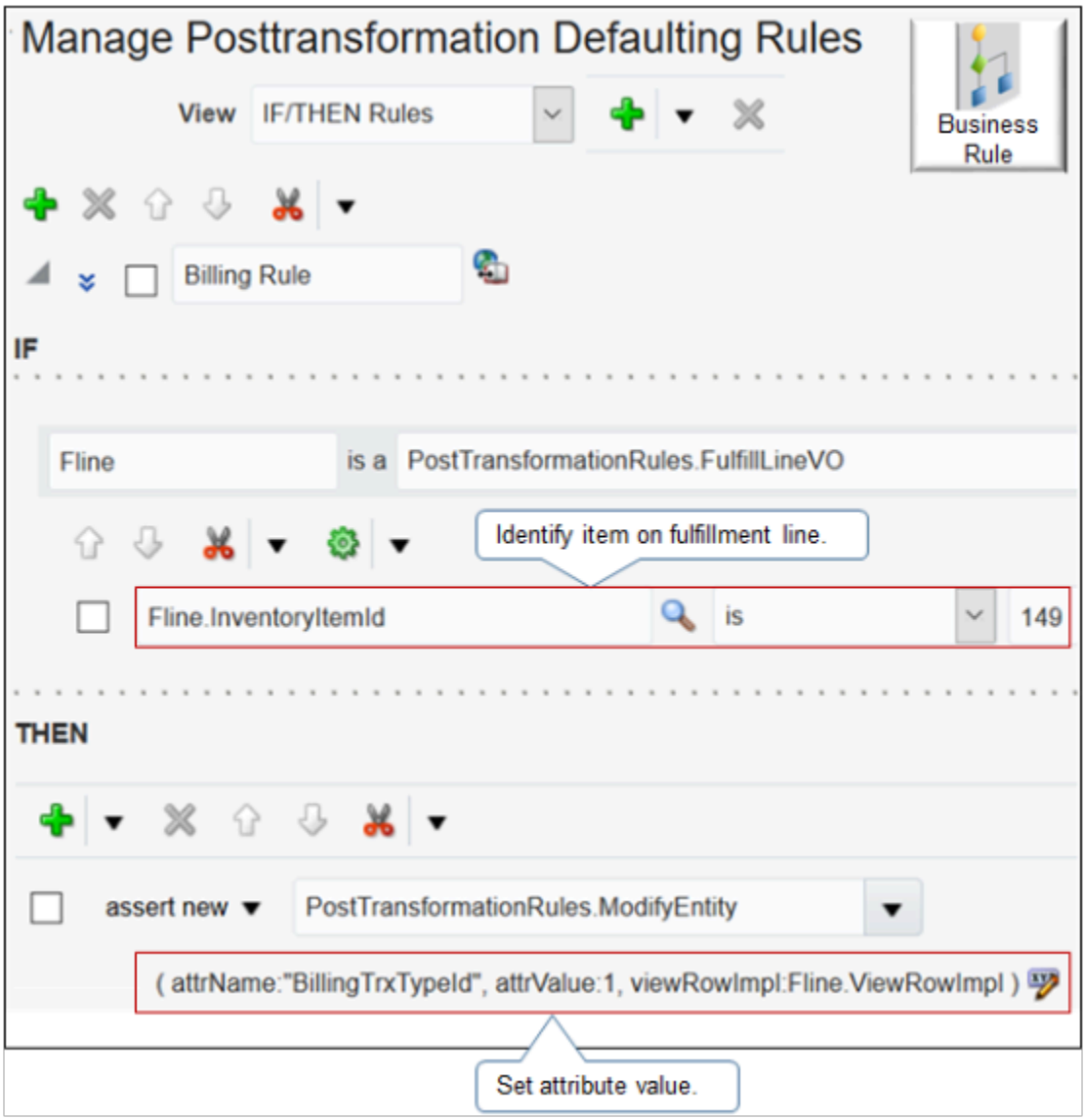
Attributes	Description
○ Line type and line category, and business unit ○ Line type or line category, and business unit	○ Use more than one AR Transaction Type for order lines that you're fulfilling. ○ Use more than one AR Transaction Type for return order lines.
Order category and business unit	Use this combination when your source system sends a source order that includes return order lines but doesn't include order lines that you're fulfilling.
Order type and business unit	Use this combination when a sales order that you create in the Order Management work area includes return order lines but doesn't include order lines that you're fulfilling.

Set Up Transformation Rule

You will create a transformation rule that specifies the transaction type to send from Order Management to Accounts Receivable.

- If the Inventory Item Id attribute on the fulfillment line is 149, then set the Transaction Type attribute to 1.

For example:



For details about how to create a business rule, see *Overview of Using Business Rules With Order Management*.

Try it.

1. Run an SQL to get the value you need for the Billing Transaction Type ID attribute.

```
SELECT distinct LOOKUP_TYPE,
 LOOKUP_CODE
FROM fusion.FND_LOOKUP_VALUES_TL
WHERE LOOKUP_TYPE = 'ORA_DOO_ORDER_TYPES'
order by 1, 2
SELECT RaCustTrxTypesAll.NAME ,
 RaCustTrxTypesAll.CUST_TRX_TYPE_SEQ_ID,
 RaCustTrxTypesAll.SET_ID ,
 SetIdSetPEO.SET_NAME ,
 SetIdSetPEO.SET_ID AS SET_ID1 ,
 RaCustTrxTypesAll.CUST_TRX_TYPE_ID ,
 RaCustTrxTypesAll.END_DATE ,
 RaCustTrxTypesAll.START_DATE ,
 RaCustTrxTypesAll.TYPE
```

```
FROM fusion.ra_cust_trx_types_all RaCustTrxTypesAll,
 fusion.fnd_setid_sets_vl SetIdSetPEO
WHERE RaCustTrxTypesAll.SET_ID = SetIdSetPEO.SET_ID
 AND RaCustTrxTypesAll.Type IN ('INV','CM','DM')
```

Assume the query returns a value of 1 for the Billing Transaction Type ID.

2. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules
3. On the Manage Posttransformation Defaulting Rules page, create a new rule.

Attribute	Value
Name	Billing Rule
Effective Date	Always
Active	Contains a check mark
Advanced Mode	Contains a check mark

4. Create the If statement.

```
Fline is a PostTransformationRules.FulfillLineVO
```

5. Create the And statement.

```
Fline.InventoryItemId is 149
```

6. Create the Then statement, then click **Save and Close**.

```
assert new PostTransformationRules.ModifyEntity (attrName:"BillingTrxTypeId", attrValue:1,
 viewRowImpl:Fline.ViewRowImpl)
```

7. Test your set up.
    - Navigate to the Order Management work area, then create a new sales order.
    - In the Order Lines area, add an item that references the inventory item you specified earlier in this procedure.
    - On the Billing and Payment Details tab, in the Order Line Details area, on the order line you must modify, click **Edit Accounting Details**.
    - In the Edit Accounting Details dialog, set the Receivables Transaction attribute, then click **Submit**.
    - Verify that Accounts Receivable correctly categorized the transaction.

Set Transaction Type Before You Transform

You can run a pretransformation rule only on some events, such as when you select the business unit, customer, or order type, or when you add an order line. For details, see *Transformation Rules*.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Extend Order Headers
- Overview of Transformation Rules

Attributes That You Can Use When You Integrate Order Management with Accounts Receivable

Get details about the attributes that you can specify when you use an integration algorithm to send charges from Oracle Order Management to Oracle Accounts Receivable.

Use Cases

Here are some examples where you might find it useful to modify attributes.

	Need	Solution
1	You have a set of lines that you can and can't ship on the sales order. You need to have the same actual ship date populated for lines that you can ship and for lines that you can't ship so the invoice contains the actual ship date.	Use the FulfillmentIntegration service mapping to modify the ShipDateActual attribute for the line that you can't ship so it has the same value as the ShipDateActual attribute for the line that you can ship. This technique also helps Accounts Receivable to use the same general ledger date and transaction date for the lines that you can and can't ship.
2	You need to send discount lines to Accounts Receivable, and you need to set some of the tax determinants differently on the discount line than they are on the line that isn't discounted.	Use the FulfillmentIntegration service mapping to set the value of the attributes that affect tax determinant differently than the ones that don't affect tax.
3	You need to capture additional details that aren't on the sales order or the fulfillment line, and then display them on the invoice.	Create an extensible flexfield in Order Management to capture the additional details on the sales order, then use the FulfillmentIntegration service mapping to map the extensible flexfield on the sales order to a descriptive flexfield on the invoice in Accounts Receivable.
4	You need to send an attribute value that's available in Order Management but that isn't available in Accounts Receivable.	Use the FulfillmentIntegration service mapping to map the attribute on the sales order to a descriptive flexfield on the invoice in Accounts Receivable.

	Need	Solution
	For example, you have the line number from a purchase order on the sales order or fulfillment line, and you need to display it on the invoice.	
5	You need to capture details for an attribute that isn't available in Order Management but that is available in Accounts Receivable, and then display those details on the invoice. For this example, assume you need the accounting rule duration.	Create an extensible flexfield in Order Management to capture the accounting rule duration on the sales order, then use the FulfillmentIntegration service mapping to map the extensible flexfield on the sales order to the Accounting Rule Duration attribute on the invoice in Accounts Receivable.
6	You need to recognize freight as revenue and send a shipping charge as an invoice line. For example, you have a freight charge where the LineType equals LINE and another line where the LineType equals FREIGHT.	Use the FulfillmentIntegration service mapping to map the line that has the shipping charge record where the LineType equals LINE.
7	Your order line includes recurring billing, and you need to set the first invoice date according to a value that you set on the order line instead of from the sales order date.	Create an extensible flexfield in Order Management to capture the first invoice date on the fulfillment line, then use the FulfillmentIntegration service mapping to map the extensible flexfield on the sales order to the TrxDate attribute and the GlDate attribute on the invoice in Accounts Receivable.
8	Your order line includes recurring billing, and you notice that the invoice sometimes has an extra billing period. This might happen because there's a conversion problem between the Contract Start Date and Contract End Date on the sales order in Order Management, and the Legal Entity Time Zone on the invoice in Accounts Receivable.	Use the FulfillmentIntegration service mapping to set the Contract Start Date and Contract End Date to their current values with a time value of 00:00:00.
9	You have a sales order that includes a kit, and the kit has Included Items that you don't currently send to Accounts Receivable. Order Management stores the delivery number and waybill number only on the included item and not the kit.	Use two descriptive flexfields on the invoice, one for the delivery number and another for the waybill number. Use the FulfillmentIntegration service mapping to send the delivery name and the waybill from the included item to Accounts Receivable, then display them in the descriptive flexfields on the invoice.

	Need	Solution
	You need to display the delivery number and waybill number on the invoice.	
10	You have a sales order that includes a covered item on one line and a coverage item on another line. You need to have the invoice display the actual ship date, not the order date, for these lines. You have observed that the GlDate attribute on the coverage line contains the order date while the GlDate attribute on the covered line contains the actual ship date. This causes Accounts Receivable to split the coverage item and the covered item into separate invoices. Note that the GlDate attribute stores the accounting date of the transaction.	Use the FulfillmentIntegration service mapping to map the actual ship date to the TrxDate attribute for the covered line and for the coverage line. This way, Accounts Receivable won't split the invoice. Alternatively, you can map to the GlDate attribute, or you can map to the order date but not map to the ship date, depending on your needs.

Guidelines

- You must be very careful when you implement logic that changes attribute values that affect applications, such as Oracle Order Management, Oracle Receivables, Oracle Shipping, Oracle Costing, Oracle Tax, and Oracle Revenue Management. You must thoroughly test your modifications and verify that your modifications don't cause data or behavior problems in these applications.
- If you modify an attribute that contains an amount, then you might have a mismatch between the amount that you see on the sales order and the amount on the invoice.
- Be careful when you modify data that might affect the credit memo for a referenced return line. The Import AutoInvoice scheduled process validates invoice details for the original fulfillment line when it creates the credit memo. Make sure you consider how your modification might affect the referenced return line and the return order. For details, see *Update Intercompany Receivables Invoice Import Details*.
- You must test any custom logic that you create that involves these attributes, and fix any problems that you find.

ARInterfaceLine

You can use these attributes on the ARInterfaceLine entity.

Attribute	Description
AccountingRuleDuration	Duration of the rule for the revenue schedule.
AccountingRuleId	Value that uniquely identifies the revenue scheduling rule.
AccountingRuleName	Name of the revenue scheduling rule.
AddressVerificationCode	Abbreviation that identifies the verification for the credit card address from Oracle Payment Server.

Attribute	Description
Amount	Amount on the transaction line.
AmountIncludesTax	Option that indicates whether the line amount includes tax. If its empty, then we use the value from the tax rate code.
ApplicationId	Value that uniquely identifies the application that imports transactions into Oracle Receivables.
ApprovalCode	Abbreviation that identifies the payment approval from the company that issues the credit card.
AssessableValue	The price that a tax authority uses to calculate tax for the item on the line.
AuthorizationComplete	Indicates whether billing for the credit check authorization is done.
AuthorizationNumber	Authorization number from the credit check for the invoice.
BatchSourceName	Name of the source that provides the transaction on the line.
BillContactPartyNumber	Value that uniquely identifies the bill-to contact party for the transaction.
BillCustomerAccountNumber	Value that uniquely identifies the account number of the bill-to customer of the transaction.
BillCustomerSiteNumber	Value that uniquely identifies the bill-to customer site for the transaction.
BillingDate	Billing date to use when you create the invoice. Provide a value in this format: YYYY/MM/DD.
Comments	Comments for the transaction line.
ConsBillingNumber	Number that identifies the consignment for billing and print formatting. Don't use it for a balance forward bill.
ContractedPeriods	Number of recurring invoices for the contract.
ContractEndDate	If the transaction line has a contract, then this attribute includes the contract's end date.
ContractId	Value that uniquely identifies the contract.
ContractLineAmount	If the transaction line has a contract, then this attribute specifies the original item amount on the contract or the new item amount after calculating credit on the contract.
ContractLineId	Value that uniquely identifies a line on the contract.
ContractLineQuantity	If the transaction line has a contract, then this attribute specifies the original quantity for the item on the contract or the new quantity after calculating credits on the contract.
ContractLineUnitPrice	If the transaction line has a contract, then this attribute specifies the original unit price of the item on the contract or the new unit price after calculating credits on the contract.
ContractStartDate	If the transaction line has a contract, then this attribute includes the contract's start date. If you have a recurring invoice, then this attribute contains the start date of the billing period for the first invoice.
ConversionDate	Date that Order Management uses to calculate the currency conversion rate. Order Management uses this value only if the currency on the transaction line isn't the ledger currency. You must use the YYYY/MM/DD format.
ConversionRate	Rate that Order Management uses to calculate the currency conversion. Order management uses this value only if the currency on the transaction line isn't the ledger currency.
ConversionType	Conversion type for the currency on the transaction line.
CreditMethodForAccountRule	The credit method that Order Management uses to credit a transaction when the transaction uses a rule to schedule revenue.

Attribute	Description
	You can set this attribute to: • PRORATE. Credit an equal percentage to all account assignments. • UNIT. Reverse the revenue for the number of units that you specify from an original line of the invoice. If the LineType is TAX or CHARGES, or if the order header has freight charges, then don't use UNIT. • LIFO. (Last In First Out). Removes revenue starting with the most recent general ledger period and reverses all prior periods until it uses up all of the credit memo.
CreditMethodForInstallments	The credit method that Order Management uses to credit a transaction when the transaction has split payment terms. You can set this attribute to: • PRORATE Credit the installments of the credited transaction and prorate them according to the amount that remains for each installment. If the LineType is TAX or CHARGES, or if the order header has freight charges, then don't use PRORATE. • FIFO. (First In First Out). Credit the first installment, first. • LIFO. (Last In First Out). Credit the last installment, first. • Empty. Don't specify a credit method for the installments.
CurrencyCode	Abbreviation that identifies the currency on the transaction line. Use the three character ISO currency code. For example, use USD for US Dollars.
CustomerBankAccountName	Name of the bank account for the bill-to customer.
CustomerTrxTypeName	The transaction type that's on the line.
CustomerTrxTypeSequenceId	Value that uniquely identifies the transaction type.
DefaultTaxationCountry	Default taxation country for tax reporting purposes.
DeferralExclusion	Contains Y or N: • Y. AutoInvoice doesn't include the transaction line from the automated revenue deferral. • N. AutoInvoice does include the line.
Description	Description of the transaction line.
DocumentNumber	Number that identifies the document for the transaction.
DocumentSubType	Type of document. In some countries, a tax or government authority defines and classifies document types for reporting purposes.
EnforceSequenceDateCorrelation	Value that determines whether to enforce chronological sequencing on the document.
ExceptionId	Value that uniquely identifies the tax exception.
ExemptionId	Value that uniquely identifies the tax exemption.
FifthOverrideAmount	A value that you can use to override the fifth amount.
FifthOverridePeriod	Number that identifies the period that has an override amount. For example, the first invoice is for period 1, the second invoice is for period 2, and so on.
FifthOverrideQuantity	A value that you can use to override the fifth quantity.

Attribute	Description
FinalDischargeLocationCode	Abbreviation that identifies the final destination location or the final customer location where Order Management will ship the item.
FinalDischargeLocationId	Value that uniquely identifies the location of the final destination or customer location where Order Management will ship the item.
FirstOverrideAmount	A value that you can use to override the first amount.
FirstOverridePeriod	Number that identifies the period that has an override amount. For example, the first invoice is for period 1, the second invoice is for period 2, and so on.
FirstOverrideQuantity	A value that you can use to override the first quantity.
FirstPtyRegId	Value that uniquely identifies the tax registration for the first party of the transaction. Order Management gets this value from the transaction's legal entity.
FirstPtyRegNumber	Number that identifies the tax registration for the first party of the transaction. Order Management gets this value from the transaction's legal entity.
FOBPoint	The location where the ownership title for the item transfers from the seller to the buyer.
FourthOverrideAmount	A value that you can use to override the fourth amount.
FourthOverridePeriod	Number that identifies the period that has an override amount. For example, the first invoice is for period 1, the second invoice is for period 2, and so on.
FourthOverrideQuantity	A value that you can use to override the fourth quantity.
GlDate	Accounting date of the transaction. This value must reference an accounting period that's open or that you can enter at some point in the future.
IntendedUseClassifId	Value that uniquely identifies the intended use of the item.
InternalNotes	More comments.
InventoryItemId	Value that uniquely identifies the inventory item for the transaction.
InvoicedLineAcctgLevel	Accounting level for the lines that are eligible to create an invoice in the source system.
InvoicingRuleId	Value that uniquely identifies the invoicing rule for the transaction.
InvoicingRuleName	Name of the invoicing rule for the transaction. Use one of these values: • Advance Invoice • Arrears Invoice If you have a rule for an invoice line, then you must enter a value. Otherwise, its optional. You can use the InvoicingRuleId attribute or the InvoicingRuleName attribute, but don't use both of them.
ItemNumber	Number that identifies the inventory item.
LastPeriodToCredit	Last period for the credit. Use this attribute only for a credit memo. You must use a value that's greater than 0 and less than the duration of the accounting rule for the invoice. You must use only a positive integer.

Attribute	Description
LastTrxDebitAuth	Contains Y or N: • Y. The current transaction is the last transaction of a recurrent series of direct debit collections for a debit authorization according to the ISO20022 specification. • N. The current transaction isn't the last transaction.
LegalEntityId	Value that uniquely identifies the legal entity that's responsible for issuing the transaction line.
LineIntendedUse	Classify an item when the intended use of the item affects tax.
LineType	Specify the line type for this transaction. Use one of these values: • LINE. • TA. • FREIGHT. • CHARGES. CHARGES means finance charges. You must enter a value.
MemoLineName	Name of the memo line for this transaction.
MemoLineSequenceId	Value that uniquely identifies the sequence for the memo line for this transaction.
OrgId	Value that uniquely identifies the business unit on the line.
OrigSystemBatchName	Value that identifies the original batch from the source system.
OrigSystemBillAddressId	Value that uniquely identifies the bill-to customer address from the source system.
OrigSystemBillAddressReference	Value that uniquely identifies the reference to the bill-to customer address from the source system.
OrigSystemBillContactId	Value that uniquely identifies the bill-to customer contact from the source system.
OrigSystemBillContactReference	Value that uniquely identifies the reference to the bill-to customer contact from the source system.
OrigSystemBillCustomerId	Value that uniquely identifies the bill-to customer from the source system.
OrigSystemBillCustomerReference	Value that uniquely identifies the reference to the bill-to customer from the source system.
OrigSystemShipAddressId	Value that uniquely identifies the ship-to address from the source system.
OrigSystemShipAddressReference	Value that uniquely identifies the reference for the ship-to address from the source system.
OrigSystemShipContactId	Value that uniquely identifies the ship-to contact from the source system.
OrigSystemShipContactReference	Value that uniquely identifies the reference for ship-to contact from the source system.
OrigSystemShipCustomerId	Value that uniquely identifies the ship-to customer from the source system.
OrigSystemShipCustomerReference	Value that uniquely identifies the reference for the ship-to customer from the source system.
OrigSystemShipPartyId	Value that uniquely identifies the ship-to customer from the source system.
OrigSystemShipPartyReference	Value that uniquely identifies the reference for the ship-to customer from the source system.
OrigSystemShipPartySiteId	Value that uniquely identifies the customer's ship-to address from the source system.
OrigSystemShipPartySiteReference	Value that uniquely identifies the reference for the customer's ship-to site from the source system.
OrigSystemShipPtyContactId	Value that uniquely identifies the customer's ship-to contact from the source system.

Attribute	Description
OrigSystemShipPtyContactReference	Value that uniquely identifies the reference for the contact for the ship-to customer from the source system.
OrigSystemSoldCustomerId	Value that uniquely identifies the sold-to customer from the source system.
OrigSystemSoldCustomerReference	Value that uniquely identifies the reference for the sold-to customer from the source system.
OrigSystemSoldPartyId	Value that uniquely identifies the sold-to customer from the source system.
OrigSystemSoldPartyReference	Value that uniquely identifies the reference for the sold-to customer from the source system.
OverrideAutoAccounting	Contains Y or N: • Y. AutoInvoice uses the deferred revenue account that the interface table provides when it does revenue accounting. • N. AutoInvoice uses the accounts that AutoAccounting creates.
PaymentAttributes	Payment attributes that group transaction lines.
PaymentServerOrderNumber	Number that indicates whether the Oracle Payment Server authorized the credit card payment.
PaymentSetId	Value that uniquely identifies the payment set.
PaymentTermsId	Value that uniquely identifies the payment terms on the line.
PaymentTermsName	Name of the payment term on the line.
PaymentTrxnExtensionId	Value that uniquely identifies a transaction from Oracle Payments that Order Management uses to process payment for a credit card or a bank account.
Periodicity	Period of the recurring invoice. Use one of these values: • DAY • WEEK • MONTH • QUARTER • YEAR If the RecurringBill attribute contains Y, then you must include a value in the Periodicity attribute.
PrimarySalesrepNumber	Number that identifies the primary salesperson.
PrintingOption	Contains Y or N: • Y. You can print the transaction. • N. You can't print it.
ProdFcCategId	Value that uniquely identifies the fiscal classification of the inventory item.
ProductCategory	Specifies that an item is a noninventory item, or for other classifications that you might need for tax purposes.
ProductFiscClassification	Indicates whether tax calculations must include the item tax.
ProductType	Identifies the item's type for tax purposes. Use one of these values:

Attribute	Description
	• Goods • Services
PurchaseOrder	Purchase order number for the transaction.
PurchaseOrderDate	Date of the purchase order.
PurchaseOrderRevision	Revision number for the purchase order.
Quantity	Number of units shipped on the transaction line, or the number of units on the credit memo line.
QuantityOrdered	Original number of units ordered for the transaction.
ReasonCode	Abbreviation that identifies the reason for the transaction.
ReasonCodeMeaning	Meaning of the reason code for the credit memo.
ReceiptMethodId	Value that uniquely identifies the receipt method for the transaction.
ReceiptMethodName	Name of the receipt method for the transaction.
RecurringBill	Option that indicates whether the transaction line is for a recurring invoice. Use one of these values: • Y. • N or empty.
RelatedBatchSourceName	Name of the batch source for the document that references this transaction.
RelatedTrxNumber	Number that identifies the document that references this transaction.
ResetTrxDate	Indicates whether AutoInvoice resets the transaction date to the accounting date when the source doesn't provide the transaction date.
ResourceSalesrepId	Value that uniquely identifies the primary salesperson for this transaction.
RuleEndDate	Date to stop running the revenue scheduling rule for the transaction.
RuleStartDate	Date to start running the revenue scheduling rule for the transaction.
SalesOrder	Sales order number.
SalesOrderDate	Date that Order Management created the sales order or created the return materials authorization. You must use the YYYY/MM/DD format.
SalesOrderLine	Line number from the sales order for the transaction.
SalesOrderRevision	Number of the sales order revision.
SalesOrderSource	Source of the sales order.
SalesTaxId	Value that uniquely identifies the sales tax.
SecondBillingPeriodStartDate	If the first invoice doesn't cover a full billing period, then SecondBillingPeriodStartDate contains the start date of the billing period on the second invoice. SecondBillingPeriodStartDate applies only for a recurring invoice.
SecondInvoiceDate	If the first invoice doesn't cover a full billing period, then SecondInvoiceDate contains the transaction date of the second invoice.

Attribute	Description
SecondOverrideAmount	A value that you can use to override the second amount.
SecondOverridePeriod	Number that identifies the period that has an override amount. For example, the first invoice is for period 1, the second invoice is for period 2, and so on.
SecondOverrideQuantity	A value that you can use to override the second quantity.
SetOfBooksId	Value that uniquely identifies the ledger.
ShipContactPartyNumber	Value that uniquely identifies the contact for the ship-to party for the transaction.
ShipCustomerAccountNumber	Value that uniquely identifies the account for the ship-to customer for the transaction.
ShipCustomerSiteNumber	Value that uniquely identifies the site for the ship-to customer for the transaction.
ShipDateActual	Date that Order Management actually shipped the item.
ShipVia	Identifies the shipping method.
SoldCustomerAccountNumber	Sold-to customer account for the transaction.
SourceApplicationId	Value that uniquely identifies the source application. Used for tax purpose.
SourceEntityCode	Abbreviation that identifies the source entity. Used for tax purpose.
SourceEventClassCode	Abbreviation that identifies the class for the source event. Used for tax purpose.
SourceTrxDetailTaxLineId	Value that uniquely identifies the detail tax line for the source transaction. Used for tax purpose.
SourceTrxId	Value that uniquely identifies the source transaction. Used for tax purpose.
SourceTrxLineId	Value that uniquely identifies the source transaction line. Used for tax purpose.
SourceTrxLineType	Value that identifies the line type for the source transaction. Used for tax purpose.
Tax	Code that represents a charge that a fiscal or tax authority imposes in a tax regime.
Taxable	Contains Y or N: • Y. The current line is taxable. • N or empty. It isn't taxable.
TaxableAmount	Amount to tax.
TaxCode	Abbreviation that identifies the tax classification for the transaction line. If the LineType attribute contains: • LINE, then you must include a value in the TaxCode attribute. • CHARGES or FREIGHT, then leave the TaxCode attribute empty.
TaxExempt	Value of the tax exemption for the transaction line.
TaxExemptNumber	Number that identifies the tax exempt certificate for a line item that's exempt from taxes. If the LineType attribute: • Contains LINE, and if the TaxExemptFlag contains E, then set a value in the TaxExemptNumber attribute.

Attribute	Description
	• Doesn't contain LINE, then leave TaxExemptNumber empty. The value E in TaxExemptFlag means the item is tax exempt.
TaxExemptReasonCode	Abbreviation that identifies the reason why the item is tax exempt.
TaxExemptReasonCodeMeaning	Text description of the code that you specify in the TaxExemptReasonCode attribute.
TaxInvoiceDate	Date on the fiscal document that Order Management creates when it ships the item.
TaxInvoiceNumber	Number on a fiscal document that identifies the tax invoice.
TaxJurisdictionCode	Abbreviation that identifies the geographic area where a government applies tax.
TaxPrecedence	Number that indicates the sequence to use when calculating tax when more than one tax applies to an invoice line.
TaxRate	Tax rate for this tax line. If the LineType attribute contains: • TAX, then you must enter a value in the TaxRate attribute or the Amount attribute. • LINE, CHARGES, or FREIGHT, then don't enter a value in the TaxRate attribute.
TaxRateCode	Abbreviation that identifies the numeric value to use when calculating the amount of tax.
TaxRegimeCode	Abbreviation that identifies the tax regime.
TaxStatusCode	Abbreviation that identifies the tax status.
ThirdOverrideAmount	A value that you can use to override the third amount.
ThirdOverridePeriod	Number that identifies the period that has an override amount. For example, the first invoice is for period 1, the second invoice is for period 2, and so on.
ThirdOverrideQuantity	A value that you can use to override the third quantity.
ThirdPtyRegId	Value that uniquely identifies the tax registration for a third party of the transaction. Order Management gets this value from the transaction's bill-to customer.
ThirdPtyRegNumber	Number that identifies the tax registration for a third party of the transaction. Order Management gets this value from the transaction's bill-to customer.
TranslatedDescription	Translated description of the transaction line.
TrxBusinessCategory	Classifies the transaction for tax purposes.
TrxDate	Transaction date. Provide a value in the YYYY/MM/DD format.
TrxNumber	Number that identifies the transaction.
UnitSellingPrice	Selling price for each in inventory.
UnitStandardPrice	Standard price for each unit in inventory.
UOMCode	Abbreviation that uniquely identifies the unit of measure.
UOMName	Name that identifies the unit of measure.
UserDefinedFiscClass	Classify your own tax requirement when the existing fiscal classification types don't meet your needs.

Attribute	Description
VATTaxId	Value that uniquely identifies a value added tax.
WarehouseCode	Abbreviation that identifies the inventory organization where you ship the item from.
WarehouseId	Number that identifies the Inventory organization where you ship the item from.
WaybillNumber	Number that identifies the waybill.

ARChargeInterfaceLine

You can use all the attributes described above for the ARInterfaceLine entity on the ARChargeInterfaceLine entity.

You can also use these attributes on ARChargeInterfaceLine, but you can't use them on ARInterfaceLine:

Attribute	Description
OrderChargeComponentId	Value that uniquely identifies a charge component.
OrderChargeId	Value that uniquely identifies a charge.

Other Attributes That You Can and Can't Use

You can use these attributes:

- TransactionInterfaceGdf
- TransactionInterfaceHeaderDff
- TransactionLineDff
- TransactionLineInterfaceGdf

You can't use these attributes:

- TransactionInterfaceLineDff
- TransactionInterfaceReferenceDff
- ReferenceLineId
- TransactionInterfaceLinkToDff

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Pricing Algorithms
- Service Mapping

More Stuff

Integrate Recurring Billing Between Order Management and Accounts Receivable

You can integrate recurring billing between Oracle Order Management and Oracle Accounts Receivable for service items and for other items.

Example 1: Bill All Periods Immediately

Assume you want to bill the full amount for the entire duration of a service contract in the first billing period.

- You set up a contract that starts on Jan 1, 2024 and that has a duration of 12 Months, so it lasts for one year.
- You want to date the first invoice as Jan 1, 2024, and you want to invoice the full amount for the entire duration of the contract on that first invoice, which is $1,200 ($100 a month for 12 months).

Here's how it works.

1. You create an order line that has these attributes.

Attribute	Value
Charge Type	Recurring at $100 a month You use one time billing even though the charge is recurring.
Quantity	1
UOM	Ea
Service Duration	1 Year
Service Duration Extended Amount	1200
Contract Start Date	01-JAN-2024
Contract End Date	31-DEC-2024
Billing Frequency	ONE TIME
Number of Periods	1

2. You submit the sales order, and then the Invoice fulfillment task sends these attributes to Accounts Receivable.

Attribute	Value
Quantity	1
UOM	Ea
Billing Frequency	ONE TIME
Number of Periods	1
Amount	1200 The math is: o Service Duration Extended Amount of 1200 divided by 1 Number of Periods equals 1200

For details, see *Fulfillment Tasks*.

3. Accounts Receivable reads the data and finds that there's only one row, sees that Number of Periods equals 1, and then uses that one row to create an invoice for $1200 with an Invoice Date of 01-JAN-2024.
4. Accounts Receivable automatically runs the Import AutoInvoice scheduled process. The scheduled process creates the invoice and sets the invoice date to 01-JAN-2024.
5. Accounts Receivable sends the invoice details to Order Management, such as the invoice number, invoice date, amount, and so on.
6. Order Management displays these details on the fulfillment line, and then sets the line's status to Closed.

Example 2: Bill Each Period Monthly

Now let's say you want to date the first invoice on Jan 1 2024, and then date 11 more invoices on the first day of each subsequent month for the duration of the 12 month service.

1. You create an order line that has these attributes. The bold attribute values indicate what's different from example 1.

Attribute	Value
Charge Type	Recurring at $100 a month
Quantity	1
UOM	Ea
Service Duration	1 Year
Service Duration Extended Amount	1200
Contract Start Date	01-JAN-2024
Contract End Date	31-DEC-2024
Billing Frequency	**MONTH**
Number of Periods	**12**

2. You submit the sales order, then the Invoice task sends these values to Accounts Receivable on a single row.

Attribute	Value
Quantity	1
UOM	Ea
Billing Frequency	**MONTH**
Number of Periods	**12**
Amount	**100** The math is:

Attribute	Value
	○ Service Duration Extended Amount of 1200 divided by 12 Number of Periods equals 100

3. Accounts Receivable reads the data and finds that there's only one row, but sees that Number of Periods equals 12. So it creates 12 rows, one for each billing period, and sets the date on each of these rows. The Contract Start Date happens on the first day of each month, so Accounts Receivable sets the date on each row to the first day of each month, such as 01-JAN-2024, 01-FEB-2024, 01-MAR-2024, and so on.
4. Assume you set up the Import AutoInvoice scheduled process so it runs one time on the first day of each month. Accounts Receivable automatically runs the Import AutoInvoice scheduled process. The scheduled process creates an invoice and sets the invoice date to 01-JAN-2024.
5. Accounts Receivable sends the invoice details to Order Management, including the invoice number, invoice date, and amount.
6. Order Management displays these details on the fulfillment line. If the invoice task is the final task in the orchestration process, then Order Management sets the fulfillment line's status to Closed.
7. Receivables automatically sends an invoice for period 2 on February 1, with an Invoice Date of 01-Feb-2024 and an amount of $100. Receivables repeats this step for period 3 through period 12 on the first day of the month, using the same amount but updating the invoice date for each month.
8. Order Management adds a new row for each invoice in the billing details on the fulfillment line.
    - It does this even if the fulfillment line is closed.
    - It adds a new row on the first day of each month.
    - It displays the invoice on the same fulfillment line. It doesn't create a new line for each invoice.
    - In this example, the fulfillment line won't have all 12 invoices until 01-Dec-2024.

Example 3: Override Recurring Billing

Continuing Example 2, assume you need to override some of the values. Assume you need to bill the first period at $50 and the last period at $150. Here's what you need to do.

1. Set these attributes on the Edit Recurring Billing page:
    - Set Override Period to 1 and Override Amount to $50.
    - Set Override Period to 12 and Override Amount to $150.
2. Don't enter a value in these attributes:
    - Recurring Billing Start Date
    - Recurring Billing End Date
    - Recurring Invoice Start Date

Order Management will do the same fulfillment flow that it does in Example 2 but with these differences:

- The first invoice will have an amount of $50, which is the value that you set for Override Period 1.
- The last invoice will have an amount of $150, which is the value that you set for Override Period 2.

Example 4: Change the Start Date

Continuing Example 3, assume you need to add a start date for your billing period. Assume the contract:

- Starts on 18-JAN-2024
- Ends on 17-JAN-2025

Here's what you need to do.

1. Set the Recurring Billing Start Date to 01-FEB-2024 on the Edit Recurring Billing page.
2. Don't enter a value in these attributes:
    - Recurring Billing End Date
    - Recurring Invoice Start Date

Order Management will do the same fulfillment flow that it does in Example 3 except the scheduled process will send these values to Receivables on the first invoice:

- SecondBillingPeriodStartDate is 01-FEB-2024.
- Date is 18-JAN-2024.
- Billing period is 18-JAN-2024 through 31-JAN-2024.

The billing period on all subsequent invoices will align with the month. For example, the billing period for the second invoice will be 1-FEB-2024 through 28-FEB-2024.

Example 5: End Billing and Give Credit

Now let's revisit Example 2, Bill Each Period Monthly, except assume recurring billing already started and you billed your customer for January and February at $100 for each month.

Your customer calls, wants to cancel the contract and get credit for February, so you need to cancel billing for March through December and give a $100 credit for February.

Here's what you need to do.

1. Create a referenced return. Make sure you reference the recurring billing line from the original order, but don't modify any other details on the line.
2. Set the Order Date on the line to 01-FEB-2024. This is the date that you want to stop future billing and get credit for periods that you already billed. If you're importing the order, then set the Cancellation Effective Date instead. Note that you can use Cancellation Effective Date only through import.
3. Submit your return. Order Management will close the fulfillment line when it sends the line to Receivables.

Receivables will:

- Delete the rows that it hasn't billed yet from the interface table and create a credit according to the dates that it gets from Order Management.
- Create only one credit memo for the entire cancelled part of the contract. It won't create recurring credits even if you need to credit more than one period that you already billed.

Note

- If you don't want to give any credit but instead want to stop billing for all periods that you haven't billed yet, then set the Order Date or the Cancellation Effective Date to the next billing period that you haven't billed. In this example, set it to 01-March-2024.
- To get credit for all billing periods that you already billed, and to stop billing for all periods that you haven't yet billed, set the date to the first billing period. In this example, set it to 01-JAN-2024.

Use a Different Start Date

Order Management sets the date on the first invoice to the order date. You can use a different date. Here's how:

1. Add an extensible flexfield on the order line.

2. Display the extensible flexfield only when the Billing Frequency attribute on the line doesn't contain One Time Billing.
3. Use a service mapping to map the value in the extensible flexfield to the TRX DATE column in the interface table.
4. At run time, enter your date in the extensible flexfield, then click Submit.

For details, see *Overview of Using Extensible Flexfields in Order Management* and *Use a Service Mapping to Integrate Order Management with Other Oracle Applications*.

Override the Period, Amount, or Quantity

- We recommend that you create an order management extension to make sure that the sum of all values for the period, amount, and quantity equals the duration extended amount or the extended amount. This includes values that you do and don't override.
- If you override an amount, then Receivables will use the override and ignore any discount that you send for the period. Receivables will apply your discount only on periods that you don't override.

For details, see *Overview of Creating Order Management Extensions*.

Remove Extra Billing Periods in Receivables

You might see an extra billing period in Receivables. This might happen because Order Management converts the contract dates according to the legal entity's time zone. To avoid this problem, we recommend that you create a service mapping that sends the actual contract dates to Receivables instead of doing a conversion. For details, see *Use a Service Mapping to Integrate Order Management with Other Oracle Applications*.

Include Price, Discounts, and Shipping Charges in Your Payloads

Make sure that the values you send in the payloads you use to integrate with various systems include accurate values for price, discounts, and shipping charges.

Starting in update 20A, Order Management multiplies the unit price by the quantity to determine the total amount, then sends the quantity, unit price, and total amount for each discount or shipping charge of the fulfillment line to Accounts Receivable. Receivables doesn't do any calculation. Instead, it uses these values on the invoice or credit memo. This behavior makes the integration between Order Management and Receivables more consistent for the price, discounts, and shipping charges on the item.

You must make sure the values that you send in the payload that you use to integrate with various systems or applications uses this formula:

```
HeaderCurrencyUnitPrice equals HeaderCurrencyExtendedAmount divided by Quantity
```

If you don't use this formula, then Order Management might send an incorrect amount to Accounts Receivable for billing. This requirement applies during order import or with any fulfillment task you set up in Order Management that populates charge or charge component data, such as a task that converts shipping cost to billable charges.

For example, assume your payload includes a quantity of 2, unit price of $10, and amount of $30 for a discount. Order Management will send the quantity of 2 and $10 unit price to Receivables. Order Management will ignore the $30 amount and instead calculate the amount as quantity 2 multiplied by unit price of $10 equals $20, then send $20 as a discount to Receivables. Receivables bills the amount as $20, not $30. Order Management applies the same logic for a shipping charge.

Note

- If the unit price for the discount or shipping charge isn't available, then Order Management uses the same behavior it uses in updates before 20A.

- If the $10 unit price isn't available for the discount or shipping charge, and if only the $30 amount and quantity of 2 are available, then Order Management sends only the amount and quantity to Receivables. Receivables calculates the unit price as the $30 amount divided by a quantity of 2 equals $15 for the discount.
- If the fulfillment line contains a service duration, then Order Management doesn't send the unit price for the item, discounts, or shipping charges. This is the same behavior across the current and earlier updates.
- You must make sure your payload includes a value for the quantity, unit price, and amount. If you don't include these values, then the amount for the discount or shipping charge between the sales order and billing will be different.

Related Topics

- Overview of Integrating Order Management with Other Oracle Applications

Create One Invoice for Sales Orders with Items That Can and Can't Ship

Set up Order Management so you have a single invoice when the sales order includes order lines that you can ship and lines that you can't ship.

If your sales order has order lines that you can ship and lines that you can't, then Accounts Receivable creates one invoice for each line.

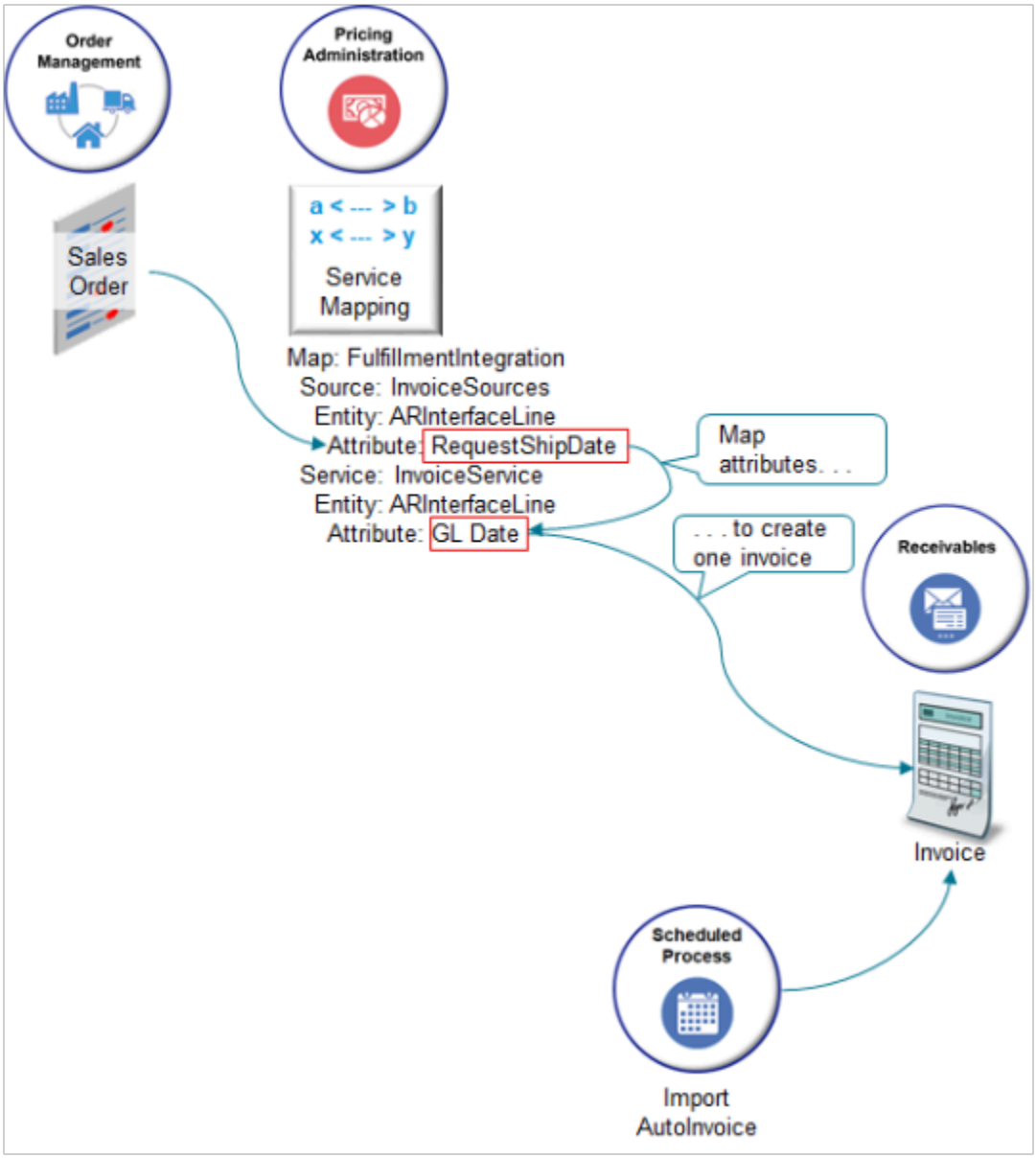
Assume order line 1 contains the AS54888 desktop computer, which you can ship, and line 2 contains the Warranty for AS54888, which you can't ship. Accounts Receivable will create two separate invoices, one for each line, but you only want one invoice because you're selling the computer and the warranty that covers the computer to the same customer on the same sales order.

This problem happens because you can ship the warranty immediately, but it takes more time to ship the computer. So the Ship Date Actual attribute on the order line for the computer is different than the Ship Date Actual attribute on the line for the warranty.

As a result, the GL Date (General Ledger Date) is different, and accounting creates two different invoices, one for each date.

- The warranty doesn't have a fulfillment date or ship date in Order Management or in Oracle Shipping.
- Order Management sets the SHIP_DATE_ACTUAL column in the RA_INTERFACE_LINES_ALL table for the:
    - Computer to the actual ship date.
    - Warranty to the ordered date.
- The Import AutoInvoice scheduled process determines the General Ledger date. If the GL_DATE column:
    - Contains a value, then AutoInvoice uses it.
    - Doesn't contain a value, and if the SHIP_DATE_ACTUAL column contains a value, then AutoInvoice uses SHIP_DATE_ACTUAL.
- The computer and the warranty each have different SHIP_DATE_ACTUAL values, so accounting creates a different invoice for each order line.

Here's how you can fix that.



Note

- Use the Pricing Administration work area to create a service mapping that maps an attribute value from the sales order in Order Management, such as the requested ship date, to the GL Date attribute in Oracle Receivables. You modify the FulfillmentIntegration service mapping. It doesn't affect pricing, but it does use some of the same processing that Pricing uses. That's why its in the Pricing Administration work area.
- Wait until Order Management finishes shipping the shippable item, then run the Import AutoInvoice scheduled process to update the invoice. If you run the process before Order Management finishes shipping the shippable item, then you're still going to get two invoices. So, make sure you wait.

  If you set up Import AutoInvoice to run automatically on a schedule, then make sure there's enough time in the schedule to ship the shippable item. For example, if it takes 13 hours to ship the item, but the schedule runs every 12 hours, then modify the schedule so it run less frequently, such as every 24 hours.

Summary of the Set Up

1. Enable the feature.
2. Set up service mapping.
3. Test your set up.

Enable the Feature

Enable the Invoice option in the Enable Custom Payloads for Downstream Integration feature. For details, see *Get Started with Integrating Order Management with Other Oracle Applications*.

Set Up Service Mapping

Set up the service mapping so it can map the GL Date attribute.

1. Open the mapping.
    - Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
    - Go to the Pricing Administration work area.
    - Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
    - On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
2. Edit the source.
    - On the Edit Service Mapping page, click **Sources**.
    - In the Source List, click the **row** that has InvoiceSources in the Source column.
    - In the InvoiceSources area, on the Entity Mappings tab, click that **row** that has ARInterfaceLine in the Entity column.
    - In the ARInterfaceLine area, on the Attribute Mappings tab, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Attribute	GlDate

Expression	Enter `now()`.
	
	This value sets GlDate to the current date of the Oracle server instead of using the actual ship date. It assumes you're shipping the item on the current data and time.
	
	You can also set the GL Date only for business units that you specify.
	
	`OrgId in [list]?now():null`
	
	where
	
	- *`list`* is a comma-separated list of values that uniquely identifies the business units.
	For example, set GL Date only for business units 204 and 205.
	
	`OrgId in [204,205]?now():null`

Attribute	Value
	use **`fun_all_business_units_v`** to get the values that identify business units. You can also set GL Date to the value of an attribute, and only for a specific business unit. **`OrgId in list?attribute_name():null`** where - *attribute_name* is the name of an attribute that contains the value you will set. For example, set GL Date only for business units 204 and 205, and set it to the requested ship date. **`OrgId in [204,205]?RequestShipDate:null`**
View Object Attribute	To populate your date according to an attribute, leave Expression empty and specify the attribute in View Object Attribute. For example, to set the GL Date to the same value as the requested date, enter **`RequestShipDate`** in View Object Attribute. If you set a value in the Expression, then leave View Object Attribute empty.

3. Edit the service.
    - Click the **Services** tab.
    - In the Service list, click the **row** that has InvoiceService in the Service column.
    - In the InvoiceService area, in the Entity list, click the **row** that has ARInterfaceLine in the Entity column.
    - In the ARInterfaceLine area, click **Actions > Add Row**, set the value, then click **Save and Close**.

Attribute	Value
Attribute	GlDate

For example, query for sales order 56486.

```
select gl_date, trx_date, ship_date_actual, creation_date, inventory_item_id from
 ra_interface_lines_all where interface_line_attribute1 = '56486'
```

For details, see *Use SQL to Query Order Management Data*.

4. Verify that ship_date_actual contains the date value of the Oracle server on both order lines.

Recall that you set GlDate to the current date of the Oracle server when you set up the service mapping.

5. Go to the Scheduled Processes work area, then run the Import AutoInvoice scheduled process. For details, see *Update Intercompany Receivables Invoice Import Details*.

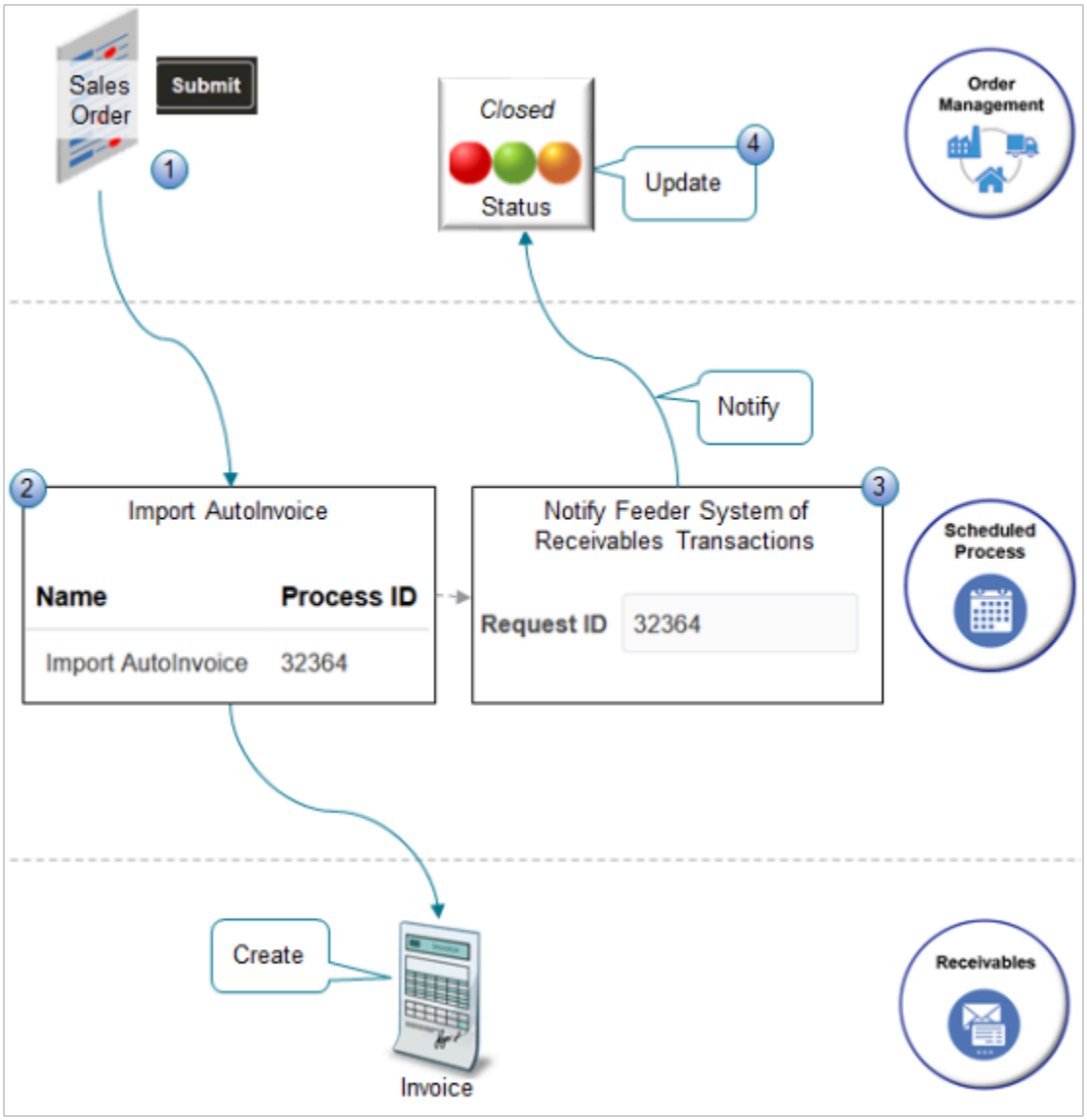
6. Examine the results of running Import AutoInvoice. Verify that the process created a single invoice, and didn't create two invoices.

Related Topics

- Get Started with Integrating Order Management with Other Oracle Applications
- Use SQL to Query Order Management Data
- Create a Sandbox So You Can Edit Service Mappings

Get Things Moving Again in Accounts Receivable

Order Management and Accounts Receivable use predefined scheduled processes to automatically communicate invoice status. You can manually run them to resolve a sales order that's stuck in a particular status.



What the Numbers Mean

1. You create a sales order in Order Management and click Submit. A bunch of fulfillment processing happens, such as scheduling in inventory and shipping the item to the customer.
2. The Import AutoInvoice scheduled process automatically creates an invoice in Accounts Receivable for each sales order, then runs the Notify Feeder System of Receivables Transactions scheduled process. For details, see *Update Intercompany Receivables Invoice Import Details*.
3. The Notify Feeder System of Receivables Transactions scheduled process uses a web service to send details about the invoice and credit memo that Accounts Receivable created to Order Management. In this context, Order Management is the *feeder system*, and Accounts Receivable is notifying Order Management about what its been up to.
4. Order Management uses the details that Notify Feeder System of Receivables Transactions sends to update the fulfillment line status, usually to Closed status.

Here are some notes about using Notify Feeder System of Receivables Transactions.

- You can manually run Notify Feeder System of Receivables Transactions to resolve a sales order that's stuck in a particular status. For example, it might get stuck in the Awaiting Billing status or Billed status, but you expect it to be Closed.
- Make sure you have these privileges. You need them to manage Accounts Receivable when you run the scheduled process:
    - Manage Orchestration Generic Web Service
    - Manage Orchestration Order Billing Interface Web Service
    - Manage Orchestration Order Modification

  If you don't have these privileges, Notify Feeder System of Receivables Transactions will fail with a WSFunctionPermission error.
- if Notify Feeder System of Receivables Transactions finishes successfully, and if lines are still in Awaiting Billing Status, then run the Process Responses from Order Fulfillment scheduled process. For details, see the Process Responses from Order Fulfillment subtopic in *Import and Fulfill Large Volumes of Sales Orders*.

For details about how to set up privileges, see *Security Reference for Order Management*.

Try it.

1. Run an SQL query to identify the Request ID for a specific invoice.

```
SELECT rcta.trx_number ,
rcta.ct_reference,
rctla.Sales_order,
rctla.request_id ,
rctla.line_number
FROM fusion.ra_customer_trx_all rcta ,
fusion.ra_customer_trx_lines_all rctla,
fusion.doo_fulfill_lines_all dfla
WHERE rcta.customer_trx_id = rctla.customer_trx_id
AND rctla.interface_line_attribute5 = TO_CHAR(dfla.fulfill_line_id)
AND
(
dfla.source_order_number = 'ORDER_NUMBER'
OR rctla.INTERFACE_LINE_ATTRIBUTE1 = 'ORDER_NUMBER'
OR rctla.Sales_order = 'ORDER_NUMBER'
)
ORDER BY rcta.customer_trx_id,
```

```
rctla.line_number
```

Replace each instance of ORDER_NUMBER in your query with the order number from Order Management. For example, here's the code you use for order 57485.

```
dfla.source_order_number = '57485'
OR rctla.INTERFACE_LINE_ATTRIBUTE1 = '57485'
OR rctla.Sales_order = '57485'
```

For details about using SQL, see *Use SQL to Query Order Management Data*.

2. Go to the Scheduled Processes work area.
3. Locate the instance of the Import AutoInvoice scheduled process that you must reference.

   Assume you see three instances, but you only need the most recent one that Import AutoInvoice processed for the Vision Operations business unit. Click each instance, expand the Parameters area, then examine the value in the Business Unit attribute. Assume your instance is.

Name	Process ID
Import AutoInvoice	32364

4. Click **Schedule New Process**, then run the Notify Feeder System of Receivables Transactions scheduled process.

Parameter	Value
Request ID	32364 Use the value that your SQL returned earlier in this procedure. Set the Request ID parameter to the Process ID that identifies a specific instance of Import AutoInvoice. If you don't, and if Import AutoInvoice runs more than one time, then Notify Feeder System of Receivables Transactions might not work as expected because it can't identify which instance of Import AutoInvoice to reference. Notify Feeder System of Receivables Transactions might send details about the wrong invoice, or send no details at all.

5. If the problem with the order status persists or the scheduled process doesn't run successfully, wait a few hours then run Notify Feeder System of Receivables Transactions again. Sometimes there's a delay or a change in the fulfillment system that affects the scheduled process.
6. If the problem persists, then run the Process Responses from Order Fulfillment scheduled process. For details, see the Process Responses from Order Fulfillment subtopic in *Import and Fulfill Large Volumes of Sales Orders*.

Related Topics

- Use SQL to Query Order Management Data

Revenue

Overview of Integrating Order Management with Revenue Management

Automate how Revenue Management gets sales order data from Order Management so Revenue Management can recognize revenue for the sales order.

Revenue Management uses this data to create a customer contract and record the performance obligations for the contract. Revenue Management recognizes revenue for the performance obligations when Order Management fulfills the sales order.

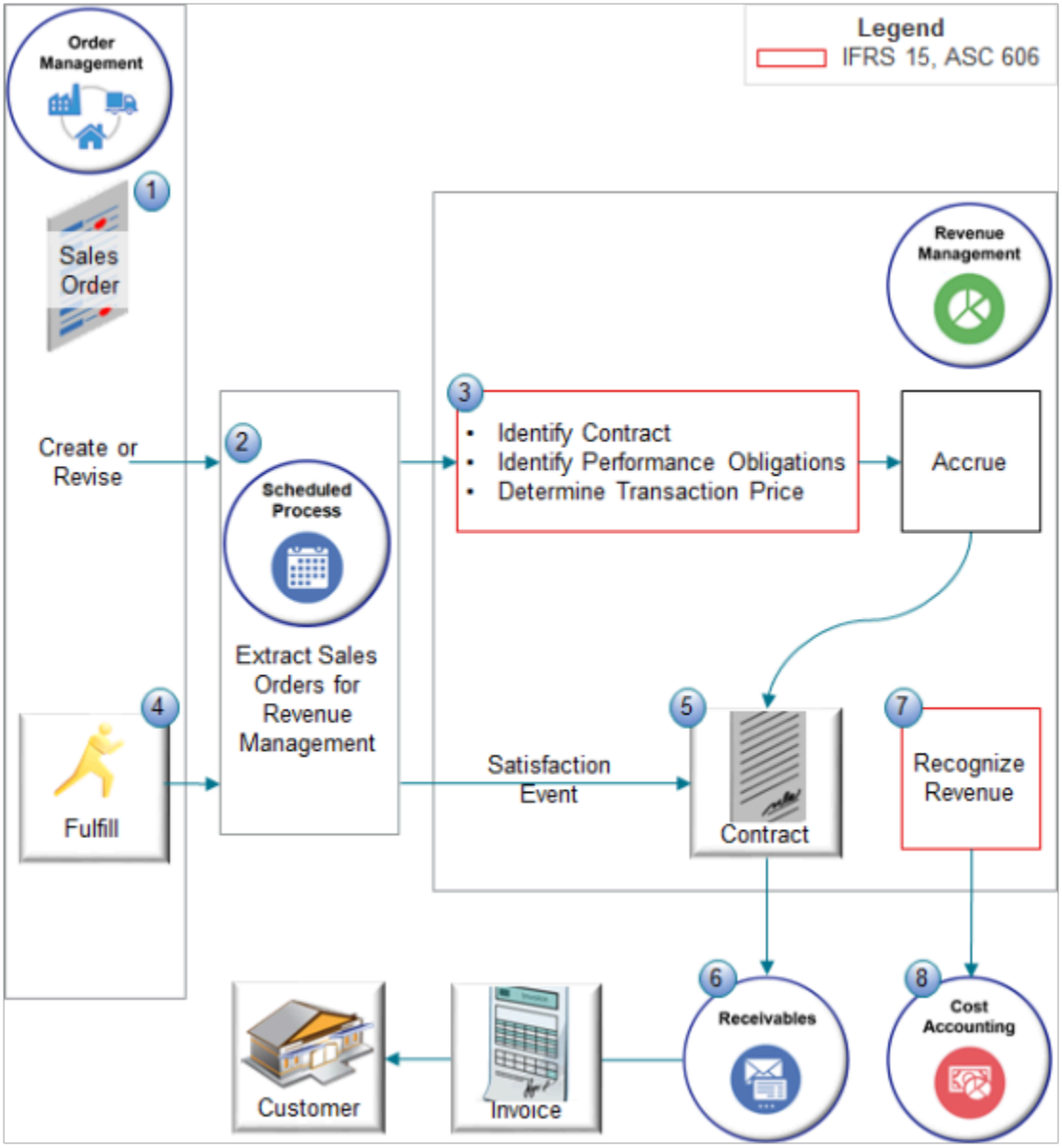
Use this integration to meet IFRS 15 (International Financial Reporting Standard) and ASC 606 (Financial Accounting Standards Board and International Accounting Standards Board) requirements.

- Identify the contracts you have with your customers for your order lines.
- Identify the performance obligations that the contract specifies.
- Determine the transaction price, which is the price you charge your customer to fulfill the order line according to the contract requirements.
- Recognize the revenue that you realize when you meet the performance obligation, such as fulfilling the order line by the requested fulfillment date.

Note

- Revenue Management automatically calculates the transaction price according to the sum of the selling price, and allocates the transaction price to each performance obligation.
- Use this integration to manage change throughout the sales order lifecycle, including revisions to the quantity or price on the order line, return order, and so on. Revenue Management automatically modifies the contract to reflect these changes and reallocates the new price to the performance obligation.

Here's how it works.



Explanation of Numbers

1. An Order Entry Specialist creates a sales order in the Order Management work area, or you import one into Order Management.
2. You use the *Extract Sales Orders for Revenue Management* scheduled process to interface the sales order so Revenue Management can process revenue and Cost Management can process the cost of goods sold.
3. Revenue Management uses rules that you set up to identify accounting contracts and their performance obligations. One accounting contract might reference one or more sales orders from different source systems, or several contracts might reference one sales order that has more than one order line. Each sales order line becomes a promised detail line in Revenue Management.
4. Order Management sends fulfillment details, such as the fulfilled quantity, amount, and delivery date. It sends these details when the line reaches the Ship Confirm status regardless of which step you specify as the

fulfillment completion step. For details about this step, see *Guidelines for Setting Up Orchestration Process Steps*.

Order Management sends fulfillment details for each fulfillment line, such as the fulfilled quantity, amount, and delivery date. It only sends these details when the line reaches the Shipped status for an outbound line or Delivered status for a return line. Order Management uses this behavior regardless of the orchestration process step that you specify as the fulfillment completion step. Order Management uses this behavior even if you set up your own step in the orchestration process and specify it as the fulfillment completion step.

One fulfillment line in Order Management corresponds to one promised detail line in Revenue Management. If Order Management splits a fulfillment line, then Revenue Management represents each of these split lines as one promised detail line.

5. Revenue Management records the event that satisfies the performance obligations, then recognizes revenue.
    - If the item on the fulfillment line is a service item or subscription item, then Revenue Management automatically recognizes revenue for it over the full duration of the service or subscription, according to the satisfaction plan for the item.
    - You can set up the satisfaction plan in Revenue Management and use a revenue scheduling rule to assign it to each fulfillment line.
    - The default satisfaction plan uses the daily rate to calculate revenue.
6. Order Management sends the fulfillment line to Receivables, Receivables creates an invoice to debit the Receivables account and credit the Revenue Clearing account, then sends it to your customer.
7. Revenue Management sends revenue details to Cost Management so Cost Management can recognize the cost of goods sold in the same time frame and in the same proportion that Revenue Management uses to recognize revenue.
8. Cost Management can use these details to analyze gross margin.

Guidelines

- If you revise the price or quantity on an order line, then Revenue Management automatically revises the corresponding accounting contracts to reflect these revisions.
- If you return a sales order, then Revenue Management automatically modifies the contract to reflect the return. You can return part or all of a shipment, reduce the price in the return, or cancel a subscription or service that has recurring billing.

  For details about how to set up Order Management to return a sales order, see *Guidelines for Processing Return Orders*.
- Make sure each service item or subscription item includes a start date and an end date before you run the scheduled process.
- You can invoice an internal material transfer or internal order in Receivables but this integration doesn't integrate them with Revenue Management.
- Revenue Management processes all sales orders except for internal material transfers and transfer orders.
- Make sure you assign the Extraction Start Date system option.
    - The scheduled process processes sales orders that you submit on or after the Extraction Start Date in Revenue Management.
    - The first extract can include sales transactions that you submit up to 90 days before the current date.
    - The extract includes only the primary charge for the item, the profit center business unit on the fulfillment line, and flexfield data.

- If you need to set up an extensible flexfield to allow the Order Entry Specialist to identify contracts or performance obligations, then create and use a context named.
    - Revenue_Management_Information_Header on the order header
    - Revenue_Management_Information_Line on the fulfillment line

Specify the Overshipment

If you set the Quantity to Invoice for Overshipment parameter to Shipped Quantity, then an overshipment will increase the transaction price, and the integration updates the accounting contract according to the fulfilled quantity and the transaction price. For details, see *Manage Order Management Parameters*.

Here's the logic.

If the Shipped Quantity Is	And	Then
Greater than the ordered quantity	You invoice the shipped quantity	Insert a new document line, increment the version and set the quantity so its equal to the shipped quantity. Add a subline that has the quantity for the satisfaction event equal to the shipped quantity.
Greater than the ordered quantity	You invoice the ordered quantity	Add a subline that has the quantity for the satisfaction event equal to the ordered quantity.
Less than the ordered quantity	-	Same as row 1.

For details about these parameters, see *Set Up Shipping Tolerances in Order Management*.

- Default Value for Overshipment Tolerance
- Quantity to Invoice for Overshipment

Example Flows

Here are descriptions of some typical order flows.

Flow	When do we Recognize Revenue?	Description
Ship and invoice one shipment	When Order Management sends fulfillment details	Applies to shippable items, including various configured items, such as assemble-to-order, pick-to-order, kits, and combinations of these configurations.
Ship and invoice a split shipment	When Order Management sends fulfillment details	Applies when Order Management splits a fulfillment line. Some lines are fulfilled, others are pending.

Flow	When do we Recognize Revenue?	Description
Ship and invoice a shipment that uses an overshipment tolerance	When Order Management sends fulfillment details	Revenue Management automatically updates the contract with the adjusted price to reflect the overshipment.
For ship only, ship the item but don't invoice it	When Order Management sends fulfillment details	Use this flow to integrate Revenue Management with your legacy billing system.
For bill only, send the invoice but not the shipment because the item isn't shippable	After you submit the invoice	Order Management sends the fulfilled quantity to Revenue Management after you submit the invoice.
For bill only with a service or subscription, send the invoice but not the shipment because the item is a service or subscription. Include more detail that Revenue Management needs for a service or subscription.	Over time according to the daily rate or a satisfaction plan that you set up	Revenue Management automatically recognizes revenue over the duration of the service or subscription, starting according to the start date and stopping according to the end date on the fulfillment line. You can use the Accounting Rule attribute on the fulfillment line to assign the satisfaction plan on the fulfillment line or you can set up Revenue Management to assign it.

Note

- Each flow identifies contracts and performance obligations when the Order Entry Specialist clicks Submit to submit the sales order to fulfillment.
- Each flow recognizes cost of goods sold immediately after it recognizes revenue, except for the service or subscription flow because cost of goods sold aren't applicable for a service or subscription.

Example Transaction

A transaction in Order Management can include two sales orders that use the same purchase order number to identify them as falling under one contract. Revenue Management groups these orders into one accounting contract, and Cost Management matches cost of goods sold to the revenue that Revenue Management recognizes. For details, see Integrate Revenue Management with Order Management and Supply Chain Cost Management at https://www.oracle.com/webfolder/technetwork/tutorials/tutorial/cloud/r13/wn/om/releases/20C/20C-order-mgmt-wn.htm#F13859.

For other details, see ERP Integration of Revenue Management with Order Management and Supply Chain Cost Management with Demo of End to End Flow on Oracle Cloud Customer Connect at https://cloudcustomerconnect.oracle.com.

Related Topics

- Guidelines for Processing Return Orders
- Guidelines for Setting Up Orchestration Process Steps
- Set Up Shipping Tolerances in Order Management

Integrate Order Management with Revenue Management

Set up system options for Revenue Management, set up a scheduled process, and do other set ups when you integrate Order Management with Revenue Management.

Summary of the Set Up

1. Set up system options for Revenue Management.
2. Do other set ups for Revenue Management.
3. Set up the scheduled process.
4. Set up the lookup code.

In this example, assume you must create an integration for the Vision Operations business unit to integrate the AS54888 Laptop Computer item, using the daily currency exchange rate.

Set Up System Options for Revenue Management

1. Make sure you have the privileges that you need to administer Oracle Financials.
2. In the Setup and Maintenance work area, go to the task.
    - Offering: Financials
    - Functional Area: Revenue Management
    - Task: Manage System Options for Revenue Management
3. In the Source Document Types area, locate the row you will modify. If the row doesn't exist, then create a new one.

Attribute	Value
Source Document Type	Sales Order You can use this integration only with the Sales Order document type.
Ledger	Vision Operations

4. In the Currency Conversion area, set the value.

Attribute	Value
Currency Conversion Type	Daily Exchange Rate If the Order Entry Specialist uses the Edit Currency Details action on the order header to set the Type attribute in the Conversion for Accounting area, then the integration uses that value instead of the value you set in Currency Conversion Type.

5. Click **Extraction Start Date** in the row you just modified, then set the values.

Attribute	Value
Company	01 Operations
Department	130 Computer Resources
Account	2925 Revenue Clearing
Subaccount	2103 California
Product	104 Laptop Computers

6. Assign contract accounts.
7. Review exemption thresholds in the Revenue Accounting and Thresholds area.
8. Assign the IFRS 15 and ASC 606 adoption period.

Do Other Set Ups for Revenue Management

- Set up a pricing dimension structure for a standalone selling price according to the entity's pricing policy for each item on the sales order. For details, go to *Using Revenue Management*, then search for Pricing Dimensions, or search for Standalone Selling Price Profiles.
- Set up rules to group order lines into accounting contracts and distinct performance obligations. For details, go to *Using Revenue Management*, then search for Performance Obligation Identification Rules.

Set Up the Scheduled Process

Run a scheduled process so Oracle Revenue Management has the data it needs to process revenue, to determine cost of goods sold in Oracle Cost Management, and to process invoices in Oracle Receivables.

1. Make sure you set up the Extraction Start Date, described earlier in this topic.
2. Sign into Oracle Applications. Make sure you have the Extract Sales Orders for Revenue Management privilege.
3. Go to the Scheduled Processes work area, then click **Schedule New Process**.
4. Search for the scheduled process.

Attribute	Value
Name	*Extract Sales Orders for Revenue Management* The process sends sales orders, return orders, order revisions, and fulfillment details to Revenue Management.

5. Set the business unit, click **Schedule**, then schedule the process to run at least one time each day.
6. Set the parameters depending on how they impact performance.

Parameter	Description
Batch Size	Specify the number of order lines to process. Specify a minimum of 5,000 lines up to a maximum of 50,000 lines. If you don't specify a value, then the scheduled process will process up to 5,000 lines in each batch, by default.
Number of Processes	Specify the number of instances of this scheduled process to run concurrently. Running processes concurrently helps to finish the batches more quickly, but also consumes more resources. Specify a minimum of 1 instance up to a maximum of 10 instances. If you don't specify a value, then the scheduled process will run up to 4 concurrent instances, by default.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

7. Click **Submit**, then notice the process number that the confirmation dialog displays, such as 122044.
8. Click **Refresh**, then monitor the search results for your process, such as 122044.

   The search results might display more than one instance of your process depending on how you set the parameters. For example, if you set Number of Processes to 6, then it might display up to 6 instances of 122044.

If you encounter an error when you run this scheduled process, there might be a problem with the data in your sales orders. For example, you have an order line that has a service item, such as Monthly Maintenance for the AS54888 Computer, but the line doesn't have the extended amount for the duration of the service in the currency that the order header uses.

Set Up the Lookup Code

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Financials
   - Functional Area: Receivables
   - Task: Manage Receivables Lookups
2. On the Manage Receivables Lookups page, search for the value.

Attribute	Value
Lookup Type	ORA_AR_FEATURES

3. In the Lookup Codes area, click **Actions > New**, then set the values.

Attribute	Value
Lookup Code	VRM_COSTING_INTEGRATION
Reference Data Set	Common Set
Meaning	Enable integration between Order Management and Revenue Management
Display Sequence	1
Enabled	Contains a check mark.
Start Date	Set to today or a date that happens before today.

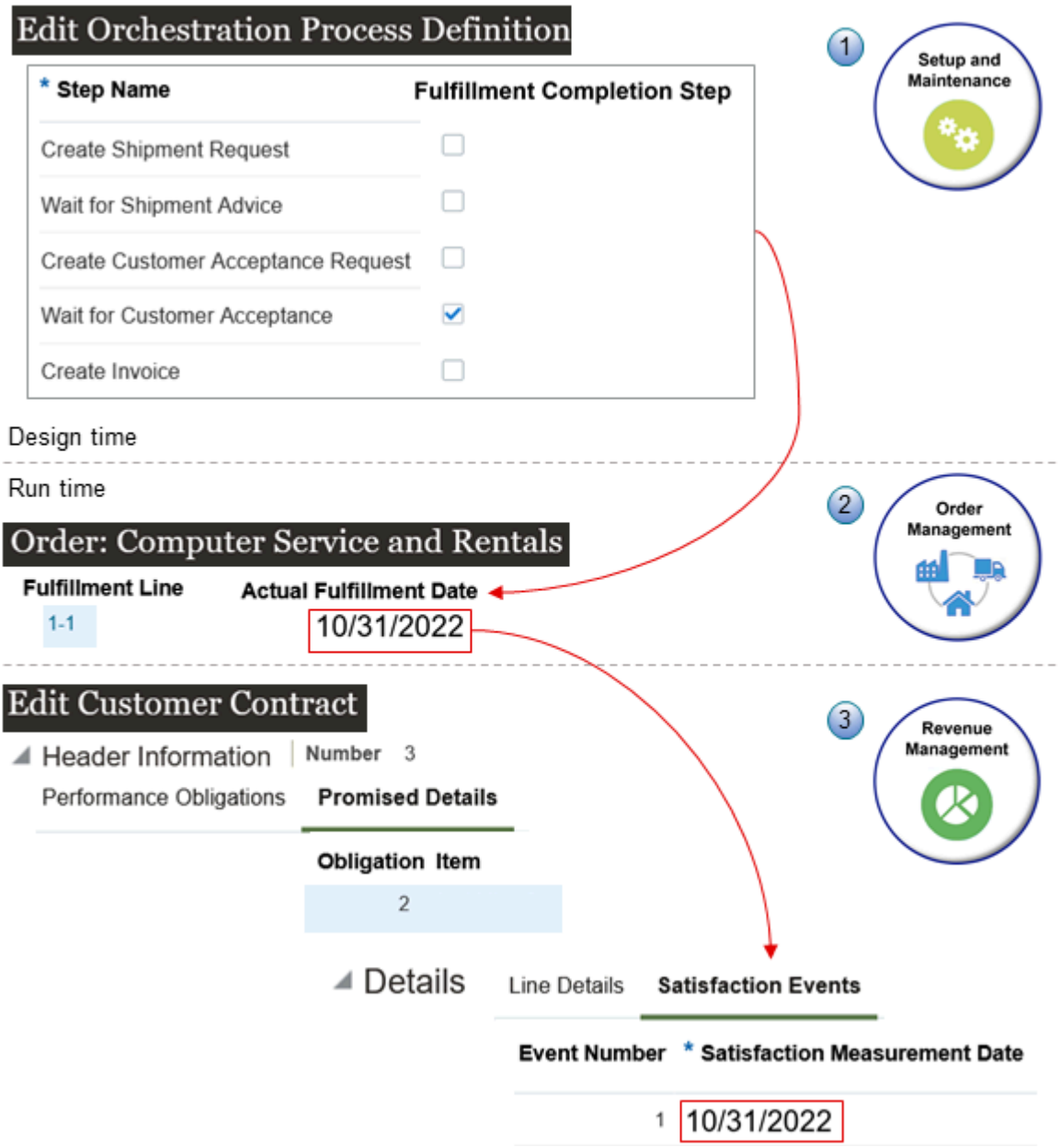
4. Click **Save and Close**.

Related Topics

- Pricing Dimensions
- Standalone Selling Price Profiles
- Performance Obligation Identification Rules

Recognize Revenue on Different Dates

Oracle Order Management comes predefined to send the Satisfaction Measurement Date as the shipment date to Oracle Revenue Management. You can send a different date.



You:

1. Enable the Fulfillment Completion Step option on an orchestration process step to determine what date to send. Where you place the step affects the date.
2. Create a sales order in the Order Management work area, then submit it.
3. Examine the Satisfaction Measurement Date in the Revenue Management work area.

Example

Assume you need to recognize revenue according to the date that your customer accepts the item, you ship the item on 10/26/2022, and your customer accepts the item on 10/31/2022. Here's what you need to do.

1. Create your own custom fulfillment task.
    - Make sure the custom task has a wait step.

- Make sure the Fulfillment Completion Step attribute on the wait step contains a check mark.
- Make sure the custom task isn't a pause task. A pause task doesn't update the fulfillment date on the fulfillment line.

For details, see *Create Your Own Task Type*.

2. Add a new step in the orchestration process that you use to fulfill the item. Add it after the Wait for Shipment Advice step and before the Create Invoice step.

Attribute	Value
Step Name	Create Customer Acceptance Request
Step Type	Service
Fulfillment Completion Step	Not Enabled
Task Type	Reference the type of fulfillment task that you created in step 1.
Task	Reference the fulfillment task that you created in step 1.

3. Add another step immediately after the Create Customer Acceptance Request step.

Attribute	Value
Step Name	Wait for Customer Acceptance
Step Type	Service
Fulfillment Completion Step	Enabled For details about this option, see *Guidelines for Setting Up Orchestration Process Steps*.

Note

- Set up this step so it waits for your customer to accept the shipment.
- Don't place the fulfillment completion step before the shipping step or the receiving step because it might cause data problems between Oracle Order Management and other Oracle applications, such as Oracle Revenue Management, Oracle Cost Management, and Oracle Accounts Receivable.

4. Send the TaskFulfillmentDate in the delayed response. In this example, the TaskFulfimmentDate is the Customer Acceptance Date of 10/31/2022. The orchestration process will send the 10/31/2022 fulfillment date that's on the fulfillment line when it finishes the wait step. You can send the delayed response two minutes after you send the immediate response.
5. Set up the *Extract Sales Orders for Revenue Management* scheduled process so it runs on a schedule. For details, see *Integrate Order Management with Revenue Management*.
6. Create a sales order, add an order line, click **Submit**, then notice the sales order number. Assume it is 54759. The Wait for Customer Acceptance step will set the fulfillment line's Fulfillment Date attribute to 10/31/2022 when the orchestration process fulfills the line.

7. Wait for the Extract Sales Orders for Revenue Management scheduled process to run.
   The scheduled process will send the fulfillment line's Actual Fulfillment Date attribute to Revenue Management.
8. Verify the result.
   - Make sure you have the privileges that you need to manage customer contracts in the Revenue Management work area.
   - Go to the Revenue Management work area.
   - On the Overview page, click **Tasks > Manage Customer Contracts**.
   - On the Manage Customer Contracts page, set the values, then click **Search**.

Attribute	Value
Search Contracts By	Promised Detail
Source Document Type	DOO Sales Order
Source Document	54759

   - In the search results, click the **link** in the Number attribute.
   - On the Edit Customer Contract page, click **Promised Details**, then click **Satisfaction Events**.
   - Verify the value.

Attribute	Value
Satisfaction Measurement Date	10/31/2022

Note

As an option, the response that your custom fulfillment task sends to Order Management can include a date in the TaskFulfillmentDate attribute. For example, you can send the actual delivery date, the current date, the system date, or any other date that you want to record as the fulfillment date. If the TaskFulfillmentDate in the delayed response is empty, then the orchestration process will set the fulfillment date to the wait step's task completion date.

Subscriptions

Overview of Integrating Order Management with Subscription Management

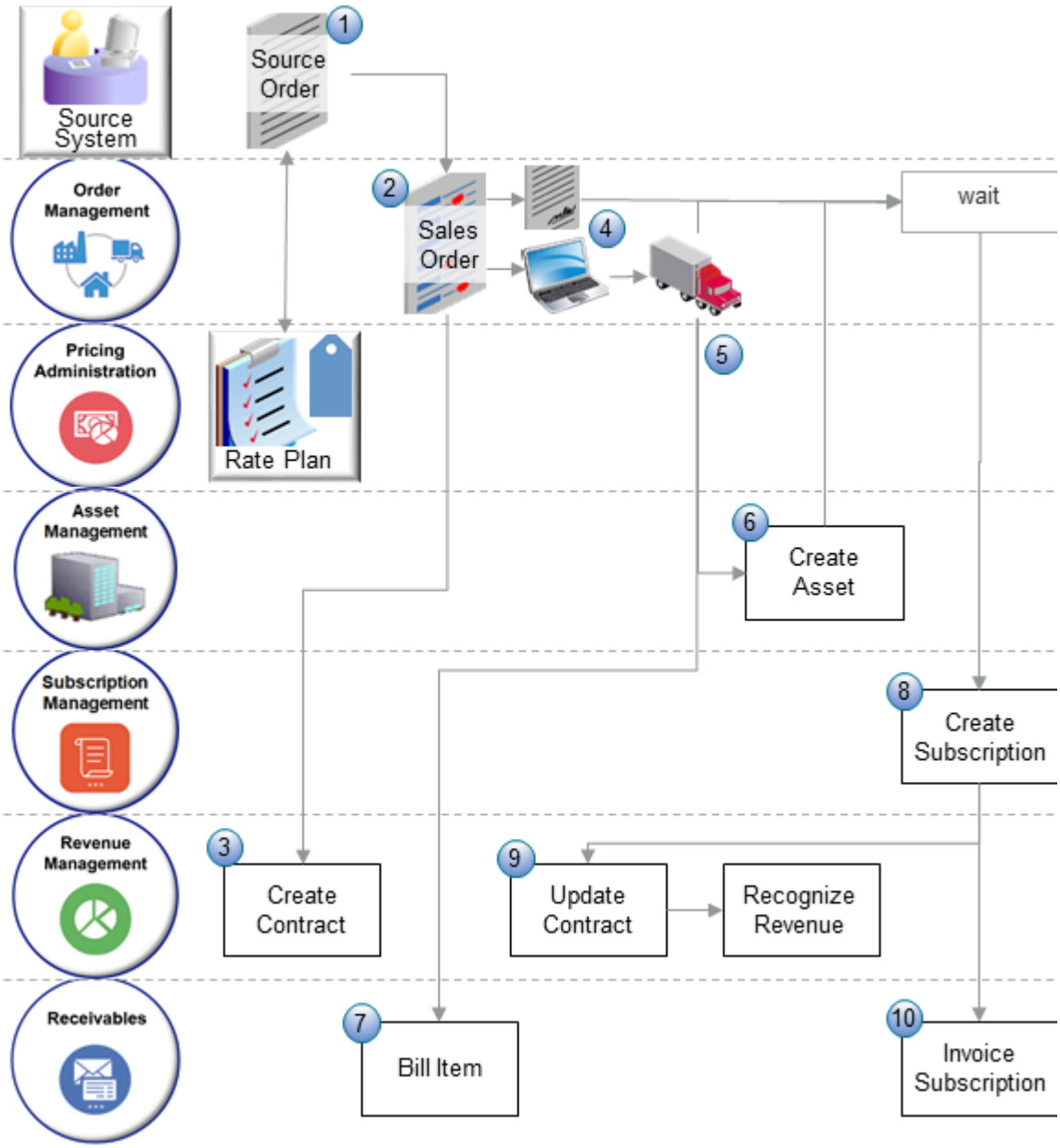
Integrate the order-to-cash flow that you use in Order Management with Subscription Management so you can fulfill subscriptions and coverages in Order Management, then send them to Subscription Management.

Improve how you manage your order-to-cash flow for subscriptions. Fulfill each sales order more quickly when the order includes a physical item and a coverage or subscription, such as an extended warranty, service level agreement, or preventive maintenance.

- Use a single source to manage fulfillment for items, coverages, and subscriptions.
- Create, modify, and end coverages and subscriptions.

- Manage orchestration for your subscription and the item that your subscription covers. For example, sell a subscription for a mobile phone plan and the phone that the plan covers. Orchestrate fulfillment so you don't start the plan until you deliver the phone to your customer.
- Send revenue data to Oracle Revenue Management when you submit the order, and then update this data during fulfillment.
- As an option, you can use a rate plan with your subscription. A rate plan can have one-time charges, recurring charges, usage charges, manual price adjustments, and so on.

You can use Oracle Order Management to orchestrate fulfillment for different order lines across applications, including Oracle Subscription Management, Oracle Supply Chain Management, and Oracle Receivables.



What the Numbers Mean

1. Create a source order or quote in your source system. Interact with Oracle Pricing to get pricing for the source order according to your rate plan. The rate plan can include usage charges, one time charges, and recurring charges for your coverage or subscription.

   In this example, assume your source order has the AS54888 laptop computer, a 2 Year Warranty for the computer, and a 2 Year Subscription for Cloud Storage for the computer. Your sales representative can adjust values and get updated pricing as necessary, then submit the order.
2. Import a source order into Order Management, or create a sales order in the Order Management work area, and then Order Management will use an orchestration process to orchestrate fulfillment, including sending data to your downstream systems. It starts by sending order details to Revenue Management.
3. Revenue Management creates a contract for the item, warranty coverage, and subscription.
4. Order Management continues to orchestrate fulfillment:
   - Shipping ships the laptop as soon as it has supply that's available to ship, then Order Management sends details to Oracle Asset Management and Oracle Accounts Receivable.
   - Shipping doesn't ship the warranty or the subscription because they're not shippable. Instead, Order Management waits until Shipping finishes shipping the computer and Order Management receives a reply from Oracle Assets.
5. Shipping finishes shipping the computer, then Order Management sends details about the order to Asset Management and Accounts Receivable.
6. Asset Management creates an asset for the computer, then sends a reply to Order Management to release the wait.
7. Accounts Receivable bills the computer, but not the coverage or the subscription.
8. Subscription Management creates a subscription for the cloud storage and creates the extended warranty.
9. Revenue Management updates the contract, then recognizes the revenue that you will realize from the contract.
10. Accounts Receivable invoices the subscription that it received from Subscription Management for the cloud storage and the warranty.

Click *here* to get details about how Order Management creates a subscription from the Order Management work area. The flow starts at 02:40 and a demonstration starts at 03:40.

Guidelines

- You can use Configure, Price and Quote as your upstream order capture system to create, amend, or end a subscription. A predefined integration already exists that you can use for your create flow and your end flow.
- You can integrate a coverage or subscription, such as an extended warranty, service level agreement, preventive maintenance, and so on.
- To update or cancel a subscription or coverage, you create a new sales order that references the original coverage or subscription.
- You can revise or cancel the order line in the Order Management work area only if your revisions don't affect pricing or subscription details.
- You can use this feature only with order lines that you create after you opt into the feature. Order Management won't apply this feature to order lines that you create before you opt in.
- You can apply a coverage to the parent or to the child of a configured item. You can apply a subscription to a configured item.

Make sure you have these privileges so you can access this feature in the Order Management work area:

- Initiate Order (FOM_CREATE_ORDER_PRIV)
- Revise Order (FOM_REVISE_ORDER_PRIV)

- Monitor Sales Order (DOO_MONITOR_SALES_ORDER_PRIV)
- Update Order Pricing details (FOM_UPDATE_PRICING_DETAILS_PRIV)

Use REST API

You can use REST API to import subscriptions.

Get some example payload that explain how to import a subscription. Go to *REST API for Oracle Supply Chain Management Cloud*, expand **Sales Orders for Order Hub**, then click **Create Sales Orders**. Scroll down to the example section then have a look. Here are some of the more relevant examples.

Example	Description
33	Create and submit a sales order that has a subscription item.
52	Create a sales order that has a subscription and coverage line that's associated with an item.
53	Amend a sales order that has a subscription and coverage line that's associated with an item.
54	End a subscription and coverage line that's associated with an item.
55	Create a sales order that has a subscription and coverage line that's associated with a KIT or configured item.
57	Create a sales order that has a subscription that has a rate plan.
58	Amend a subscription that has rate plan.
59	End a subscription that has a rate plan.

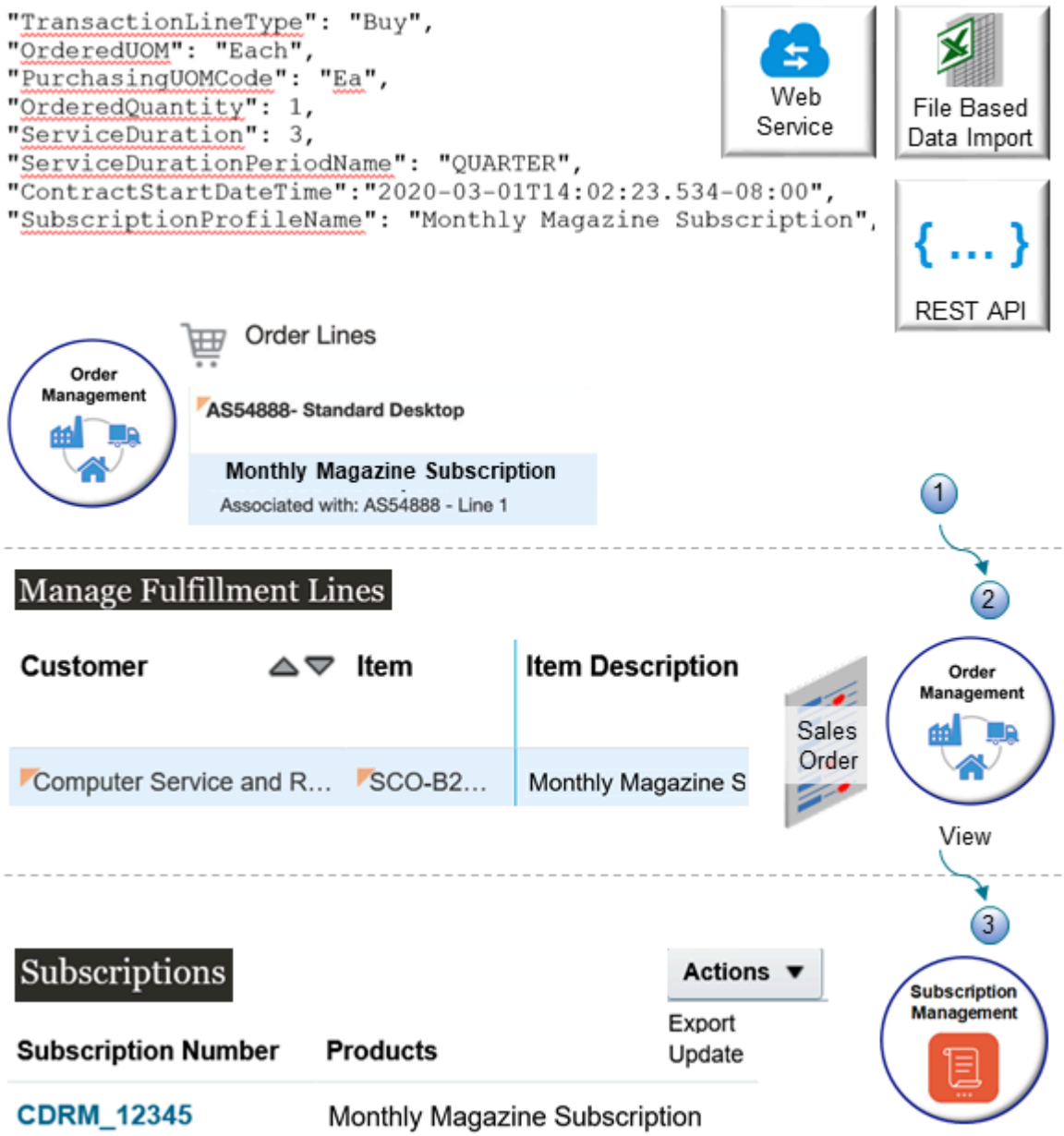
If you import through REST API, then make sure you have these privileges:

- Create Sales Order Requests Using REST Services (FOM_SALES_ORDER_REQUEST_REST_POST_PRIV)
- Create Sales Orders Using REST Services (FOM_SALES_ORDER_REST_POST_PRIV)
- Update Sales Order Requests Using REST Services (FOM_SALES_ORDER_REQUEST_REST_PATCH_PRIV)
- Update Sales Orders Using REST Services (FOM_SALES_ORDER_REST_PATCH_PRIV)

Integrate Order Management with Subscription Management

Integrate Oracle Order Management with Oracle Subscription Management so you can fulfill coverages and subscriptions in Order Management, then send them to Subscription Management.

Here's a summary of your setup.



What the numbers mean:

1. You import the source order into Order Management through REST API, ADFi, or file based data import (FBDI). You can also add a subscription to an order line in the Order Management work area.

   Your import payload must specify all pricing for the item. Although you can use Oracle Pricing to set up pricing for your upstream system, you must include the charges or rate plan as part of your import payload, and your payload must set the Freeze Pricing attribute, Freeze Tax attribute, and Freeze Shipping Charge attribute to Y. For details, see *Freeze Price on Sales Orders*.
2. You can use a fulfillment view in the Order Management work area to view the coverage or subscription after you import it. You can also use the Order Management work area to revise your sales order, but pricing is frozen so don't modify the quantity unless you need to cancel the entire order line.

You can use the Subscription Management work area to export or update the coverage or subscription, but only after Order Management fulfills it.

Summary of the Setup

1. Opt into the feature.
2. Manage administrator profile values.
3. Manage pricing.
4. Set up orchestration.
5. Set up the item.
6. View subscriptions in sales orders.
7. Revise your sales order.

In this example, assume you must Integrate Order Management with Subscription Management so it can process the Monthly Magazine Subscription item, for a period of one year.

Opt Into the Feature

1. Go to the Setup and Maintenance work area, then select the Sales offering.
2. Click **Change Feature Opt In**.
3. On the Opt In page, in the row that has Subscriptions in the Name column, click the **pencil**.
4. Enable the features that you need.
    - Integrate Order Management with Subscription Management to Process Subscriptions
    - Integrate Order Management with Subscription Management to Process Coverages
    - Rate Usage with Events. Enable this feature if you want to use rate plans to send usage charges.

Note

- You can use this feature only with order lines that you create after you opt into the feature.
- Order Management won't apply this feature to order lines that you create before you opt in.

Manage Administrator Profiles

1. Go to the Setup and Maintenance work area, click **Tasks > Search**, then search for Manage Administrator Profile Values.
2. Search for the values.

Attribute	Value
Profile Option Code	RCS_DEFAULT_UOM_CLASS_CODE_FOR_SVC_DURATION
Profile Option Name	SCM Common: Default UOM Class for Service Duration

   You use this profile to specify the UOM class that contains the time measurements for your service duration.
3. In the Profile Values area, click **Actions > New**, then set the values.

Attribute	Value
Profile Level	Site

Attribute	Value
Profile Value	Time

4. Click **Save and Close > Done**.

Map the time units from your class to standard time units.

1. On the Setup page, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Time Unit Mappings
2. Use the Manage Time Unit Mappings page to map time units. Use these values.

User Unit	Base Unit	Base Unit
YEAR	Year	1
MONTH	Month	1
DAY	Day	1
QUARTER	Month	3
WEEK	Day	7
YEAR	Month	12

    These values will accommodate most coverages and subscriptions in a typical Order Management implementation. If necessary, you can use different values. For an example that describes how to map time units, see *Set Up Coverages for Sales Orders*.
3. Click **Save and Close**.
4. Repeat these steps, but do it in the Sales offering.

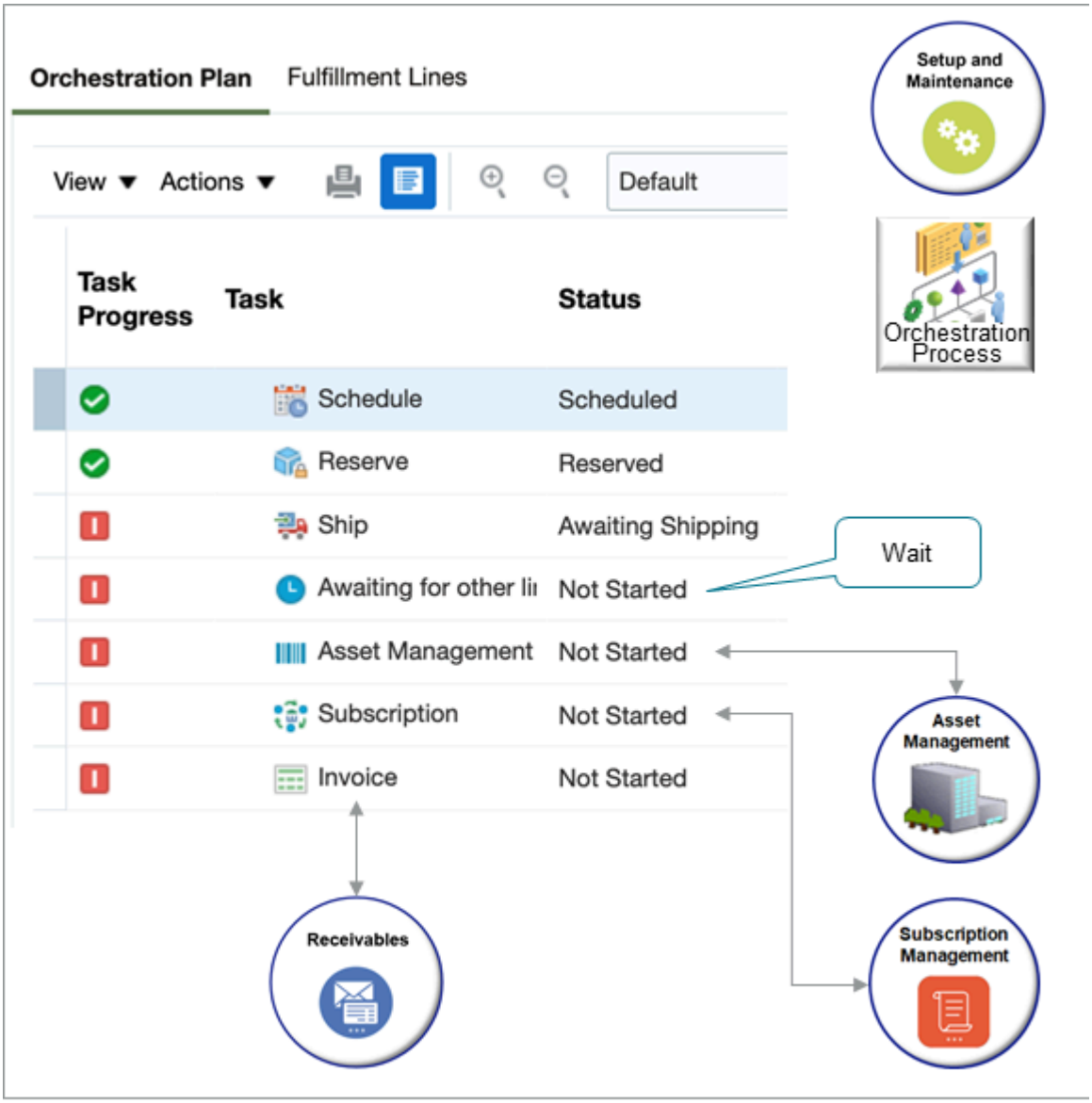
Manage Pricing

Set up pricing rules. For details, see *Pricing Rules*.

Make sure you use the same value in the Price Periodicity UOM Class attribute in your pricing set up that you used when you set up the SCM Common: Default UOM Class for Service Duration profile option earlier in this procedure. For details, see *Manage Pricing Charge Definitions*.

Set Up Orchestration

You can use this predefined orchestration process to orchestrate fulfillment for your subscription across Oracle Applications:



Try it.

1. Go to *Technical Reference for Order Management (Doc ID 2051639.1)*.
2. Download the Integrate Subscriptions attachment.
3. Open the attachment that you just downloaded, locate the SubscriptionIntegrationOrchestrationProcesses_20220615.zip file, then use it to import the CustomDOO_PauseforShipSubscriptionsProcess orchestration process into your environment. For details about how, see the Import subtopic in *Migrate Orchestration Processes in Order Management*.

Your orchestration process must wait, and then send the item to Subscription Management only after Order Management finishes fulfilling it.

If this doesn't work for you for some reason, then you will need to create your own orchestration process. If you are currently using your own orchestration process, then you must modify it. For details, see the Alternative Orchestration Process section in the Integrate Subscriptions attachment in Doc ID 2051639.1.

Set Up the Item

1. Go to the Product Information Management work area, then click **Tasks > Manage Items**.

   Assume you already have a subscription item, and you now just need to verify that you can use it with this feature.
2. On the Manage Items page, search for the value, then open the item for editing.

Attribute	Value
Item	Monthly Magazine Subscription

3. In the Unit of Measure area, set the value.

Attribute	Value
Primary Unit of Measure	Select one: o · For a subscription, select Each. o · For a coverage, chose a service duration. These values come from the class that you specified earlier in this procedure.

4. Click **Specifications > Service**, then set the values.

Attribute	Value
Enable Contract Coverage	If your item is a coverage, then make sure this attribute contains a check mark.
Service Duration Type	Select Variable or Fixed. You can't use Open Ended.
Duration	12
Duration Period	Month
Service Start Type Service Start Delay Allow Suspend Allow Terminate Requires Fulfillment Location	These are optional attributes that you can use, as needed.

Attribute	Value
Requires Item Association	

5. Click **Sales and Order Management**, then set the values.

Attribute	Value
Customer Ordered	Contains a check mark.
Returnable	If you set Allow Terminate to Yes on the Service tab, then you can add a check mark to Returnable.
Sales Product Type	If you item is a: o · Subscription, then set this attribute to Subscription. o · Coverage, then set it to one of: - Included Warranty - Extended Warranty - Service Level Agreement - Software Maintenance - Preventative Maintenance

Manage Subscriptions in Sales Orders

Assume you import sales order 522987, and that order includes a coverage and a subscription for the AS54888 Desktop Computer.

1. Go to the Order Management work area, search for, then open sales order 522987.
2. On the Order page, in the Order Lines area, look at the order lines to get details about the coverage, subscription, and pricing.

Item	Status	Duration	Period	Amount for Total Duration	Quantity
AS54888 Desktop Computer	Asset Interface Completed	-	-	-	1
2 Year Warranty	Subscription Interfaced	2	Year	Subscription Fee	1
Cloud Backup	Subscription Interfaced	2	Year	Activation Fee Subscription Fee	1

3. Scroll to the right to get more details.

Item	Subscription Profile Name	Subscription Number	Pricing Term's Start Date	Pricing Term's Period	Pricing Term's Application Method	Pricing Term's Adjustment Percentage
AS54888 Desktop Computer	-	-	-	-	-	-
2 Year Warranty	zOSS_SP_ ServiceStartActual	CDRM_30015	-	-	-	-
Cloud Backup	zOSS_SP_ ServiceStartActual	CDRM_30015	3/1/23 10:02 PM	1 Quarter	Markup	10

You can also view these details in a fulfillment view. For example:

1. Go to the Order Management work area, then click **Tasks > Manage Fulfillment Lines**.
2. In the search results, click the **order number**.
3. On the Order page, click **Fulfillment Lines**, then notice your lines.

Fulfillment Line	Item
1-1	AS54888 Desktop Computer
1:1-1	2 Year Warranty
1:1-2	Cloud Backup

4. Click **1:1-2**.
5. On the Fulfillment Line page, scroll down to the Service Details section, then examine the attributes.

Revise Your Sales Order

You can use REST API, FBDI, or the Order Management work area to revise a sales order that has a coverage or subscription.

- Pricing is frozen so you must not modify values in attributes that affect pricing in the Order Management work area. If you must modify pricing, then do it in an import payload.
- To use the Order Management work area to cancel a coverage or subscription, enter the entire quantity. Don't enter only part of the quantity. For example, if the original quantity is 10, then enter 10 to cancel.
- You can't copy a sales order that has a coverage or subscription that you integrate with Subscription Management.
- You can use the Order Management work area to associate a subscription with another item, or you can add a subscription by itself and not associate it to another item. See *Add Subscriptions to Sales Orders*.
- You can set other values on the order line that affect the subscription. For example, choose a rate plan, add a pricing term, override a charge, and so on. The values that you can set depend on how you set up pricing. See *Manage Price Lists That Have Rate Plans*.

Create Coverages for Assets

Create an order line that contains a coverage for an asset in the Order Management work area, and then integrate the line with Oracle Subscription Management.

Integrate Oracle Order Management with Oracle Subscription Management so you can use the Order Management to create a coverage that covers an asset that already exists in Oracle Assets, and then integrate the coverage with Subscription Management. Use this integration to help manage your assets.

For example, manage assets when your customer orders several serialized items on an order line, but later calls and wants to cover only one of these serials. For another example, provide a coverage on an asset that you don't sell.

You can specify the asset as the covered item, and you can do this through the Order Management work area, REST API, or FBDI with REST API. This allows you to use a single integration to manage all of your sales orders that involve assets.

You can also use REST API to import an order line that has a coverage, and then use Oracle Pricing to price that line when you create it.

Try it.

1. Set up the integration. See *Integrate Order Management with Subscription Management*.
2. Opt into the Integrate Order Management with Subscription Management to Process Coverages feature.
3. Create a sales order in the Order Management work area.
4. Add a coverage to the sales order.
5. Use the Select Covered Item dialog to add coverage to an asset that already exists in Oracle Asset Management.
    - Select the asset that you want to cover.
    - Set the contract start date and the contract end date.

    You can also set other values on the order line that affect the coverage. For example, add a pricing term, override a charge, and so on. The values that you can set depend on how you set up pricing.

Click *here* for a nifty demonstration. It starts at 7:16.

Create Integrated Coverage Lines in Order Management

Create an order line that contains a coverage in the Order Management work area, and then integrate the line with Oracle Subscription Management.

- Manually adjust the price when you negotiate pricing. Apply your adjustment to the contract's entire duration or only for a specific period of time. For example, give your customer a 10% discount for the first 3 months of a 1 year contract.
- Add pricing terms to the contract anytime during the duration of the contract. For example, to adjust the price by an amount, percent, or to reprice after a specific date that you specify. Order Management will adjust the price and terms, then display the result in the Your Price attribute on the order line as the unit price for the contract's duration, and will display the unit price multiplied by the quantity in the Amount attribute. The price breakdown will also display any discounts you apply and the time period that each discount is in effect.
- Choose a rate plan on the order line.
- End the contract.
- Use REST API to import an order line that has a coverage, and then use Oracle Pricing to price that line when you create it.

Try it.

1. Set up the integration. See *Integrate Order Management with Subscription Management*.
2. Create a sales order in the Order Management work area.
3. Add an item that you can cover to the sales order. For example, add the AS54888 Computer.
4. Add a coverage to the covered item. For example, add a 2 Year Warranty.

You can also set other values on the order line that affect the coverage. For example, add a pricing term, override a charge, and so on. The values that you can set depend on how you set up pricing.

Click *here* for a demonstration. It starts at 05:12 in the presentation.

Guidelines

- You can use this feature only with order lines that you create after you opt into the feature. Order Management won't apply this feature to order lines that you create before you opt in.
- The asset that you cover must already exist in Oracle Asset Management.
- You can cover only one asset on each order line. That asset can be a child asset or a parent asset.
- You can create or end a coverage or subscription in the Order Management work area or through REST API.
- You can renew or amend a coverage or subscription only through REST API.

You can use different orchestration processes to orchestrate fulfillment. To orchestrate fulfillment for:

- The item and a subscription or coverage that covers the item, use DOO_ProductFulfillmentWithIntegratedSubscription
- Only the subscription or to return a subscription, use DOO_IntegratedSubscriptionOnly

As an alternative, create your own orchestration process and add the subscription step. See *Set Up Orchestration Processes for Coverage Items*.

Renew Coverages and Subscriptions in Order Management

Use Order Management to renew your coverages and subscriptions through REST API.

1. Subscription Management creates a draft subscription when the subscription is up for renewal, and then uses a business event to notify the upstream system that it's up for renewal.
2. You can negotiate the renewal in your upstream system, approve it, then use REST API to send it Order Management. Include a reference to the subscription that you're renewing.
3. Use Order Management to send it to Subscription Management as a renewal.
4. Subscription Management will create a new subscription, and that new subscription will have a reference to the old subscription that you just renewed.

1. Set up the integration. See *Integrate Order Management with Subscription Management*.
2. Opt into these features depending on whether you're renewing coverages or subscriptions. You can opt into both features:
    - Integrate Order Management with Subscription Management to Process Coverages
    - Integrate Order Management with Subscription Management to Process Subscriptions
3. Create a REST API payload that references the renewal. For details about how, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management**, then click **Sales Orders for Order Hub**.
4. Submit the payload and create the sales order.

Note

- You can use this feature only with order lines that you create after you opt into the feature.
- Order Management won't apply this feature to order lines that you create before you opt in.

You can use different orchestration processes to send the coverage to Subscription Management. If the covered item is on:

- The same sales order. Use the DOO_ProductFulfillmentWithIntegratedSubscription orchestration process.
- A different sales order. Use the DOO_IntegratedSubscriptionOnly orchestration process.

Use Rate Plans with Your Subscriptions

You can use a rate plan to provide different levels of service for each subscription.

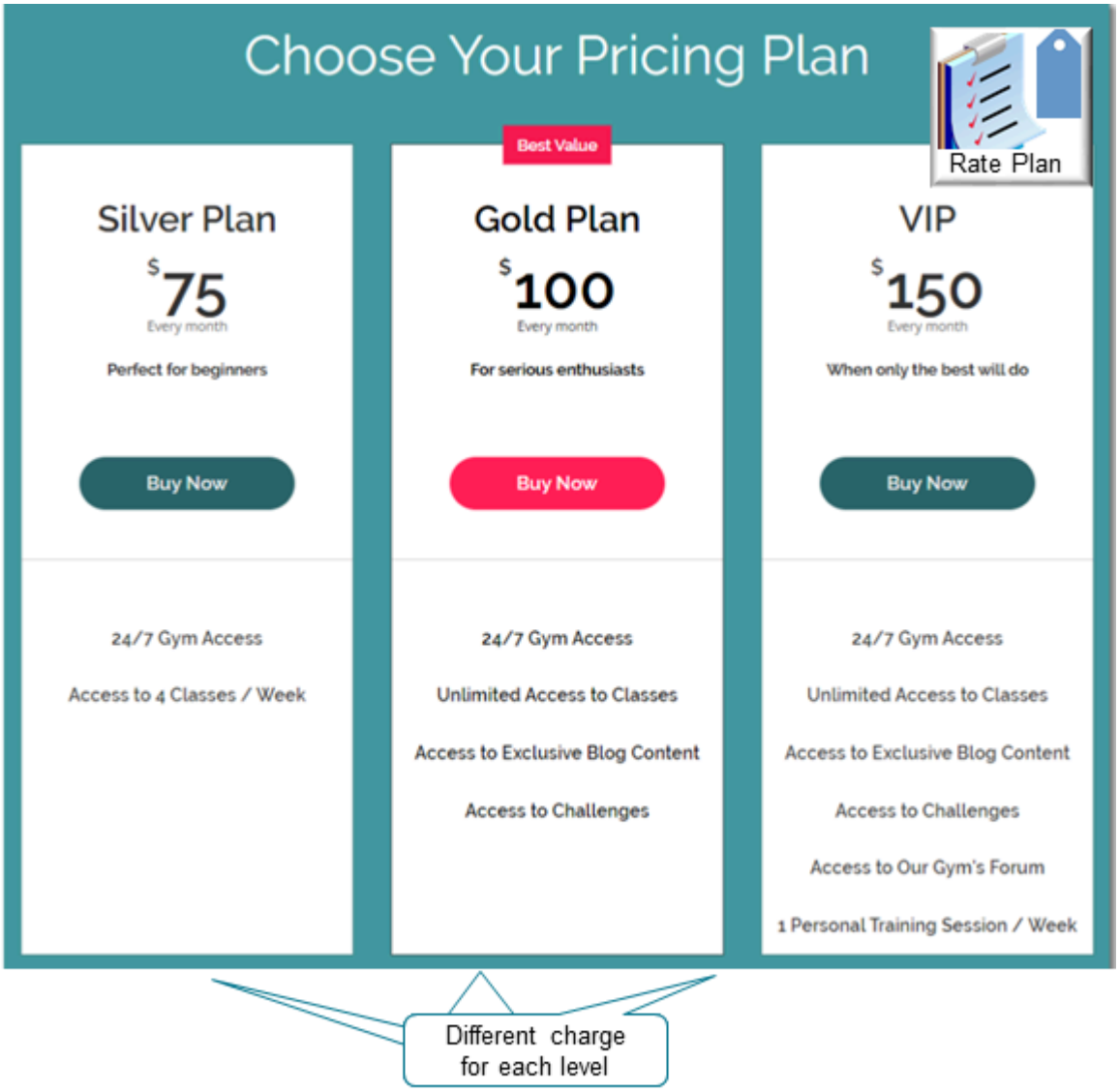
Here are some important concepts:

- **Usage.**Usage of the subscription, such as using a gym, using cloud storage, or using electricity.
- **Usage charge.** A pricing charge that measures usage, such as $2.44 for each gigabyte.
- **Rate plan**. A plan that includes a collection of charges for a subscription item or set of items for each customer. It can include a one-time charge, such as an activation fee, recurring charge such as a monthly fee, or a usage charge that you can apply according to consumption discounts, and so on.

  For example, get a data plan for your cellular phone with a one-time $30 set up fee, with a recurring charge at the rate of $25 for each month for 25 gigabytes of data, and a usage charge of $10 for each 10 gigabytes that you use over the 25 gigabyte limit in each month.

Examples

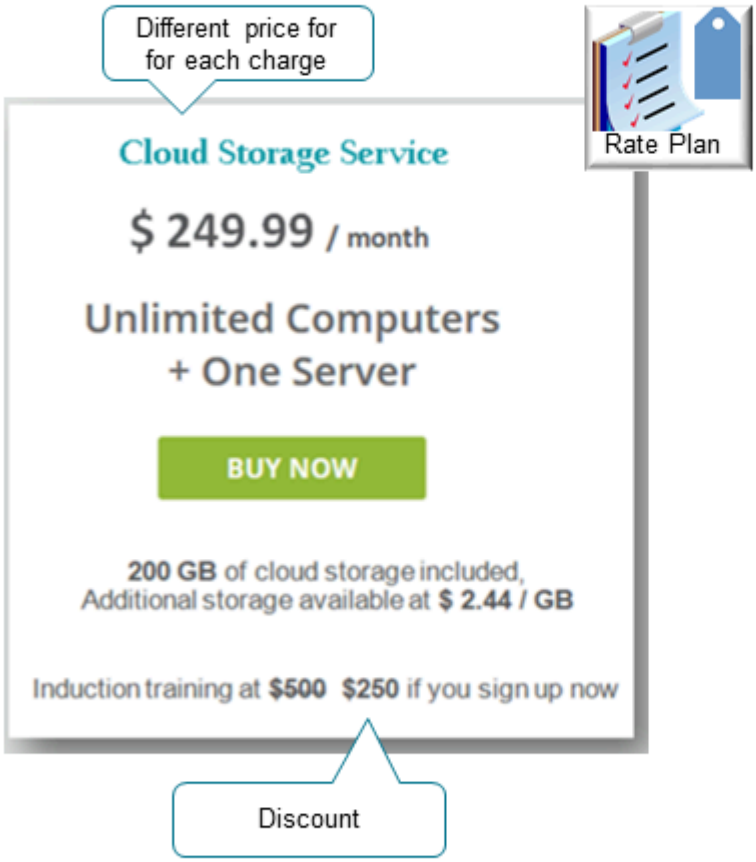
Assume you offer this plan.



Here's a summary of the plan.

Tier	Recurring Charge
Silver Plan	$75.00 for each month
Gold Plan	$100.00 for each month
VIP Plan	$150.00 for each month

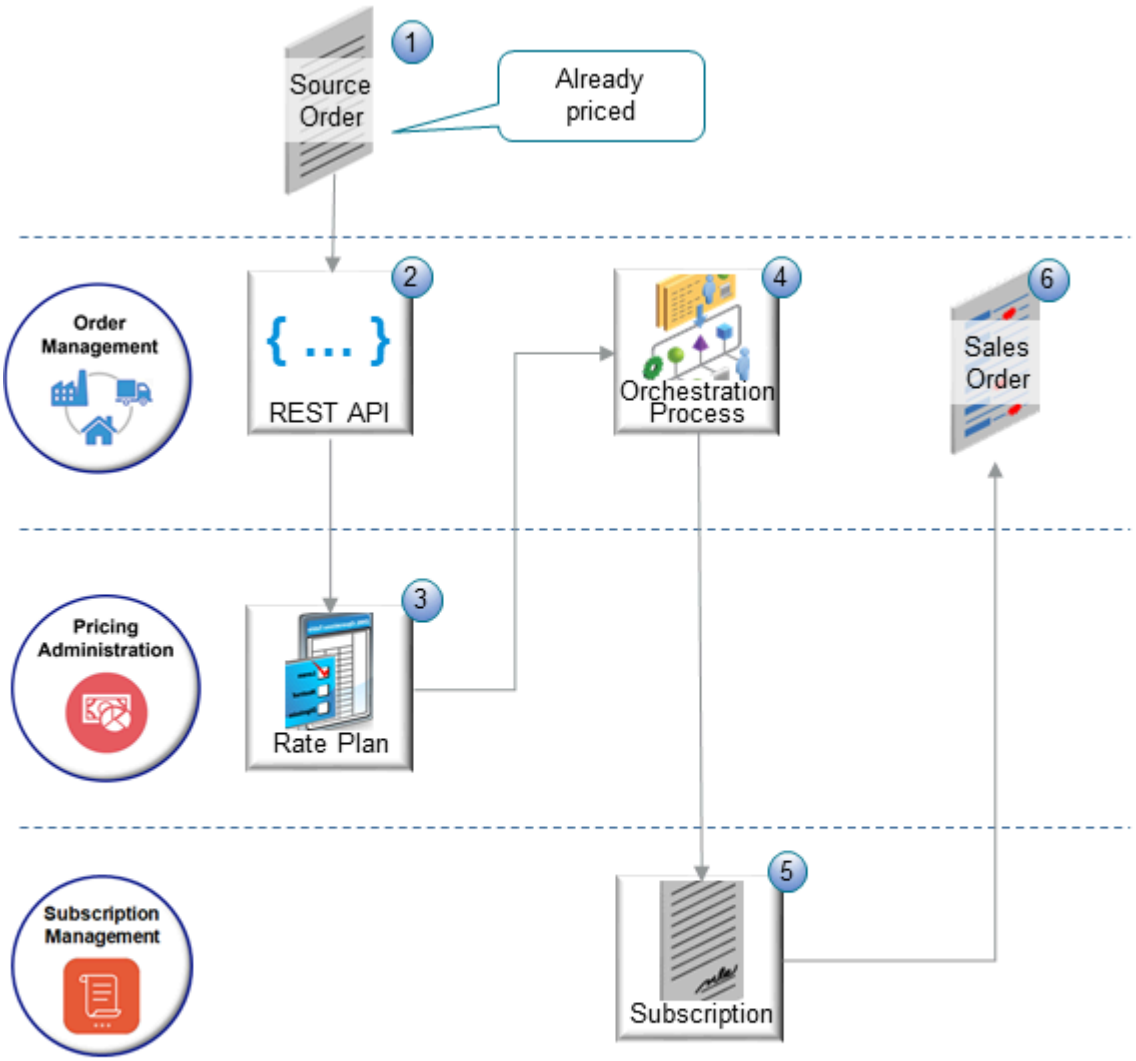
You have a lot of flexibility in how you set up your plan. Here's another example.



Here's a summary of the plan.

Charge	Price	Discount
Training Fee	$500.00	50% off the registration fee if you sign up during October through December.
Service Fee	$249.99 for each month	No service fee for the first month of service.
Consumption Fee	No fee for up to 200 GB. $2.44 for each GB over 200 GB.	-

How it Works



Note

1. You create a source order that has a subscription in your source system. The subscription can include pricing details. As an alternative, you can include pricing details in the Rate Plan REST API.
2. You use REST API to import the source order into Order Management. See the Import Your Source Order subtopic for details.
3. You create a price list that references your rate plan in Oracle Pricing. For details, see *Manage Price Lists That Have Rate Plans*.
4. Order Management uses the orchestration process that you assign to the fulfillment line to orchestrate fulfillment. Order Management fulfills the item, then sends order data, subscription data, and the rate plan's identifier to Subscription Management.

5. Subscription Management activates the subscription. Pricing uses the rate plan and usage amounts to calculate the amount to bill your customer, and sends updated details to display on the sales order to Order Management.
6. Order Management displays the subscription number on the sales order in the Order Management work area.

Note

- If you have usage charges, then you must use a rate plan.
- You can use a rate plan only with a subscription. You can't use a rate plan with a coverage.

Import Your Source Order

Use these guidelines:

- Import your rate plan through the Sales Orders for Order Hub REST API. You must use Sales Orders for Order Hub. You can't use one of Oracle Pricing's REST APIs to import the rate plan.
- Price the sales order before you import it.
- Include data for the charge or for the rate plan on each order line in the import payload. You can Include a one time charge or a recurring charge.
- Include the rate plan identifier and the usage rate's details.

Consider this example:

Type of Charge	Look For
One time	`PriceTypeCode": "ONE_TIME`
Recurring	`"PriceTypeCode": "RECURRING"`
Usage	`"RequestedRatePlanId": "300100573902628"`

Here's the payload.

```
{
 "SourceTransactionNumber": "SUBS_RATEPLAN-0427_01",
 "SourceTransactionSystem": "ORA_BM_CPQ",
 "SourceTransactionId": "SUBS_RATEPLAN-0427_01",
 "BusinessUnitId": 204,
 "BuyingPartyId": 1006,
 "BuyingPartyContactId": 2663,
 "TransactionalCurrencyName": "US Dollar",
 "PartialShipAllowedFlag": false,
 "RequestingBusinessUnitId": 204,
 "RequestingLegalEntityId": 204,
 "FreezePriceFlag": "Y",
 "FreezeTaxFlag": "Y",
 "FreezeShippingChargeFlag": "Y",
 "SubmittedFlag": true,
 "RequestedShipDate": "2023-01-01",
 "TransactionTypeCode": "STD",
 "billToCustomer": [
 {
 "CustomerAccountId": 1006,
 "SiteUseId": 1025,
 "ContactFirstName": "Charles",
 "ContactLastName": "Baker"
 }
 ],
```

```
"shipToCustomer": [
{
"PartyId": 1006,
"SiteId": 1036
}
],
"lines": [
{
"SourceTransactionLineId": "CPQ_LINE_ID_SUBS_1",
"SourceTransactionLineNumber": "CPQ_LINE_NUM_SUBS_1",
"SourceTransactionScheduleId": "CPQ_SCH_ID_SUBS_1",
"SourceScheduleNumber": "CPQ_SCH_NUM_SUBS_1",
"ProductNumber": "Zoom Phone Subscription",
"PaymentTermsCode": 4,
"TransactionLineType": "Buy",
"OrderedUOM": "Each",
"PurchasingUOMCode": "Ea",
"OrderedQuantity": 1,
"ServiceDuration": 2,
"ServiceDurationPeriodName": "YEAR",
"ContractStartDateTime": "2023-01-01",
"ContractEndDateTime": "2024-12-31",
"SubscriptionProfileName": "zOSS_SP_ServiceStartActual_Advance_Month",
"ExternalPriceBookName": "CPQ_Price_List",
"PackingInstructions": "CustomDOO_PauseforShipSubscriptionsProcess",
"externalAssetReference": [
{
"ExternalAssetKey": "SUBS_RATEPLAN_AK_SUBS-0427_01"
}
],
"RequestedRatePlanId": "300100573902628",
"charges": [
{
"SourceChargeId": "C1",
"SequenceNumber": 1,
"CanAdjustFlag": true,
"ChargeDefinitionCode": "QP_SALE_PRICE",
"PriceTypeCode": "ONE_TIME",
"ChargeTypeCode": "ORA_SALE",
"ChargeSubtypeCode": "ORA_PRICE",
"ChargeCurrencyCode": "USD",
"PricedQuantity": 1,
"PricedQuantityUOMCode": "Ea",
"PrimaryFlag": true,
"ApplyToCode": "PRICE",
"RollupFlag": false,
"chargeComponents": [
{
"SourceChargeComponentId": "C1-CC1",
"SequenceNumber": 1,
"PriceElementCode": "QP_LIST_PRICE",
"PriceElementUsageCode": "LIST_PRICE",
"ChargeCurrencyCode": "USD",
"ChargeCurrencyUnitPrice": 100,
"ChargeCurrencyExtendedAmount": 100,
"ChargeCurrencyDurationExtendedAmount": 100,
"HeaderCurrencyCode": "USD",
"HeaderCurrencyUnitPrice": 100,
"HeaderCurrencyExtendedAmount": 100,
"HeaderCurrencyDurationExtendedAmount": 100,
"TaxIncludedFlag": false
},
{
"SourceChargeComponentId": "C1-CC2",
"SequenceNumber": 2,
"PriceElementCode": "QP_NET_PRICE",
```

```
"PriceElementUsageCode": "NET_PRICE",
"ChargeCurrencyCode": "USD",
"ChargeCurrencyUnitPrice": 100,
"ChargeCurrencyExtendedAmount": 100,
"ChargeCurrencyDurationExtendedAmount": 100,
"HeaderCurrencyCode": "USD",
"HeaderCurrencyUnitPrice": 100,
"HeaderCurrencyExtendedAmount": 100,
"HeaderCurrencyDurationExtendedAmount": 100,
"TaxIncludedFlag": false
}
]
},
{
"SourceChargeId": "C2",
"SequenceNumber": 2,
"CanAdjustFlag": true,
"ChargeDefinitionCode": "MONTHLYFEE",
"PriceTypeCode": "RECURRING",
"ChargeTypeCode": "PRICING2",
"ChargeSubtypeCode": "SPMPRICE2",
"ChargeCurrencyCode": "USD",
"PricedQuantity": 1,
"PricedQuantityUOMCode": "Ea",
"PrimaryFlag": false,
"PricePeriodicityCode": "0zG",
"RollupFlag": false,
"ApplyToCode": "PRICE",
"chargeComponents": [
{
"SourceChargeComponentId": "C2-CC1",
"SequenceNumber": 1,
"PriceElementCode": "QP_LIST_PRICE",
"PriceElementUsageCode": "LIST_PRICE",
"ChargeCurrencyUnitPrice": 10,
"ChargeCurrencyExtendedAmount": 10,
"ChargeCurrencyDurationExtendedAmount": 240,
"ChargeCurrencyCode": "USD",
"RollupFlag": false,
"HeaderCurrencyUnitPrice": 10,
"HeaderCurrencyExtendedAmount": 10,
"HeaderCurrencyDurationExtendedAmount": 240,
"HeaderCurrencyCode": "USD",
"TaxIncludedFlag": false
},
{
"SourceChargeComponentId": "C2-CC2",
"SequenceNumber": 2,
"PriceElementCode": "QP_NET_PRICE",
"PriceElementUsageCode": "NET_PRICE",
"ChargeCurrencyUnitPrice": 10,
"ChargeCurrencyExtendedAmount": 10,
"ChargeCurrencyDurationExtendedAmount": 240,
"ChargeCurrencyCode": "USD",
"RollupFlag": false,
"HeaderCurrencyUnitPrice": 10,
"HeaderCurrencyExtendedAmount": 10,
"HeaderCurrencyDurationExtendedAmount": 240,
"HeaderCurrencyCode": "USD",
"TaxIncludedFlag": false
}
]
}
],
"salesCredits": [
{
```

```
"Salesperson": "James Seller",
"Percent": 100,
"SalesCreditTypeId": 1,
"SourceTransactionSalesCreditIdentifier": "HSC01"
}
]
}
]
}
```

For details, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management > Sales Orders for Order Hub Requests. Click Create Sales Orders, then look at example 57.

Import Usage Charge

- If you import a usage charge, then you must also import a rate plan.
- Each usage charge can have a basic charge or a rate table that you specify in a pricing matrix.
- Each usage charge can have tier adjustments and attribute adjustments.

If you use REST API to import a usage charge, then make sure you include the attributes that you use to define the charge in the ratePlanCharges entity, that ratePlanCharges is a child of the ratePlans entity, and ratePlans is a child of the items entity.

For example:

```
"items": [
{
"Item": "Zoom Voice",
"ItemLevelCode": "ITEM",
"LineTypeCode": "ORA_BUY",
"PricingUOM": "Each",
"ratePlans": [
{
"RatePlanName": "Rate_Plan_01012021",
"RatePlanDescription": "Rate Plan for Zoom Voice",
"Currency": "US Dollar",
"StartDate": "2021-01-01",
"EndDate": "2022-12-31",
"ratePlanCharges": [
{
"PricingChargeDefinition": "Call Usage Charge",
"PricingChargeDefinitionCode": "QP_CALL_USAGE_CHARGE",
"CalculationMethodCode": "PRICE",
"UsageUOMCode": "MNS",
"BasePrice": 1.00,
"StartDate": "2021-01-01",
"EndDate": "2021-12-31"
},
```

where

- Attributes in the ratePlanCharges entity define the usage charge, such as `"PricingChargeDefinition": "Call Usage Charge"`
- ratePlanCharges is a child of the ratePlans entity
- ratePlans is a child of the items entity

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Document Prices**.

eBusiness Suite

Overview of Integrating Order Management with eBusiness Suite

If you use an application that's part of Oracle E-Business Suite, Enterprise Resource Planning (Oracle EBS ERP), and plan to migrate to Order Management in phases, then you can integrate Order Management with E-Business Suite and continue to use E-Business Suite.

You deploy a connector that sits between Order Management and the Integrated SOA Gateway in E-Business Suite. Use it to send sales orders and communicate status updates between Order Management and E-Business Suite. This connector uses a SOA (Service Oriented Architecture) and BPEL (Business Process Execution Language) composite that you can use to deploy to a SOA server.

- Set up Order Management to send sales orders to eBusiness Suite so eBusiness Suite can fulfill the sales order, including shipment and invoice.
- Set up eBusiness Suite to send status updates to Order Management. If customer details don't already exist in eBusiness Suite, then this integration synchronizes them when eBusiness Suite receives the sales order.

These integration flows use a connector that you deploy on the SOA server. They also use services from Order Management and the Integrated SOA Gateway in eBusiness Suite. The gateway is part of Oracle eBusiness Suite, release 12.1 and higher. It includes:

- **Integration Repository**. Use public APIs (application program interface) in eBusiness Suite that create and deploy the integration as a web service. Use this integration to send sales orders from Order Management to eBusiness Suite, and to use the Process Order API in eBusiness Suite as a SOA (Service Oriented Architecture) web service.

  In general, this document describes APIs that use PL (Procedural Language) or SQL (Structured Query Language).
- **Service Invocation Framework (SIF)**. Use the business events that eBusiness Suite uses, and use event details to call a web service. This integration uses the Service Invocation Framework in Integrated SOA Gateway to send status update from eBusiness Suite to Order Management. Order Management uses a web service to process the status update response that the Service Invocation Framework calls.

Oracle provides this connector and example integration only for testing purposes. Oracle doesn't support this connector or example integration in a production environment.

For more, see:

- *How Order-to-Cash Works in Order Management*.
- *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*. See the Administering Native Services, Implementing Service Invocation Framework, and Creating and Using Integration Interfaces sections.

Features This Integration Supports

Feature	Description
Synchronize sales orders.	Synchronize entities when you create a sales order in eBusiness Suite. • Customers

Feature	Description
	• Addresses • Contacts
Create new sales orders.	Create sales orders that include different types of items. • Items • Ship sets • Pick-to-order configured item • Assemble-to-order configured item • Kits • Return Material Authorization (RMA)
Revise sales orders.	• Revise order, such as modify values in the order header. • Cancel order. • Revise order lines. o Cancel order line. o Revise an order line, including revising quantity, product configuration, and pricing. o Revise addresses, contacts, and so on. o Add a new order line to an existing sales order. o Revise addresses, contacts, and so on.
Synchronize status updates.	Synchronize order status updates and sales order splits. Here's what you can synchronize. • Order Line status, including Awaiting Shipping, Shipped, Fulfilled, Awaiting Return, and Returned • Scheduled Ship Date • Scheduled Arrival Date • Warehouse • Shipping Method • Line Split

Features This Integration Doesn't Support

- Entities that a sales order or order line references, such as.
  - Sales credit
  - Attachment
  - Shipping charge
  - Tax
  - Lot serial number
- Status updates for shipping details and invoice details
- Warranty or extended warranty

- Sales order flows for assets

Requirements for Using This Integration

Requirement	Description
Release	You must use this release or higher. • eBusiness Suite release 12.1.3 and OM Patch 23249299:R12.ONT.B
Statuses	Order Management sends these statuses. • Schedule Ship Date • Shipped • Fulfilled • Returned • Canceled
Returns	This integration assumes. • Order Management allows a return order to reference the original order so it can identify the sales order you're returning. • Order Management sends only the configured item that eBusiness Suite uses for returns. It doesn't send the order lines that the configured item references.
Synchronization	You must synchronize item data before you use this integration.
Bill of Materials	The BOM (bill of materials) structure for each kit, pick-to-order, and assemble-to-order must be identical in Product Information Management and in eBusiness Suite.
Customer details	If you modify customer details in Order Management, then this integration doesn't synchronize these details directly from the customer master in Order Management to the customer master in eBusiness Suite. This integration synchronizes these details only if it uses the customer or address during ordering, and it only synchronizes the addresses and contacts that the sales order references.

Deployment Options

Use this integration with Order Management, the on-premise version of eBusiness Suite, and with connectors deployed on SOA Cloud Service (PAAS).

Related Topics

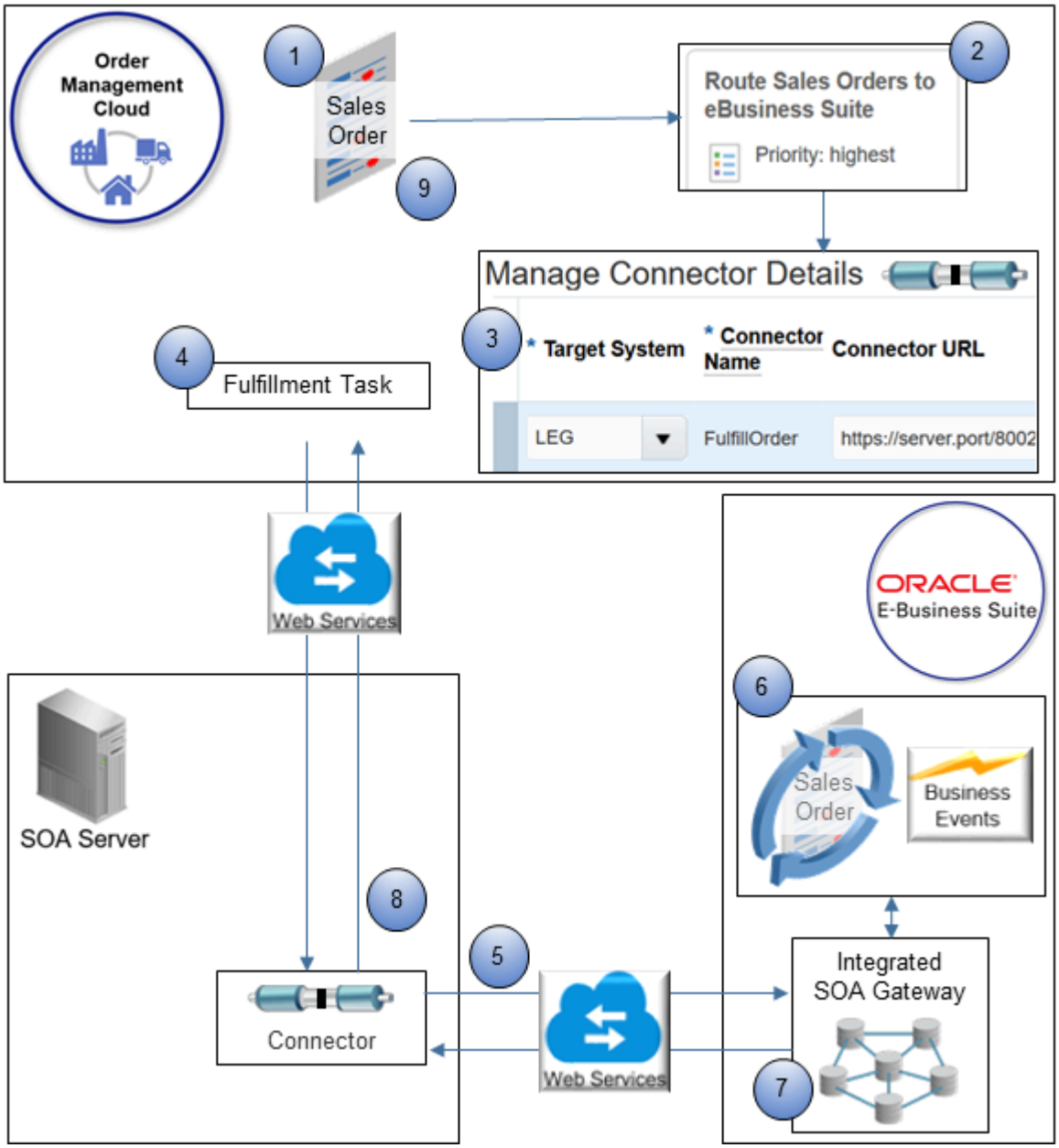
- Integrate eBusiness Suite With Order Management
- Integrate Order Management with eBusiness Suite
- Connect SOA Server to Order Management and eBusiness Suite
- How Order Management Integrates with eBusiness Suite

How Order Management Integrates with eBusiness Suite

Oracle Order Management and Oracle eBusiness Suite use connectors and web services that you set up to communicate details about each of your sales orders.

Sales Order Flow

Here's how the detail flows between Order Management and eBusiness Suite when you create a sales order.



Note

1. You create a sales order in the Order Management work area.
2. Order Management uses routing rules that you set up to determine how to route the sales order to eBusiness Suite, including the connector it will use to do routing.

3. Order Management uses a connector that you set up on the Manage Connector Details page to identify the location of the connector on the SOA server (Service Oriented Architecture). For this example, assume you named the server My_SOA_Server.
4. Order Management uses fulfillment tasks to transform the sales order to a message, and then send it to DooFulfillOrderEBSConnector on My_SOA_Server.
5. The SOA server transforms the message, uses the set up you make in Integrated SOA Gateway on eBusiness Suite to call a web service on the gateway, and then send the message to eBusiness Suite.
6. eBusiness Suite processes the message as a sales order.

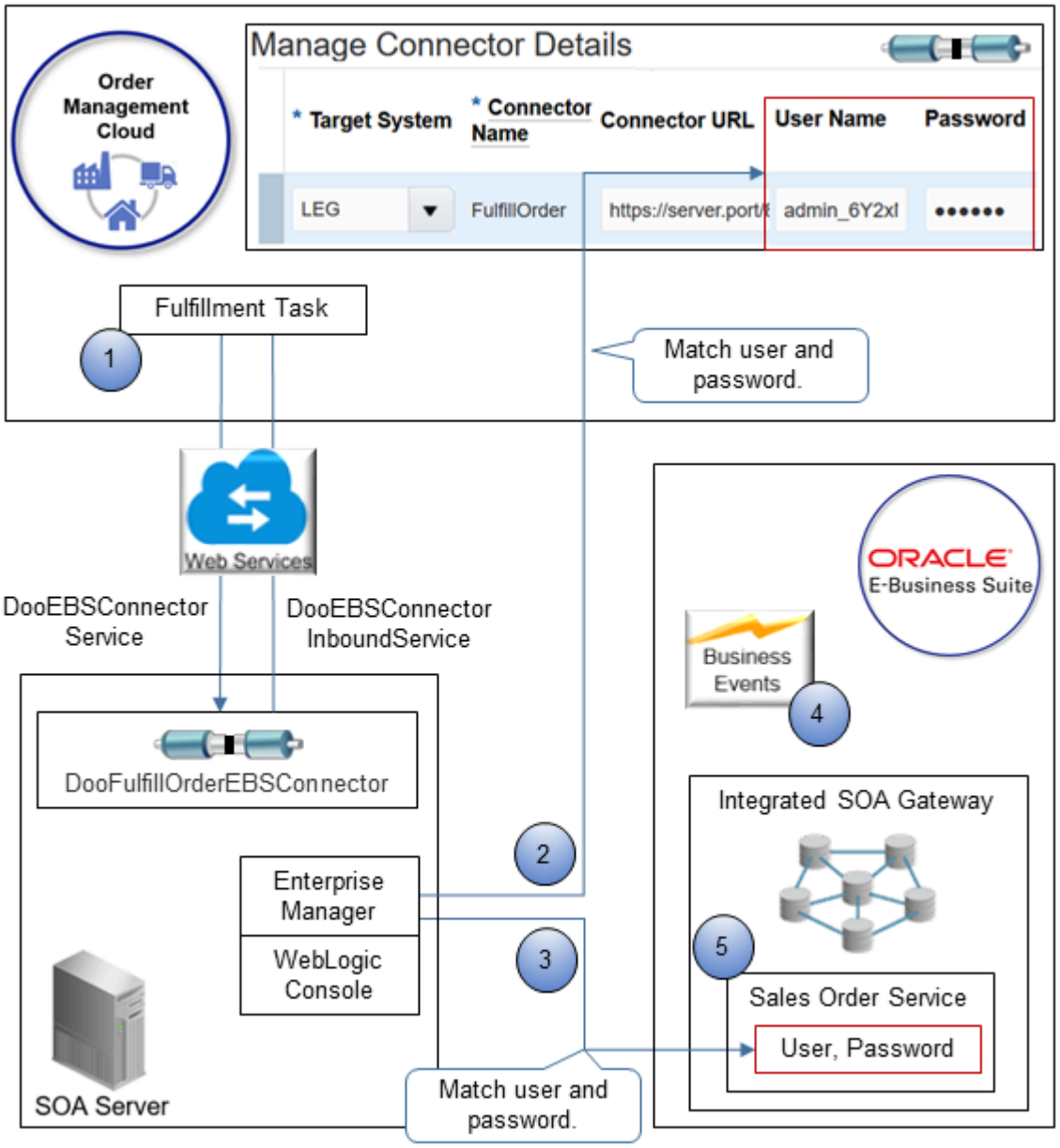
   eBusiness Suite uses a business event during processing.

   For example, if eBusiness Suite schedules the sales order for shipping, or if eBusiness Suite changes scheduled ship date or scheduled arrival date because some other attribute that influences scheduling changes, then eBusiness Suite uses a business event, and then uses the event to send these dates to Order Management. It also sends the warehouse and ship method.

   eBusiness Suite uses the set up you do on business events in eBusiness Suite to communicate status changes. You enable the business event, subscribe to it so eBusiness Suite knows when to use it, and specify the web service to use when communicating with My_SOA_Server.
7. eBusiness Suite uses your set up for the web service and business event in Integrated SOA Gateway to send the update to the connector on My_SOA_Server.
8. The connector on My_SOA_Server server sends status updates to Order Management.
9. Order Management updates sales order attributes in the Order Management work area.

Order Management also uses this flow for a status update or order line split. Other flows use a similar sequence.

Flow Through SOA Server



Note

1. The fulfillment task in Order Management uses DooEBSConnectorService to call the fulfillOrderEBSRequest operation of DooFulfillOrderEBSConnector on My_SOA_Server.

Connector Attribute	Value
Partner Link	DooEBSConnectorService This value is the name of the connector.
Port Type	FulfillOrderEBSProcess

Connector Attribute	Value
Operation	fulfillOrderEBSRequest

The fulfillment task in Order Management manages the create flow, update flow, and hold flow.

2. The fulfillment task uses the security that you set up in Oracle Enterprise Manager and the WebLogic Console on My_SOA_Server. The user and password that you use in Enterprise Manager must match the user and password you use on the Manage Connector Details page.

   For details, see *Oracle Enterprise Manager*.
3. My_SOA_Server uses the set up you do in Enterprise Manager and WebLogic Console to call the Sales Order Service in Integrated SOA Gateway on eBusiness Suite. The user and password you use in Enterprise Manager must match the user and password you use in eBusiness Suite.
4. An event happens in eBusiness Suite that requires communication with Order Management, such as an update to the sales order status.
5. Sales Order Service in eBusiness Suite calls the fulfillOrderEBSInboundRequest operation on DooFulfillOrderEBSConnector.

Connector Attribute	Value
Partner Link	DooEBSConnectorInboundService_ep
Port Type	FulfillOrderEBSProcess
Operation	fulfillOrderEBSInboundRequest

eBusiness Suite uses the security you set up in Enterprise Manager on My_SOA_Server for the eBusiness Suite user that calls DooFulfillOrderEBSConnector.

Related Topics

- Integrate eBusiness Suite With Order Management
- Integrate Order Management with eBusiness Suite
- Overview of Integrating Order Management with eBusiness Suite
- Connect SOA Server to Order Management and eBusiness Suite

Integrate Order Management with eBusiness Suite

Integrate Order Management with eBusiness Suite.

Summary of the Set Up

1. Add the connector.
2. Route sales orders to eBusiness Suite.

3. Route items to eBusiness Suite.
4. Route configured items to eBusiness Suite.
5. Route returns to eBusiness Suite.
6. Deploy the connector. For details, see *Connect SOA Server to Order Management and eBusiness Suite*.
7. Integrate eBusiness Suite. For details, see *Integrate eBusiness Suite With Order Management*.

Add the Connector

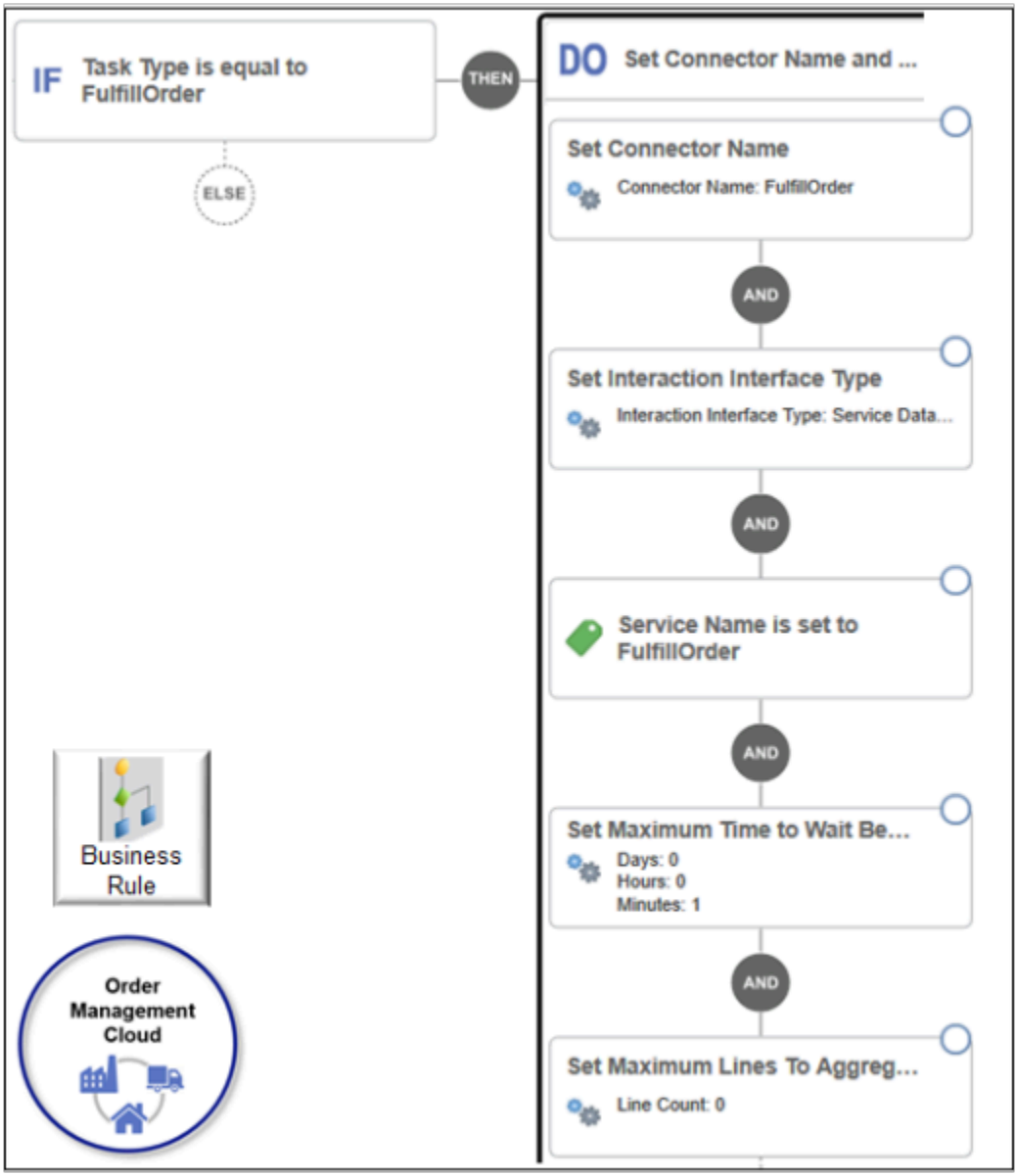
1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Interface Web Service Details
2. On the Manage Connector Details page, create a new connector.

Attribute	Value
Target System	Agile
Connector Name	FulfillOrder
Connector URL	Enter the URL that locates the connector. For example: `https://server.port/8002/soa-infra/services/default/DooFulfillOrderEbsComposite`
User Name and Password	Enter any user name and password. It isn't necessary to enter a user name and password that you already set up. Order Management uses this name and password when it calls the connector on the SOA server.

For details, see *Connect Order Management to Your Fulfillment System*.

Route Sales Orders to eBusiness Suite

You will create a routing rule.



Learn how to use the rules editor. For details, see *Manage Routing Rules*.

Route sales orders from Order Management to eBusiness Suite.

1. Open the Manage External Interface Routing Rules for Sales Orders page.

   For details, see *Overview of Using Business Rules With Order Management*.
2. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Route Sales Orders to eBusiness Suite
Description	Route sales orders from Order Management to eBusiness Suite.

Attribute	Value

3. In the If area, create a statement.

```
If Task Type is equal to FulfillOrder
```

4. In the Do area, create a statement for each attribute.

Attribute	Value
Connector Name	FulfillOrder
Interaction Interface Type	Service Data Object
Service Name	Is set to FulfillOrder
Maximum Time to Wait Before Sending	1 minute
Maximum Lines to Aggregate and Send	0

5. Click **Save and Close**.
6. Click the rule you just created. In the dialog that displays, set the values, then click **Save and Close**.

Attribute	Value
Priority	Highest
Activate Rule	Contains a check mark.

Route Items to eBusiness Suite

You will create a rule.

Do it.

1. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Route Items to eBusiness Suite
Description	Route sales orders that don't include a configured item from Order Management to eBusiness Suite.

2. In the If area, create statements.

```
If Category Code is equal to RETURN
and
Item Subtype is equal to Standard
```

3. In the Do area, create a statement for the attribute.

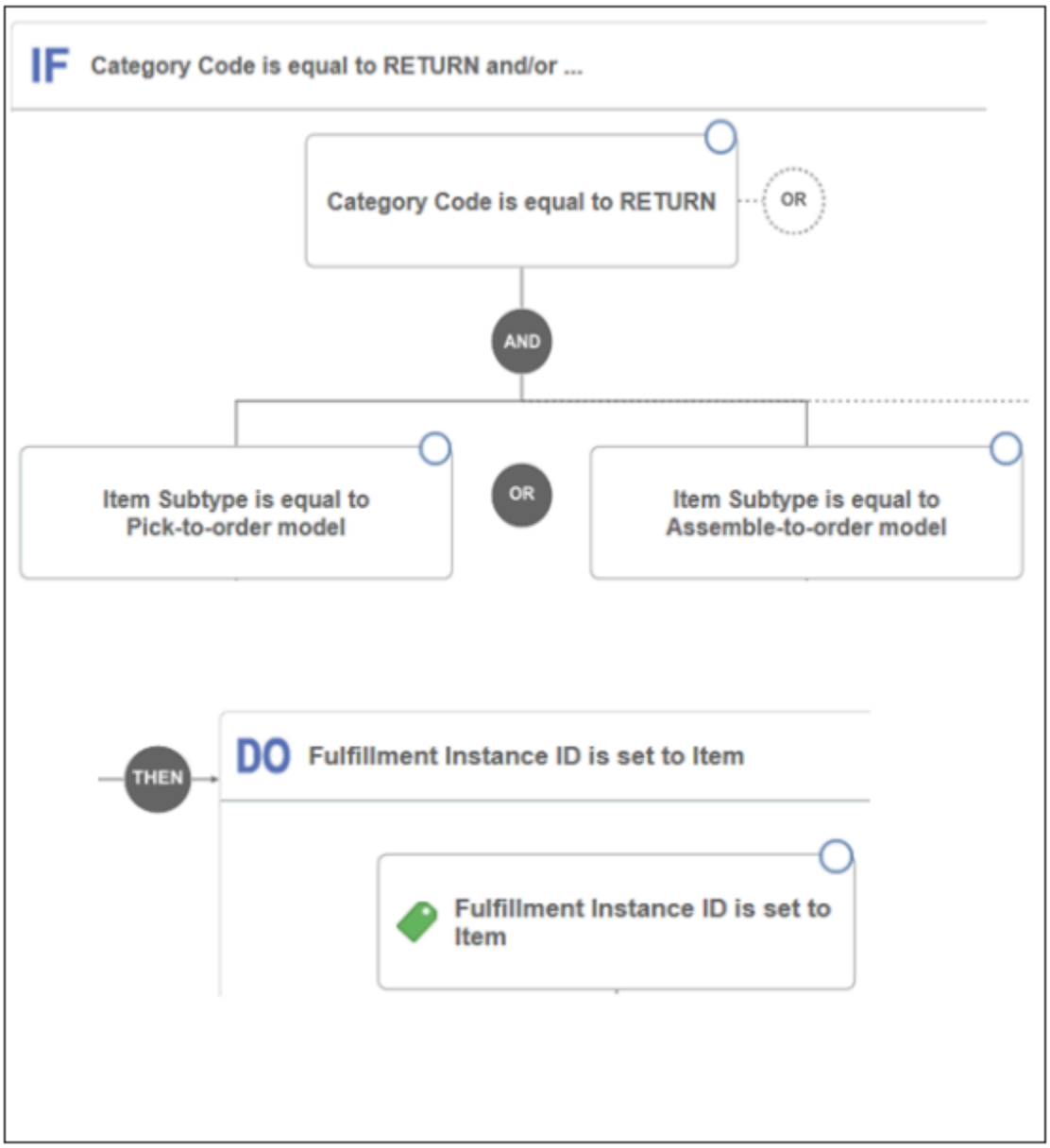
Attribute	Value
Fulfillment Instance ID	Is set to Item

4. Click **Save and Close**.
5. Click the rule you just created. In the dialog that displays, set the values, and then click **Save and Close**.

Attribute	Value
Priority	Medium
Activate Rule	Contains a check mark.

Route Configured Items to eBusiness Suite

You will create a routing rule.



Create the routing rule for the configured item.

1. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set these values.

Attribute	Value
Name	Route Configured Items to eBusiness Suite
Description	Route sales orders that include a configured item from Order Management to eBusiness Suite.

2. In the If area, create statements.

```
If Category Code is equal to RETURN
and
Item Subtype is equal to Pick-to-order model
or
Item Subtype is equal to Assemble-to-order model
```

3. In the Do area, create a statement for the attribute.

Attribute	Value
Fulfillment Instance ID	Is set to Item

4. Click **Save and Close**.
5. Click the rule you just created. In the dialog that displays, set the values, then click **Save and Close**.

Attribute	Value
Priority	Medium
Activate Rule	Contains a check mark.

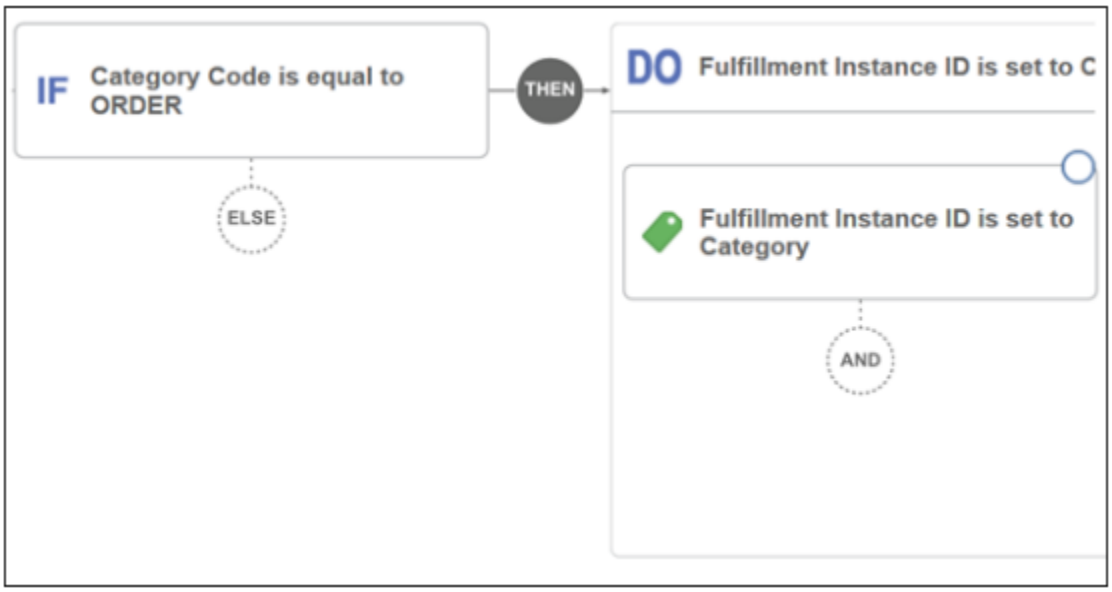
Route Returns to eBusiness Suite

Order Management sends only the line for the configured item to eBusiness Suite so eBusiness Suite can process the receipt. To create a return for a configured item, eBusiness Suite gets order lines from the original configured item and uses them to create return lines in the return material authorization that it uses for the configured item. So, you must create a rule that includes filter criteria that enables this integration to process return lines.

This example uses the Fulfill Order step to process requests.

- Create Order
- Create Return Order
- Create Return Order for Models

You will create a rule.



Create a routing rule that supports return orders that include a configured item.

1. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Route Return Orders to eBusiness Suite
Description	Route return orders that include a configured item from Order Management to eBusiness Suite.

2. In the If area, create a statement.

```
If Category Code is equal to ORDER
```

> **Tip:** Click **New Condition**, then, in the Create Condition dialog, enter `Category`, wait for the list to display values, then click `Category Code (Order Fulfill Line).`

3. In the Do area, create a statement for the attribute.

Attribute	Value
Fulfillment Instance ID	Is set to Category

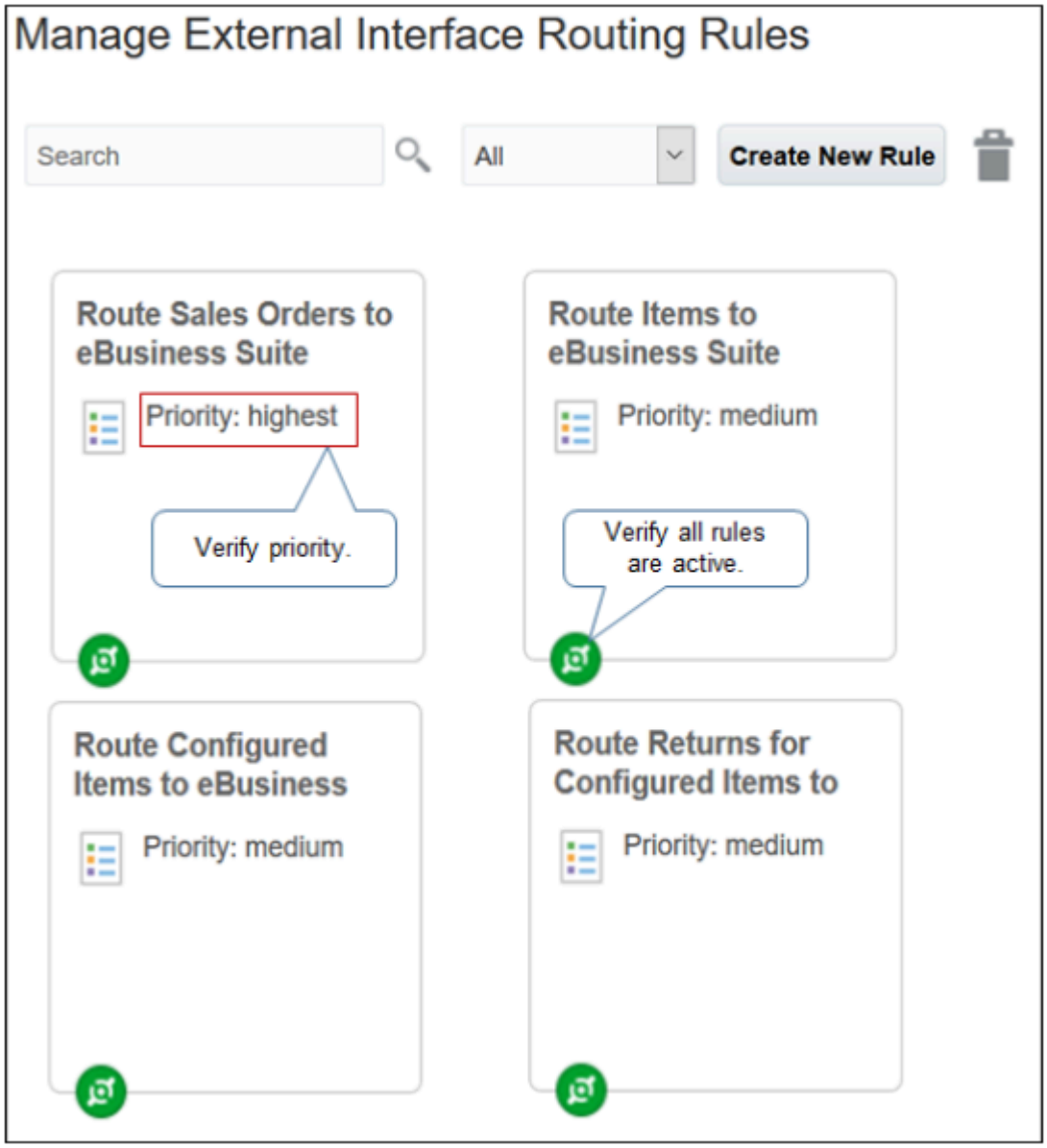
> **Note:** Click **New Action > Set a Value**. In the Create Action dialog, set the top value to Fulfillment Instance ID, then, below **is set to**, click **Attribute**, then enter `Category`. Using this technique displays attributes that are already defined, and helps to make sure you choose an attribute that the rule can use.

4. Click **Save and Close**.

5. Click the rule you just created. In the dialog that displays, set the values, then click **Save and Close**.

Attribute	Value
Priority	Medium
Activate Rule	Contains a check mark.

6. Verify that you set the Route Sales Orders to eBusiness Suite rule to the highest priority, and that you activated each of your rules.



Related Topics

- Overview of Integrating Order Management with eBusiness Suite
- Connect SOA Server to Order Management and eBusiness Suite
- Manage Connector Details Between Order Management and Your Fulfillment System
- Route Requests from Order Management to Fulfillment Systems
- How Order Management Integrates with eBusiness Suite

Connect SOA Server to Order Management and eBusiness Suite

Set up a connector on your SOA server (Service Oriented Architecture). This server uses web services to communicate each sales order between Order Management and Oracle eBusiness Suite.

Summary of the Set Up

1. Set up connector.
2. Add user and password to call Order Management.
3. Add user and password to call eBusiness Suite.
4. Administer security.

Set Up Connector

Set up the connector on the SOA server. For this example, assume the server is named My_SOA_Server.

1. Click *here* to download DooFulfillOrderEBSConnectorRev1.0.zip to My_SOA_Server.
2. Unzip DooFulfillOrderEBSConnectorRev1.0.zip.
3. Use an XML editor to open DooFulfillOrderEBSComposite_cfgplan.xml.
4. Modify the responsibility.

```
</property>
 <property name="bpel.preference.responsibility">
 <replace>EBS responsibility</replace>
</property>
```

   where

   - `EBS responsibility` specifies the responsibility.

   For example:

```
</property>
 <property name="bpel.preference.responsibility">
 <replace>ORDER_MGMT_SUPER_USER</replace>
</property>
```

5. Replace the host and port for the references.

Reference	Value
Sales Order Services OE_INBOUND_INT, Internal Name:PROCESS_ORDER_25	<reference name="OEIboundIntPOService"> <replace>http://server:port/webservices/SOAProvider/plsql/oe_inbound_int/?wsdl</replace>

Reference	Value
Sales Order Outbound Services OE_OUTBOUND_INT, Internal Name:SYNC_ORDER_25	<reference name="OEOutboundIntSyncService"> <replace>http://server:port/webservices/SOAProvider/plsql/oe_outbound_int/?wsdl</replace>
Task Layer	<reference name="FulfillOrderEBSResponse"> <replace>http://server:port/soa-infra/services/default/DooTaskFulfillOrderResponseInterfaceComposite/fulfillmentresponse?wsdl</replace>

Add User and Password to Call Order Management

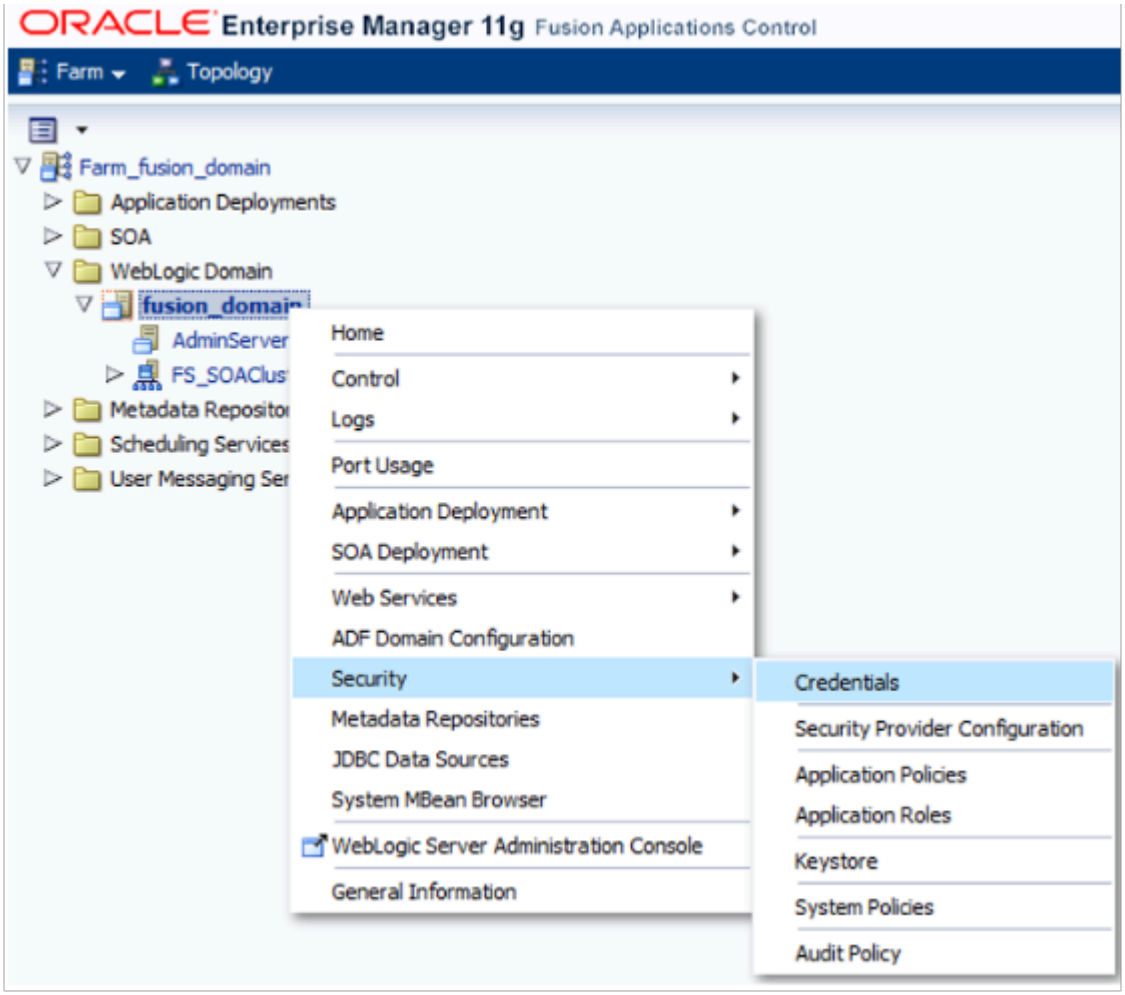
Add the user and password that the connector on SOA server must use to call Order Management.

1. Sign into My_SOA_Server.
2. Sign into Oracle Enterprise Manager.

   For details about Oracle Enterprise Manager, see https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.

3. In the Farm_fusion_domain tree, Expand **Weblogic Domain**, right-click **fusion_domain**, then click **Security > Credentials**.

   For example:



4. In the Credentials area, expand **oracle.apps.security**, then click **Create Key**.
5. In the Create Key dialog, set the values, then click **OK**.

Attribute	Value
Select Map	oracle.apps.security
Key	FUSION_APPS_FOM_CON_APPID-KEY
Type	Password
User name and Password	Use the same user name and password that the connector uses to call fulfillment tasks in Order Management.

Attribute	Value

Add User and Password to Call eBusiness Suite

Add the user and password that the connector must use to call eBusiness Suite.

1. In the Credentials area, expand **oracle.wsm.security**, then click **Create Key**.
2. In the Create Key dialog, set the values, then click **OK**.

Attribute	Value
Select Map	oracle.wsm.security
Key	FUSION_APPS_FOM_EBS_APPID-KEY
Type	Password
User name and Password	○ Use the user name and password that the connector must use to call the Process Order service in eBusiness Suite. ○ You must create grants when you create and deploy Process Order. ○ You must use the same user name for FUSION_APPS_CON_EBS_APPID-KEY that you use for the grants. For details, see *Use Oracle E-Business Suite Business Events to Trigger Integration Endpoint in Oracle Integration*.

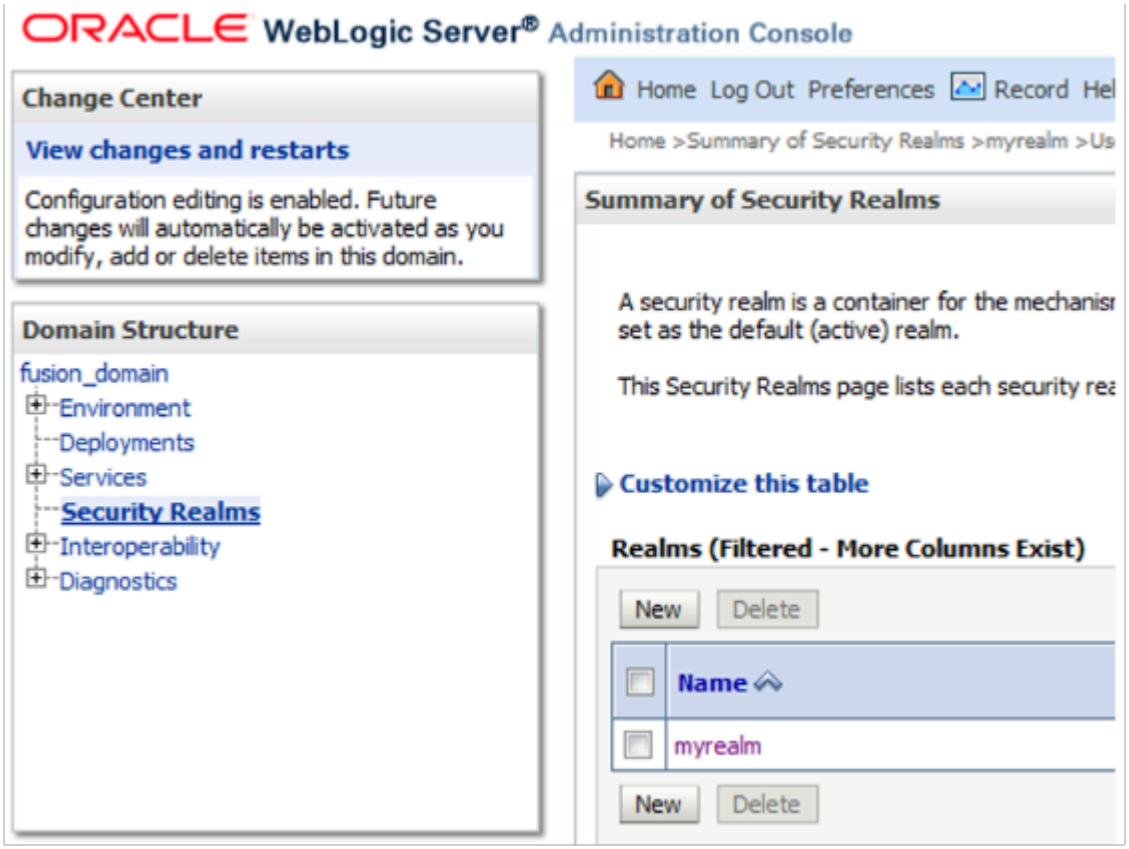
Administer Security

Administer security for the Order Management user and the eBusiness Suite user that will call the connector.

1. On My_SOA_Server, sign into Weblogic Console.

2. In the Domain Structure area, click **Security Realms**, then, in the Realms area, click `myrealm`.

   For example:



3. Click **Users and Groups**, then click **New**.
4. In the Create a New User dialog, enter the same user name and password that you enter on the Manage External Interface Web Service Details page when you do the integration. For details, see *Integrate eBusiness Suite With Order Management*.
5. Click **OK**.
6. Add another user.

   Repeat steps 1 through 6, except enter the user name and password of a user that resides in eBusiness Suite. eBusiness Suite uses this name and password when it calls DooFulfillOrderEBSConnector on the SOA server.

Related Topics

- Integrate eBusiness Suite With Order Management
- Integrate Order Management with eBusiness Suite
- Overview of Integrating Order Management with eBusiness Suite
- How Order Management Integrates with eBusiness Suite

Integrate eBusiness Suite With Order Management

Set up the Integrated SOA Gateway and business events when you integrate Oracle eBusiness Suite with Oracle Order Management.

Summary of the Set Up

1. Set up web services.
2. Set up business events.
3. Set up subscription for business event.
4. Set up security for business event.
5. Set up eBusiness Suite so it can communicate status.

Set Up Web Services

This topic describes how to set up web services in Integrated SOA Gateway (Service Oriented Architecture). For background details, see:

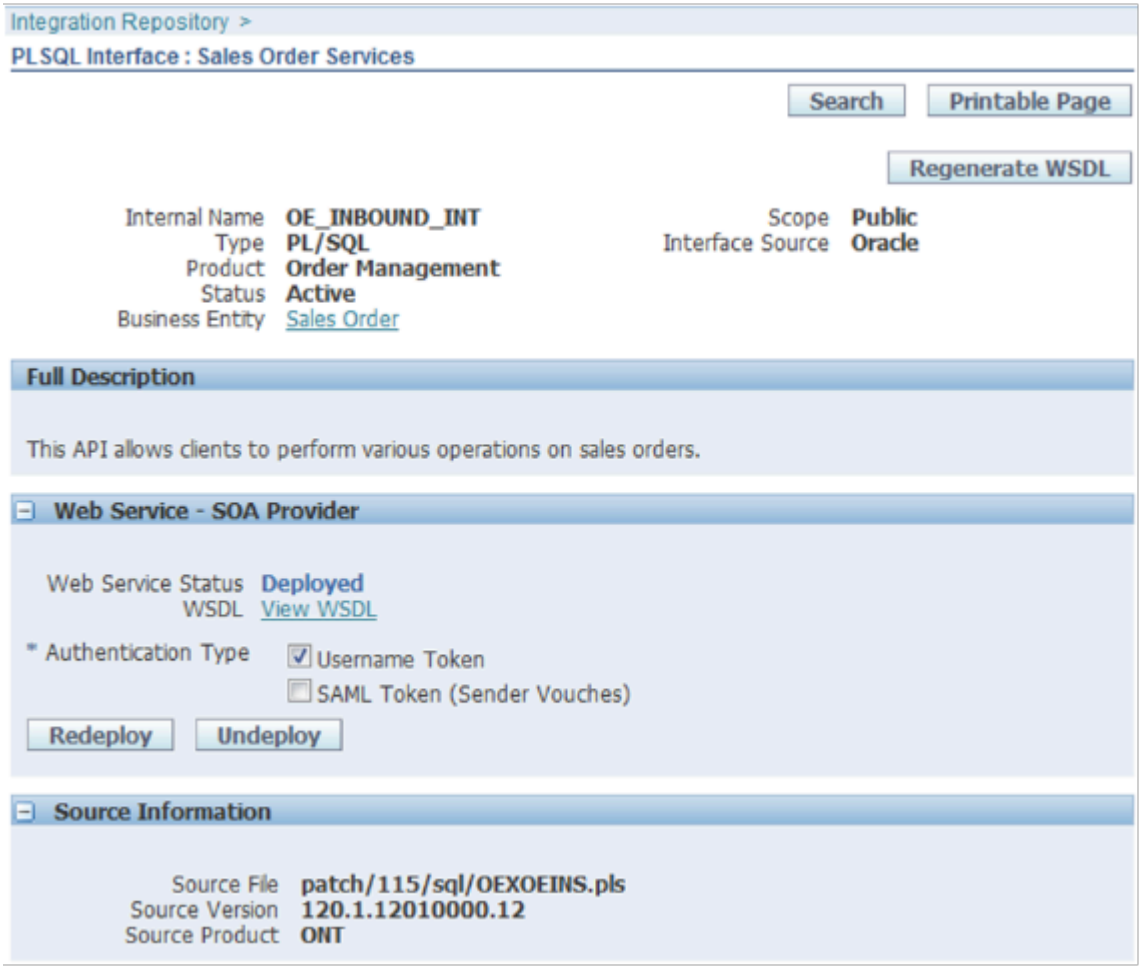
- *Oracle E-Business Suite Integrated SOA Gateway Developer's Guide*.
- *Installing Oracle E-Business Suite Integrated SOA Gateway, Release 12.1.3 (Doc ID 556540.1).*

Set up web services.

1. Sign into eBusiness Suite with administrative privileges.
2. Expand responsibility **Integrated SOA Gateway**, then click **Integration Repository**.
3. In the Integration Repository area, expand **Order Management Suite**, expand **Order Management**, then click **Sales Order**.
4. In the table that displays, click the **Sales Order Services** link.

5. In the Sales Order Services area, click **Regenerate WSDL**.

   For example:



6. In the Procedures and Functions area, add a check mark to the **Select** option for the row that includes these values.

Name	Internal Name	Description
Sales Order Service	PROCESS_ORDER_25	Dedicated for Order Management integration

7. Click **Create Grants**.

   If a global grant exists for all users, then it might not be necessary to create an individual grant.

8. Set the value.

Attribute	Value
Authentication Type	`Username Token`

Attribute	Value

9. Click **Deploy**.
10. Click **View WSDL**, then note the URL that the WSDL (Web Service Definition Language) uses.

    You use this URL when you set up the connector on the SOA server. Make sure this URL is similar to this URL:

    ```
    http://server:port/wEBServices/SOAProvider/plsql/oe_inbound_int/?wsdl
    ```

    where

    - `server:port` identifies the address of the server that hosts the web service.
11. Repeat steps 1 through 10 to create and deploy another service.

Attribute	Value
Name	Sales Order Outbound service
Internal Name	OE_OUTBOUND_INT
Description	Contains procedures to generate outbound integration information
Procedures and Functions Internal Name	SYNC_ORDER_25

This service uses an API in eBusiness Suite.

Set Up Business Events

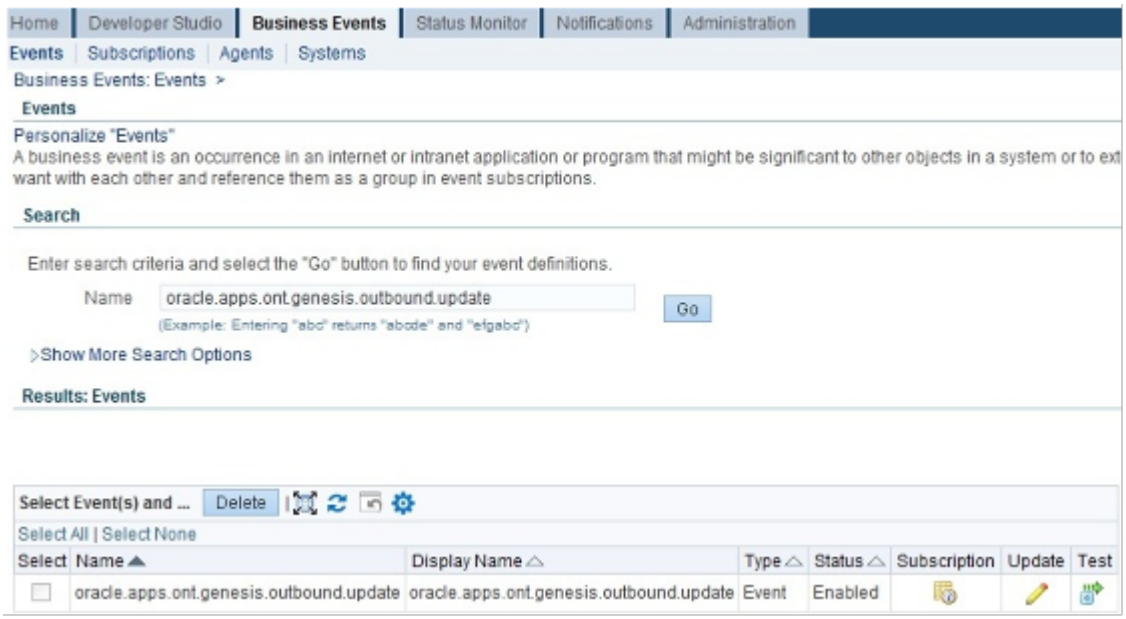
To support the status update flow, you must set up the event and subscription so they can call the response service in Order Management. The business event already exists in eBusiness Suite, but you must set it up.

1. In eBusiness Suite, expand the **Workflow Administrator Web Applications** responsibility, then click **Administrator Workflow**.
2. Click **Business Events**.

3. On the Business Events tab, in the Search area, in the Display Name attribute, enter the value.

   ```
   oracle.apps.ont.genesis.outbound.update
   ```

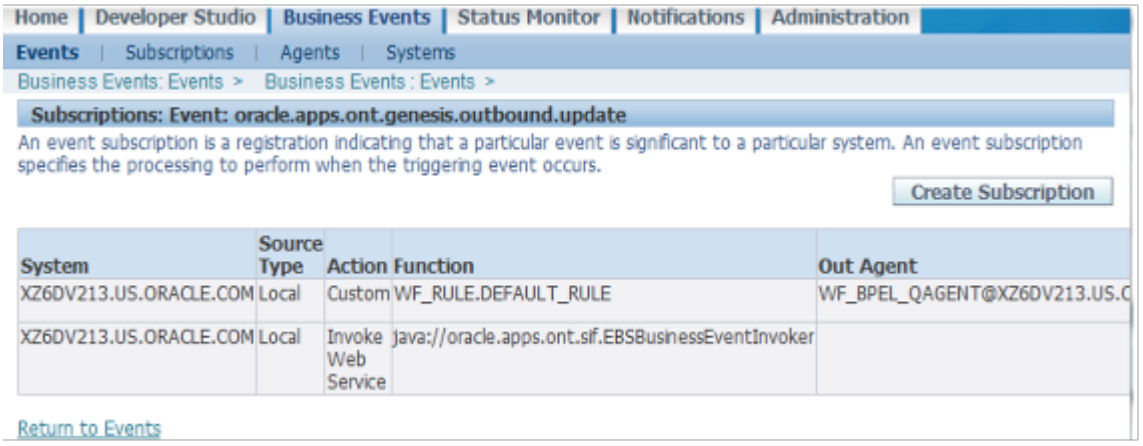
   For example:



4. Click **Go**.

Set Up Subscription for Business Event

1. Click **Create Subscription**.



2. In the Search and Select area, enter values.

Attribute	Value
Search By	System Name

Attribute	Value
Value	Enter the name of the eBusiness Suite system. For example, enter **`XZ6DV213.your.address`**.

3. Click **GO**.
4. Wait for the result to display, then click **Select**.

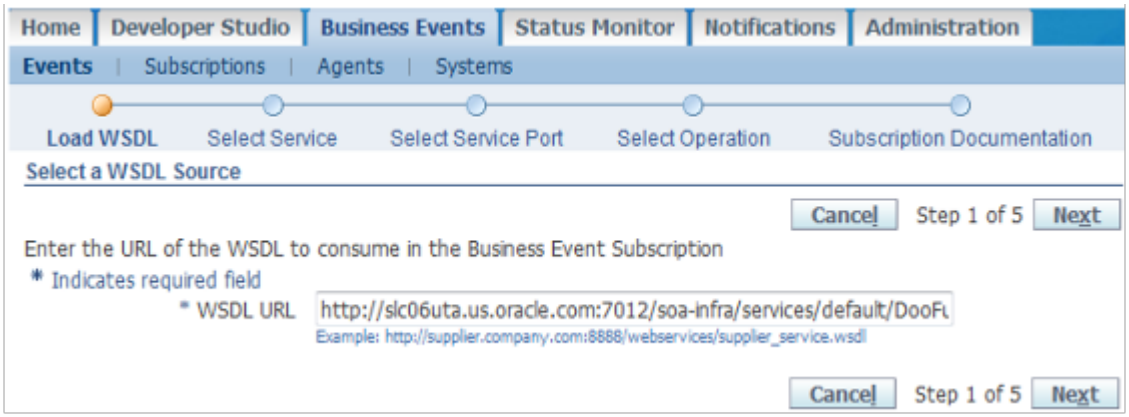
   For example, click **Select** in the row that contains these values.

Attribute	Value
System Name	XZ6DV213.your.address
Display Name	LA5099
Description	Local system created by Oracle Workflow Configuration Assistant

5. In the Update Event Subscriptions area, set the values, then click **Next**.

Attribute	Value
Source Type	Local
Event Filter	oracle.apps.ont.genesis.outbound.update
Phase	100
Status	Enabled
Rule Data	Message
Action Type	**`Invoke Web Service`**
On Error	Stop and Rollback

6. Notice that step 1 of the task-based user interface displays. This interface guides you through the set up.



7. On the Load WSDL step, enter the URL that locates the WSDL of the connector, then click **Next**.

Attribute	Value
WSDL URL	`http://server:port/soa-infra/services/default/DooFulfillOrderEbsComposite/DooEbsConnectorInboundService_ep?WSDL`

8. On the Select Service step, set the value, then click **Next**.

Attribute	Value
Service Name	DooEBSConnectorService_eb Note that DooEBSConnectorService is the name of the connector.

9. On the Select Service Port step, make sure the Select option is enabled for the service.

Service Port	Port End Point
FulfillOrderEbsProcess_pt	http://server:port/soa-infra/services/default/DooFulfillOrderEbsComposite/DooEbsConnectorInboundService_ep

10. Click **Next**.
11. On the Select Operation step, make sure the Select option is enabled for the operation.

Operation	Port Type	Operation Type
FulfillOrderEbsInboundRequest	FulfillOrderEbsProcess	ONE_WAY,0

12. Click **Next**.

13. On the Subscription Documentation step, set the values, then click **Apply**.

Attribute	Value
Java Rule Function	`oracle.apps.ont.sif.EBSFOMEventInvoker` This Java rule function is part of the eBusiness Suite patch. It allows the connector to use the status update flow.
Owner Name	Oracle Workflow
Owner Tag	FND

Set Up Security for Business Events

1. On the Event Subscription page, in the Subscription Parameters area, set the values.

Attribute	Value
WFBES_SOAP_USERNAME	Enter the user name of the service that you're calling. For example: `OPERATIONS`
WFBES_SOAP_PASSWORD_MOD	Enter the module name. For example: `ONT`
WFBES_SOAP_PASSWORD_KEY	Enter the key name. For example: `operationsKey`

2. Store the password so its secure. Sign into the database that resides on the eBusiness Suite server, then use SQL to query the Order Management database.

```
sqlplus username/password@database_name @$FND_TOP/sql/afvltput.sql ONT passwordKey password
```

For example:

```
sqlplus apps/apps@vi7kr759 @$FND_TOP/sql/afvltput.sql ONT operationsKey password
```

Set Up eBusiness Suite So It Can Communicate Status

eBusiness Suite uses business events to communicate status. You must set them up.

1. Expand **Order Management Super User > Setup > Orders**, then click **AIA Sales Order Status**.

2. In the AIA Sales Order Status Synchronization dialog, add a check mark to the `Sync Reqd?` option for each of the statuses that this integration must communicate.
    - Awaiting Shipping
    - Shipped
    - Fulfilled
    - Awaiting Return
    - Returned

Related Topics

- Use SQL to Query Order Management Data
- Integrate Order Management with eBusiness Suite
- Overview of Integrating Order Management with eBusiness Suite
- Connect SOA Server to Order Management and eBusiness Suite
- How Order Management Integrates with eBusiness Suite

Flows That Order Management Uses with eBusiness Suite

Order Management communicates order details to and from Oracle eBusiness Suite.

Flow from Order Management to eBusiness Suite

Sales Order Details That Order Management Sends to eBusiness Suite

Details	Description
Customer details	• Bill To Account • Ship To Account • Bill To Address • Ship To Address • Contacts
Order Lines	• Items, including simple items or complex items, such as kits or pick-to-order configured items. • Prices. This integration sets the CALCULATE_PRICE_FLAG attribute to P when it synchronizes the sales order so eBusiness Suite doesn't recalculate prices on the order line. This setting primarily affects freight charges. • Ship Set. • Requested Ship Date. • Payment Terms.
Scheduling Details	If you use Global Order Promising (GOP), then Order Management sends these details. • Schedule Ship Date • Schedule Arrival Date • Ship From Warehouse

Details	Description
	• Shipping Method

Synchronize Sales Order Status Between Order Management and eBusiness Suite

This integration synchronizes statuses between eBusiness Suite and Order Management while it processes the sales order through fulfillment.

Context	Order Management Status	eBusiness Suite Status
Create sales order.	Awaiting Fulfillment	Booked
Schedule or reschedule the sales order in eBusiness Suite.	Awaiting Fulfillment	Awaiting Shipping
Ship an order line in eBusiness Suite.	Shipped	Shipped
Line reaches Fulfilled status in eBusiness Suite.	Fulfilled	Fulfilled

Synchronize Customer Details in Sales Orders

This integration synchronizes customer details from Order Management to eBusiness Suite when it creates or updates a sales order.

- Parties or Accounts
- Addresses
- Contacts

If the customer, address, or contact for a new customer doesn't exist in eBusiness Suite, then eBusiness Suite creates them.

In Order Management, attribute Sold To and attribute Ship To are each a party, and attribute Bill To is an account. However, each of these attributes is an account in eBusiness Suite. To solve this problem for Ship To and Bill To, Order Management sends an address when it sends the Ship To customer or Bill To customer to eBusiness Suite. eBusiness Suite uses the combination of address and customer details to identify the account it must use.

The Sold To attribute doesn't include an address, so the integration determines the account differently depending on the condition.

Condition	Result
Order Management sends a party name that doesn't exist in eBusiness Suite.	eBusiness Suite creates the party and the customer account.
Order Management sends a party name, but more than one party in eBusiness Suite matches the name that Order Management sends.	eBusiness Suite creates an error and exits the flow.

Condition	Result
Order Management sends a party name that exists in eBusiness Suite, but eBusiness Suite doesn't contain a customer account for this party.	eBusiness Suite creates a new customer account for the party.
Order Management sends a party name that exists in eBusiness Suite, and eBusiness Suite contains one customer account for this party.	eBusiness Suite uses the party and customer account that Order Management sends.
Order Management sends a party name that exists in eBusiness Suite, but eBusiness Suite contains more than one customer account for this party.	If. • Only one of the accounts in eBusiness Suite uses the same description that the party name uses, then eBusiness Suite uses this account. • More than one of the accounts in eBusiness Suite uses the same description that the party name uses, then eBusiness Suite creates an error and exits this flow. To correct this problem, you must update the description in eBusiness Suite on one of the accounts so it uses the same description that the party name uses, and then resubmit the sales order.

Create Sales Orders for Configured Items

This integration supports configured items that use pick-to-order and assemble-to-order.

Create Sales Orders for Configured Items That Use Pick-to-Order

Here's the sequence this integration uses to create a sales order for a configured item that uses pick-to-order.

1. Order Management sends the entire configured item to eBusiness Suite. The status for the configured item and each configure option is Awaiting Fulfillment.
2. eBusiness Suite triggers events that communicate shipping statuses for Awaiting Shipping and Shipped. eBusiness Suite triggers the events only for shippable lines.
3. eBusiness Suite sends events to Order Management while it ships each line.
4. Order Management updates the shippable lines. These lines are in the Shipped status while the remaining lines, such as Model or Option Class, are in the Awaiting Fulfillment status.
5. eBusiness Suite finishes shipping each order line, the flow moves to the Fulfillment activity, then this step marks each line as Fulfilled. To allow this step to happen, you must enable the Fulfilled status.
6. eBusiness Suite creates an event for the Fulfilled status for each order line.
7. Order Management updates each status and marks each order line of the configured item as Fulfilled.

Create Sales Orders for Configured Items That Use Assemble-to-Order

Here's the sequence this integration uses to create a sales order for a configured item that uses assemble-to-order. Note that eBusiness Suite stores an assemble-to-order configured item as a separate line.

1. Order Management sends the configured item to eBusiness Suite.
2. eBusiness Suite creates the configured item in eBusiness Suite.
3. eBusiness Suite creates an event for the order lines that the configured item references while it fulfills the assemble-to-order.
4. eBusiness Suite synchronizes each status to Order Management in the same way it synchronizes for a pick-to-order configured item, except Order Management stores each configured item as an attribute while eBusiness

Suite stores each configured item as an order line. So, this integration doesn't synchronize status updates for these order lines from eBusiness Suite to Order Management.

5. Order Management updates the status for the configured item, configure options, and items in the same way it updates them for a pick-to-order configured item.
6. eBusiness Suite eventually closes each order line, then creates a status change event. The connector ignores the Closed status. This integration marks each line in Order Management as Closed when the process finishes all steps.

   If a fulfillment task is the last step in the process, then the integration marks each order line as Closed when the fulfillment task finishes.

Map Status When Creating Sales Order for Configured Item

This integration maps statuses between Order Management and eBusiness Suite when it creates a sales order for a configured item the same way it maps statuses for an item that isn't configured. For details, see the Mapping Statuses When Creating Sales Order for Items section.

Revise Sales Order

This integration supports revisions to a sales order, such as revising an attribute on the order header. Here's the sequence it uses to revise a sales order.

1. Order Management sends a request to eBusiness Suite to place each order line on hold. If eBusiness Suite.
   - **Rejects the hold request**. This integration sends an exception message to Order Management and ends this sequence.
   - **Successfully places the hold**. Order Management sends a change request to eBusiness Suite.
2. eBusiness Suite processes the change, then this integration releases the hold on the order lines.

Note

- If the change management flow starts, then this integration releases any hold that already existed on the sales order in eBusiness Suite.
- Change management applies the hold at the start of processing and releases it when the change finishes.
- eBusiness Suite can't distinguish whether change management or a user applied the hold. eBusiness Suite only applies one hold when Order Management sends a hold request. If this integration releases a hold as a result of change management, then eBusiness Suite releases the hold in Order Management.
- eBusiness Suite doesn't automatically release a change management hold. Instead, it applies the release hold request that it receives during the change management flow.

Revise Order Line

This integration supports revisions on an order line.

- Modify quantity, configuration, or pricing on an order line.
- Modify addresses or contacts on an order line.
- Add a new order line to a sales order that already exists.
- Cancel an order line.

Split an Order Line

An order line split might happen when.

- Order Management determines it can't meet a customer requirement through a single shipment, then it splits the order line during scheduling and ships the item in more than one shipment.
- An end-user splits the order line. For example, to meet a customer requirement to receive part of the sales order.
- The on-hand inventory is less than the ordered quantity and eBusiness Suite ships only part of the quantity, then a split might happen when confirming the shipment.

Ship Item That Your User Splits

Here's the sequence that this integration uses when the user splits an order line.

1. An end-user splits an order line in eBusiness Suite.
2. eBusiness Suite creates a split event that splits the order line.
3. Order Management receives the event and splits the original order line. Each order line in Order Management is in status Awaiting Fulfillment.
4. eBusiness Suite ships the order line and creates a shipped status event for each line.
5. The connector processes this event and sets the order line status to Shipped.
6. The flow in eBusiness Suite reaches the Fulfillment activity, then creates an event for each line.
7. Order Management processes each event and sets the order line status to Fulfilled.

Split Item When eBusiness Suite Ships Part of an Order Line

Here's the sequence that this integration uses when eBusiness Suite ships only part of an order line.

1. eBusiness Suite creates a split event.
2. The connector processes this event and Order Management splits the order line.
3. Order Management sets the status of order line x that eBusiness Suite shipped to Shipped, and it sets the status of order line y that contains the remaining quantity to Awaiting Shipping.
4. eBusiness Suite ships the remaining quantity and creates another event for the Shipped status.
5. Order Management sets the status of order line y to Shipped.
6. The flow in eBusiness Suite reaches the Fulfillment activity, then creates an event for each order line.
7. The connector processes each event and Order Management sets the status for each order line to Fulfilled.

Split a Configured Item

This integration supports only a proportional split when splitting a configured item. For example:

- Assume a configured item includes a quantity of 5, and configure option x of the configured item includes a quantity of 10.
- Assume you split the configured item into a quantity of 2 for configured item b and a quantity 3 for configured item c.
- A proportional split will split option x into a quantity of 4 for item b and 6 for item c.

To support this flow, you must set up a process that includes Step Level Line Criteria to send only the configured item for a return. For details, see *Integrate eBusiness Suite With Order Management*.

Here's the sequence that this integration uses when eBusiness Suite splits a configured item.

1. eBusiness Suite splits a configured item, creates a split event, then sends it to Order Management.
2. The connector processes the event and Order Management splits the configured item.

If the split references a partial shipment, then Order Management sets the status for each shippable line to Shipped for the part of the configured item that eBusiness Suite shipped, and sets the status for the entire configured item to Awaiting Fulfillment.

3. Order Management sets the status of the part of the configured item that it created for the remaining quantity, and the order lines that this part of the configured item references, to Awaiting Fulfillment.
4. eBusiness Suite sets the status of the shippable lines to Shipped and the status of the configured item to Awaiting Fulfillment when it ships the rest of the configured item.
5. The flow in eBusiness Suite reaches the Fulfillment activity, and then creates another event that sets the status to Fulfilled. Lines for each configure option and nonshippable lines remain in the Awaiting Fulfillment status until the eBusiness Suite flow reaches this step.
6. Order Management processes the event and sets the status of the order lines that the configured item references to Fulfilled.

Note

- If an assemble-to-order split happens in eBusiness Suite, then the connector ignores the configured item.
- If one of your users splits an order line in eBusiness Suite, then the order line in Order Management remains in status Awaiting Fulfillment. eBusiness Suite sets the status of the shippable lines to Shipped when it ships each line, and it leaves the nonshippable order lines in status Awaiting Fulfillment.

  This integration doesn't synchronize status Closed in eBusiness Suite with the status in Order Management. It ignores status Closed from eBusiness Suite. Instead, it sets the status of the order lines in Order Management to Closed when the process finishes all steps.

Create a Return Material Authorization

This integration can send a return material authorization (RMA) to eBusiness Suite for receiving. eBusiness Suite fulfills it, then sends the Return statuses to Order Management.

Return an Item

Here's the sequence that this integration uses to return an item.

1. Order Management sends the return lines for the return material authorization to eBusiness Suite. The integration sets the status in eBusiness Suite to Awaiting Return and the status in Order Management to Awaiting Fulfillment.
2. eBusiness Suite receives the return line for the return material authorization, sets the status to Returned, then sets the status in Order Management to Received.
3. The flow in eBusiness Suite reaches the Fulfillment activity, and then sets the status in eBusiness Suite to Fulfilled and the status in Order Management to Fulfilled. The integration doesn't update Order Management with the intermediate statuses that happen in eBusiness Suite during order fulfillment, such as Awaiting Return Disposition.

Return a Configured Item

Here's the sequence that this integration uses to return a configured item.

1. Order Management sends the configured item and a reference to the original eBusiness Suite configured item to eBusiness Suite. The return lines in Order Management are in status Awaiting Fulfillment.
2. eBusiness Suite expands the order lines in the return configured item according to the order lines that the original configured item references. The configured item in eBusiness Suite isn't in status Returned because eBusiness Suite doesn't receive the configured item.
3. The integration creates the return material authorization and sets the status of the configured item to Awaiting Fulfillment.

4. One of your users or an automated process returns the order lines that the configured item references in eBusiness Suite.
5. eBusiness Suite sends the updated statuses to Order Management.
6. Order Management updates the status to Received, then Fulfilled, then Closed.

Process a Hold

Process a Hold That Your User Creates

Here's the sequence that this integration uses to process a hold that one of your users creates.

1. One of your users creates a hold in Order Management.
2. Order Management sends the hold to eBusiness Suite.
   This integration can send only one hold at a time from Order Management to eBusiness Suite because it can't send the Hold Name to eBusiness Suite.
3. eBusiness Suite applies only the DOO O2C Change Management Hold to the sales order in eBusiness Suite. eBusiness Suite ignores any subsequent hold that Order Management sends. Note that DOO is an abbreviation for distributed order orchestration, which is an earlier version of Order Management, and O2C is an abbreviation for Order To Cash.
4. Your user releases the hold in Order Management.
5. Order Management sends the release to eBusiness Suite.
6. eBusiness Suite releases DOO O2C Change Management Hold.

Process a Hold That Change Management Creates

Here's the sequence that this integration uses to process a hold that change management creates.

1. Change management revises a sales order in Order Management.
2. Change management creates a hold in Order Management.
3. Order Management sends the hold to eBusiness Suite to stop eBusiness Suite from processing the sales order.
4. eBusiness Suite attempts to hold the sales order. If successful, then eBusiness Suite sends a reply to Order Management that it successfully placed the hold.
   If eBusiness Suite can't hold the sales order, then it sends a reply to Order Management that the hold failed, and Order Management doesn't allow the change. For example, if eBusiness Suite already scheduled the sales order for shipping, it might not be able to change it.
5. If eBusiness Suite successfully places the hold, then Order Management makes the change, releases the hold, then sends the release to eBusiness Suite.
6. eBusiness Suite releases the hold.
   eBusiness Suite can't determine whether one of your users created the hold or change management created the hold, so eBusiness Suite releases all holds for this sales order.

Schedule a Sales Order

Process a Sales Order That Order Management Schedules

Here's the sequence that this integration uses to process a sales order that Order Management schedules.

1. Order Management schedules the sales order, sets values for the Scheduled Ship Date and Scheduled Arrival Date, then sends these attributes to eBusiness Suite.
2. eBusiness Suite sends these attributes to eBusiness Suite Shipping.
   eBusiness Suite can't and doesn't reschedule these order lines.

Process a Sales Order That eBusiness Suite Schedules

Here's the sequence that this integration uses to process a sales order that eBusiness Suite schedules.

1. Order Management doesn't schedule the sales order, and it sends the Scheduled Ship Date and Scheduled Arrival Date to eBusiness Suite with empty values.
2. eBusiness Suite schedules the order lines, then uses the Schedule Ship Date Change event to send schedule details to Order Management. The order line status in Order Management remains at Awaiting Shipping.

Using Global Order Promising

Global Order Promising uses attributes from the sales order to schedule the sales order.

- Schedule Ship Date
- Schedule Arrival Date
- Ship From Warehouse
- Shipping Method

Processing is different depending who calls Global Order Promising.

Who Calls Global Order Promising	Description
Order Management	Order Management sends attributes to eBusiness Suite, and eBusiness Suite doesn't reschedule the sales order.
eBusiness Suite	eBusiness Suite sends attributes to Order Management, and Order Management doesn't reschedule the sales order.

Flow from eBusiness Suite to Order Management

This integration supports flows from eBusiness Suite to Order Management.

- Update an order line status.
- Update the status for an item, a configured item that uses pick-to-order, a configured item that uses assemble-to-order, a kit, or a return material authorization.
- Update the Scheduled Ship Date, Scheduled Arrival Date, Warehouse, and Shipping Method.
- Split an order line for.
    - Item
    - Configured item that uses pick-to-order
    - Configured item that uses assemble-to-order
    - Kit

  This split supports partial shipping or supports an action that one of your users does in eBusiness Suite.

eBusiness Suite uses business events to synchronize updates to Order Management.

Split a Return Material Authorization

Here's the sequence that this integration uses to split a return material authorization.

1. One of your users does a partial receipt, not a delivery, in eBusiness Suite.

2. eBusiness Suite receives the partial order line.
3. The eBusiness Suite user does a deliver transaction.
4. eBusiness Suite splits the order line for the return material authorization into two order lines, and then sets the status of the received line to Returned and the status of the new line to Awaiting Return.
5. eBusiness Suite sends the split event to Order Management.
6. Order Management splits the order line into two order lines, and then sets the status of the received line to Received and the status of the new line to Awaiting Fulfillment.

Split Return Material Authorization for a Configured Item

Order Management doesn't support splitting a return material authorization that includes a configured item. It processes status updates only for the configured item, and not for individual order lines that the configured item references. Instead, here's the sequence that this integration uses.

1. Order Management waits for eBusiness Suite to fulfill the entire configured item.
2. eBusiness Suite fulfills the entire configured item, then sends an update to Order Management.
3. Order Management sets the status for the configured item to Fulfilled.

   For example, if the ordered quantity is 10, and if the quantity returned is 5, then Order Management sets the status for 5 of the order lines in the configured item to Awaiting Return, and will process the configured item only after it receives the remaining 5 lines.

Map Status When Splitting a Return Material Authorization

Here are the statuses that this integration maps between Order Management and eBusiness Suite when it splits a return material authorization.

Context	Order Management Status	eBusiness Suite Status
Create a return order.	Awaiting Fulfillment	Awaiting Return
Receive or deliver.	Received	Returned
Receive or deliver part of a configured item.	Fulfilled	Closed
Finish delivery.	Closed	Closed

Related Topics

- Integrate eBusiness Suite With Order Management
- Integrate Order Management with eBusiness Suite
- Overview of Integrating Order Management with eBusiness Suite
- Connect SOA Server to Order Management and eBusiness Suite

Flexfields

Overview of Using Flexfields to Integrate Order Management with Other Oracle Applications

Use a descriptive flexfield to integrate Order Management with Oracle Receivables, Oracle Shipping, Oracle Receiving, or Oracle Procurement.

You can send each value that an extensible flexfield uses to an upstream source system or to a downstream fulfillment system. You use a web service payload to send the value.

Guidelines

Consider these guidelines.

- Oracle Receivables comes predefined with a separate entity for each descriptive flexfield. You don't have to use separate entities for descriptive flexfields in Shipping, Receiving or Procurement because their predefined entities already have the descriptive flexfields you need.
- You make attributes available in the main entity.
- You must make sure your set up only references attributes that each application supports. For a list of attributes, see *Entities and Attributes That You Can Use When You Integrate Order Management with Other Oracle Applications*.
- You must reference the database code that identifies each attribute, such as Attribute1 or `Attributedate2`. You must not use the implemented names to reference the attributes.
- If you reference a descriptive flexfield according to context, then you must make sure your set up populates AttributeCategory. If you use a global segment, then it isn't necessary to populate AttributeCategory.
- If your flexfield is on a fulfillment line, then use only one context for each entity. Your set up must not attempt to process two different contexts at the same time. For example, if you need to process two attributes, then add two segments to the same context. Add segment x and segment y to context A. Don't add segment x to context A and segment y to context B.

Automatically Map Extensible Flexfield

Each interaction uses a service data object (SDO). You must use a specific set of web services. For details, see *Web Services That You Can Use to Integrate Order Management*.

You can use the Copy-of feature in XLST (Extensible Stylesheet Language Transformations) instead of doing this work manually. This feature dynamically maps the input extensible flexfields to the Order Management entities and maps the extensible flexfield entity in Order Management to the SDO or modified node.

Here's where you can use this feature in Order Management.

- Sales order integration
- Template tasks
- Fulfillment tasks

You must manually map other task types.

Example Payloads

You can examine an example payload that uses a flexfield to send custom data to your downstream fulfillment system. Go to *Technical Reference for Order Management (Doc ID 2051639.1)*, download the Payloads and Files attachment, then see the Other section in the attachment.

Here's part of an example orderImportService payload that uses a flexfield to import custom data from an upstream source system. For the entire payload, see the attachment in Doc ID 2051639.1.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders
 xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request
 xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:Order>
 <ns2:SourceTransactionIdentifier>SN_201117_Desktop_02</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionNumber>SN_1811_01_Desktop_02</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2020-11-08T13:50:50.0340</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:RequestingLegalUnitIdentifier>204</ns2:RequestingLegalUnitIdentifier>
 <ns2:FreezePriceFlag>false</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>false</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>false</ns2:FreezeTaxFlag>
 <ns2:CustomerPONumber>SNVALE</ns2:CustomerPONumber>
 <ns2:Line>
 . . .
 <ns2:AdditionalFulfillmentLineInformationCategories
 xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate"
 xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineCategories/"
 xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/"
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <!--Single Context(DOO_FULFILL_LINES_ADD_INFO)-->
 <ns12:FulfillLineEffBPackShipInstructionprivateVO>
 <ns8:ContextCode>PackShipInstruction</ns8:ContextCode>
 <ns22:_PackingInstruction>Place bubble wrap on
 top and at the bottom</ns22:_PackingInstruction>
 <ns22:_ShippingInstruction>SOMETHINGRIGHT</ns22:_ShippingInstruction>
 <ns22:_ShippingCost>1999</ns22:_ShippingCost>
 <ns22:_NeedbyDate>2020-12-01T00:00:00</ns22:_NeedbyDate>
 <ns22:_PickDate>2020-12-01T00:00:00</ns22:_PickDate>
 </ns12:FulfillLineEffBPackShipInstructionprivateVO>
 </ns2:AdditionalFulfillmentLineInformationCategories>
 </ns2:Line>
 . . .
 <ns2:AdditionalHeaderInformationCategories
 xsi:type="ns12:j_HeaderEffDooHeadersAddInfoprivate"
 xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerCategories/"
 xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerContextsB/"
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ns8:Category>DOO_HEADERS_ADD_INFO</ns8:Category>
 <ns12:HeaderEffBComplianceDetailsprivateVO>
 <ns8:ContextCode>ComplianceDetails</ns8:ContextCode>
 <ns22:_ComplianceInfo>This order complies with the
 all regulations</ns22:_ComplianceInfo>
 <ns22:_ComplianceReason>ASLDFSome compliance reason</ns22:_ComplianceReason>
 <ns22:_ComplianceDate>2015-07-02T10:10:10</ns22:_ComplianceDate>
 <ns22:_CompleteCompliancedate>2015-07-01T10:10:10</ns22:_CompleteCompliancedate>
```

```
<ns22:_ComplianceValue>12345</ns22:_ComplianceValue>
</ns12:HeaderEffBComplianceDetailsprivateVO>
<ns12:HeaderEffBHeaderContext1privateVO>
<ns8:ContextCode>HeaderContext1</ns8:ContextCode>
<ns22:_H1AttributeChar1>3zVendorrelocationtoUSy</ns22:_H1AttributeChar1>
<ns22:_H1AttributeChar2>ABC - xyz</ns22:_H1AttributeChar2>
<ns22:_H1AttributeNum1>1999</ns22:_H1AttributeNum1>
<ns22:_H1AttributeDateTime1>2015-01-01T10:10:10</ns22:_H1AttributeDateTime1>
</ns12:HeaderEffBHeaderContext1privateVO>
<ns12:HeaderEffBEND_5FUSE_5FPARTYprivateVO>
<ns2:ContextCode>END_USE_PARTY</ns2:ContextCode>
<ns22:endUserCustomerName>Attr 1 and 3</ns22:endUserCustomerName>
<ns22:endUserAddress>4405</ns22:endUserAddress>
<ns22:endUserCustomer>1289</ns22:endUserCustomer>
<ns22:endUserAddress2>4420</ns22:endUserAddress2>
</ns12:HeaderEffBEND_5FUSE_5FPARTYprivateVO>
</ns2:AdditionalHeaderInformationCategories>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Related Topics

- Overview of Integrating Order Management with Accounts Receivable
- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Web Services That You Can Use to Integrate Order Management
- Overview of Using Extensible Flexfields in Order Management

Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications

Use a descriptive flexfield to integrate Order Management with Oracle Receivables, Shipping, Receiving, or Procurement.

This example describes how to integrate Order Management with Oracle Receivables. Integration with Shipping, Receiving, or Procurement is similar.

In this example, you use a descriptive flexfield named Invoice Lines to integrate Order Management with Oracle Receivables.

Summary of the Setup

1. Get values that identify the descriptive flexfield.
2. Modify the service mapping.

This topic uses example values. You might need different values, depending on your business requirements.

Get Values That Identify the Descriptive Flexfield

1. Enable the Invoice option in the Enable Custom Payloads for Downstream Integration feature.
   For details, see *Get Started with Integrating Order Management with Other Oracle Applications*.
   Use a spreadsheet editor, such as Microsoft Excel, to create a spreadsheet that contains columns and values.

Attribute	Source	Value
Namespace	targetNamespace attribute from the XSD file.	-
`Xsitype`	complexType attribute from the XSD file.	-
Context Code	DefaultValue property of the FLEX_Context view attribute from the XML file.	-
`Segmentname`	Name of the view attribute from the XML file.	-

   The Source column describes the source you will use to get the value you will enter in the Value column. Leave the Value column empty for now. You will add values to it during this procedure.
2. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Financials
   - Functional Area: Receivables
   - Task: Manage Receivables Descriptive Flexfields
     The task name is different for other Oracle Applications. For example, if you're integrating with Purchasing, then go to the Manage Purchasing Descriptive Flexfields task instead.
3. On the Manage Receivables Descriptive Flexfields page, search for the value.

Attribute	Value
Name	Invoice Lines

4. In the search results, click the **row** that contains the value.

Attribute	Value
Name	Invoice Lines

5. Download the archive and open it for editing.
    - Click **Actions > Download Flexfield Archive**.
    - In the Confirmation dialog, wait for the archive to finish, click **Download**, then save the file to your local computer.

      The name of this file will include the value of the Flexfield Code attribute. For this example, the file name is 222_RA_CUSTOMER_TRX_LINES.zip.
    - Use Windows Explorer to navigate to 222_RA_CUSTOMER_TRX_LINES.zip.
    - Expand `C:\Users\user_name\Downloads\222_RA_CUSTOMER_TRX_LINES.zip\oracle\apps\flex\financials\receivables\transactions\autoInvoices\TransactionLineDff\view`.
    - Use an editor, such as Notepad++ or an XML editor, to open one of the files.

If the Flexfield That You Define in Receivables	Then You Should Open
Includes a context.	RACUSTOMERTRXLINESCon1.xsd This folder contains several XSD files. You open the XSD file that contains the context code you use for this example, which is RA_CUSTOMER_TRX_LINES_Con1.
Doesn't include a context and you set it up it globally.	TransactionLineFLEX.xsd Note the values of targetNamespace and complexType. If you don't set up a context, then the DefaultValue attribute is empty, and _FLEX_Context is empty in the Sources tab of the service mapping. If you set up the context for only one attribute in the flexfield, then you can use DefaultValue of this attribute when you specify _FLEX_Context in the VO.xml file.

6. Copy values to your spreadsheet, then close the editor. Make sure you include the single quotation marks (') in each value.

Attribute	Value
targetNamespace	'http://xmlns.oracle.com/apps/financials/receivables/transactions/shared/model/flex/TransactionLineDff/'
complexType	If you. - Set up a context, then use 'RACUSTOMERTRXLINESCon1' - Didn't set up a context, then use 'TransactionLineFLEX'

Attribute	Value

For example, the RACUSTOMERTRXLINESCon1.xsd file contains the attributes on lines 2 and 4.

7. In Windows Explorer, expand **View**, then use an XML editor to open the file that includes the public view object with the context you need.

   For this example, open TransactionLineFLEXRA_5FCUSTOMER_5FTRX_5FLINES_5FCon1VO.xml.
8. Copy the DefaultValue property of the FLEX_Context view attribute to your spreadsheet.

   In this example, the default value is RA_CUSTOMER_TRX_LINES_Con1.

9. Identify the name of the view attribute you must map, then copy it to your spreadsheet.

   In this example, search the XML file for a view attribute that contains RACUSTOMERTRXLINESSSeq1.

10. Manage the context.
    - On the Edit Descriptive Flexfield page, click **Manage Contexts**.
    - On the Manage Contexts page, search for the value.

Attribute	Value
Context Code	'RA_CUSTOMER_TRX_LINES_Con1'

- In the search results, click **Actions > Edit**.
- On the Edit Context page, copy the value to your spreadsheet.

Attribute	Value
API Name	RACUSTOMERTRXLINESCon1

- Cancel pages until you're back on the Setup and Maintenance page.

11. Verify that your spreadsheet contains the values.

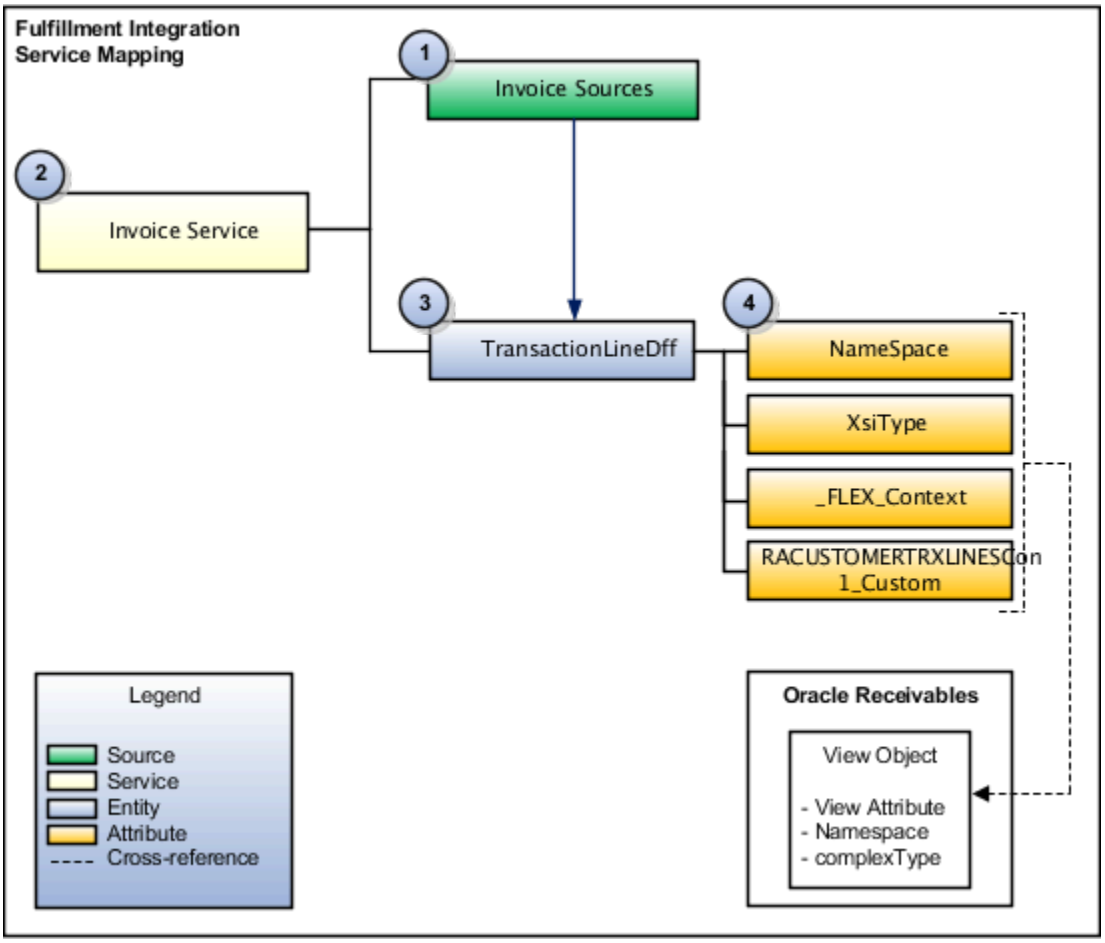
Attribute	Source	Value
Namespace	The targetNamespace attribute from the XSD file.	'http://xmlns.oracle.com/apps/financials/receivables/transactions/shared/model/flex/TransactionLineDff/'
`Xsitype`	The complexType attribute from the XSD file.	'RACUSTOMERTRXLINESCon1'
Context Code	The DefaultValue property of the FLEX_Context view attribute from the XML file.	'RA_CUSTOMER_TRX_LINES_Con1'
`Segmentname`	The name of the View attribute from the XML file.	RACUSTOMERTRXLINESSSeg1

You must enclose each string value with single quotation marks.

Modify the Service Mapping

You will modify a service mapping that implements the service data object.

Here's the service data object.



This SDO (service data object) includes objects.

1. **Source**. Provides a structure so Order Management can model data in the input SDO (service data object).

   In this example, you will do these steps.

   - Modify the InvoiceSources source on the predefined FulfillmentIntegration service mapping. You will map this source to the TransactionLineDff entity. This entity comes predefined with attributes NameSpace, XsiType, and _FLEX_Context.
   - Add the RACUSTOMERTRXLINESCon1_Custom attribute. The FulfillmentIntegration context comes predefined with the descriptive flexfields and global descriptive flexfields that Order Management supports.
2. **Service**. Requests the service mapping and receives the output SDO. In this example, the InvoiceService service references the entities and attributes that the integration algorithm uses to get the value of the descriptive flexfield.
3. **Entity**. The entity that the service mapping requires to structure the output SDO. In this example, you modify the TransactionLineDff entity so it references the descriptive flexfield.
4. **Attributes**. The attributes that the service mapping requires to structure the output SDO. You will set up these attributes so they reference objects and properties in Oracle Receivables.

For details about the service data object, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

Modify the service mapping.

1. Make sure you have the privileges that you need to administer Order Management.

   Use these privileges so you can access the integration algorithm and service mappings that you use in the Pricing Administration work area to set up the integration.
2. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Go to the Pricing Administration work area.
4. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
5. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.
6. On the Edit Service Mappings page, on the Entities tab, click the **row** that contains the value.

Attribute	Value
Entity	TransactionLineDff

7. Click **Sources**.
8. In the InvoiceSources Details area, on the Entity Mappings tab, click the **row** that contains the value.

Attribute	Value
Entity	TransactionLineDff

   Include only the attributes that you want. For example, the integration uses the POLineNumber attribute to populate the ATTRIBUTE_CATEGORY column of the RA_INTERFACE_LINE_ALL table. If you don't want any values in ATTRIBUTE_CATEGORY, then make sure the Entity Mappings tab doesn't contain the POLineNumber attribute.
9. On the Attribute Mappings tab, add expressions, then click **Save**.

Attribute	Expression
NameSpace	"http://xmlns.oracle.com/apps/financials/receivables/publicFlex/TransactionLineDff/"
XsiType	"RACUSTOMERTRXLINESCon1"
RACUSTOMERTRXLINESCon1_Custom	Leave empty.

   Copy values from your spreadsheet into the expression. You must use one set of double quotation marks (") to enclose each string that you define in the expression.

10. Click **Services**, then click the **row** that contains the value.

Attribute	Value
Entity	TransactionLineDff

11. In the TransactionLineDff Entities area, add a row for each entity.
    - _FLEX_Context
    - XsiType
    - NameSpace
    - RACUSTOMERTRXLINESCon1_Custom

    Note
    - Make sure the Read attribute and the Write attribute each contain a check mark in each row.
    - You must add a value in the Alias attribute in each row, and the alias must begins with an upper case character, such as A. Don't begin the alias with a lower case character, such as a.
12. Click **Save and Close**.

Related Topics
- Get Started with Integrating Order Management with Other Oracle Applications
- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Pricing Algorithms
- Service Mapping
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together

Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications

Use an extensible flexfield to integrate Order Management with Oracle Receivables, Oracle Shipping, Oracle Receiving, or Oracle Procurement.

Assume you must make sure the primary salesperson is the same on the order header in Order Management and on the line in Oracle Receivables.

Assume you implemented a context and segment for the header extensible flexfield.

- HeaderContext as the context
- `H1AttributeNum` as the segment

You will you add an extensible flexfield on the order header in the service mapping, store the Id of the primary salesperson in the extensible flexfield, then map it to the Id of the primary salesperson on the Accounts Receivable line.

Summary of the Setup

1. Get values that identify the extensible flexfield.
2. Modify the service mapping.
3. Create an integration algorithm.

This topic uses example values. You might need different values, depending on your business requirements.

Get Values That Identify the Extensible Flexfield

1. Use a spreadsheet editor, such as Microsoft Excel, to create a spreadsheet that contains columns and values.

Attribute	Source	Value
ViewObject	XML file.	`HeaderEffBHeaderContext1privateVO`
ViewAttribute	XML file.	`_H1AttributeNum1`

2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields
3. On the Manage Order Extensible Flexfields page, search for the value.

Attribute	Value
Module	Process Order

4. In the search results, click the **row** that contains the value.

Attribute	Value
Name	Header Information

5. Click **Actions > Download Flexfield Archive**
6. In the Confirmation dialog, wait for the archive to finish, click **Download**, then save the file to your local computer.

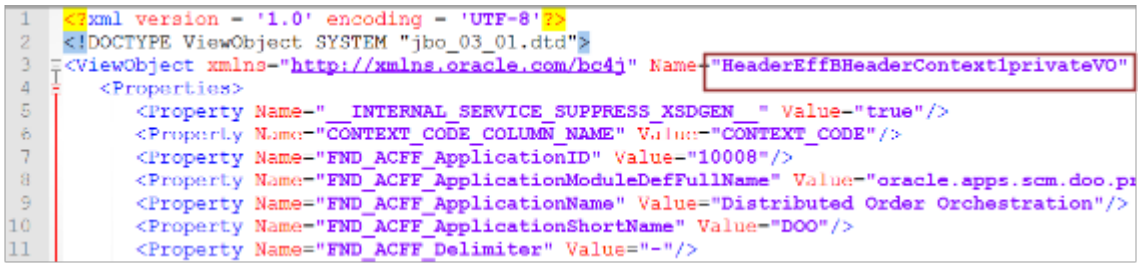
    The name of this file will include the value that the Flexfield Code attribute contains. For this example, you're interested in an attribute that resides on the order header, so the file name you need is 10008_DOO_HEADERS_ADD_INFO.zip.
7. Use Windows Explorer to navigate to 10008_DOO_HEADERS_ADD_INFO.zip.
8. Expand `10008_DOO_HEADERS_ADD_INFO.zip > Oracle Apps > SCM > DOO > processOrder > Flex > headerContextsB > View`, then use an editor, such as Notepad++ or an XML editor, to open HeaderEffBHeaderContext1privateVO.xml.

    This view lists the XML files that are available for each context.

    You will examine the HeaderContext1 extensible flexfield context.
9. Copy the value to your spreadsheet.

Attribute	Value
Name of ViewObject	`HeaderEffBHeaderContext1privateVO`

For example, the HeaderEffBHeaderContext1privateVO.xml file contains view object name.

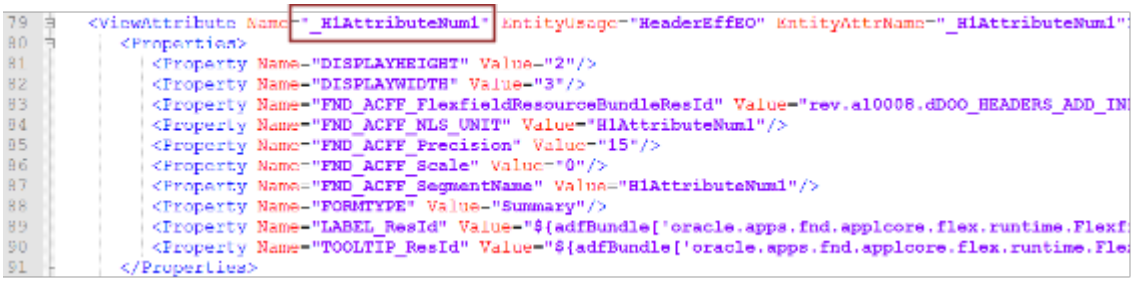


10. Copy the name of the attribute you will use to store the Id of the primary salesperson to your spreadsheet.

    Look for the ViewAttribute that matches the segment you're using to store the value you need. For this example, here's the value you copy.

Attribute	Value
Name of ViewAttribute	`_H1AttributeNum1`

For example, the HeaderEffBHeaderContext1privateVO.xml file contains the view attribute name.



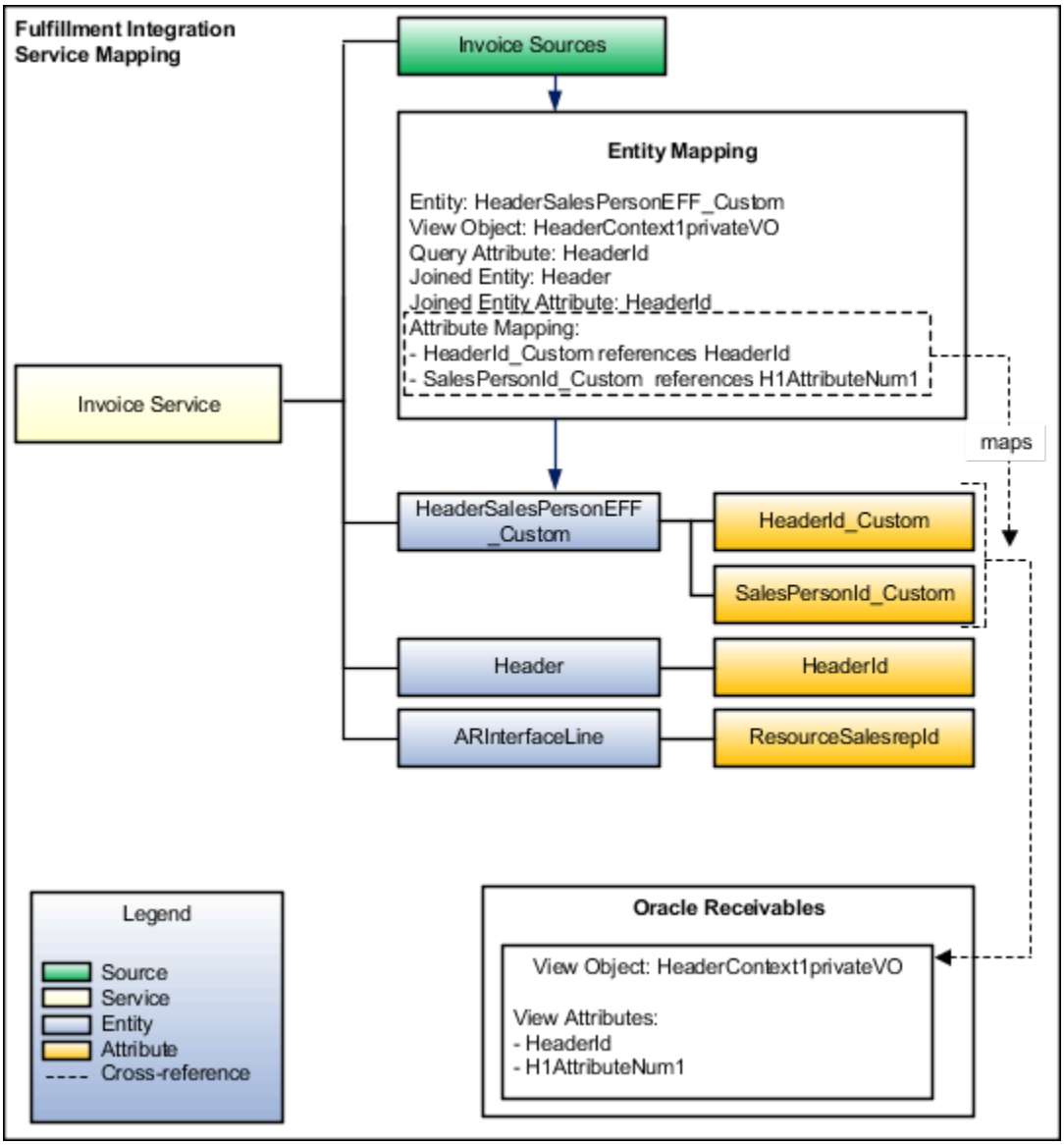
11. Verify that your spreadsheet contains values.

Attribute	Source	Value
ViewObject	XML file.	`HeaderEffBHeaderContext1privateVO`
ViewAttribute	XML file.	`_H1AttributeNum1`

Modify the Service Mapping

You will modify the predefined FulfillmentIntegration service mapping.

Here's the modification.



Try it.

1. Make sure you have the privileges that you need to administer Order Management.

   Use these privileges to access the integration algorithm and service mappings you use in the Pricing Administration work area to set up the integration.
2. Go to the Pricing Administration work area.

   Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
3. On the Manage Service Mappings page, in the Name column, click **FulfillmentIntegration**.

4. On the Edit Service Mappings page, on the Entities tab, click **Actions > Add Row**, then set the values.

Attribute	Value
Entity	HeaderSalesPersonEFF_Custom
Description	Context for an extensible flexfield on the order header. It contains details about the salesperson.

5. In the Details area, add attributes, then click **Save**.

Attribute	Type
HeaderId_Custom	Long
SalesPersonId_Custom	Long

If you use a descriptive flexfield as part of this solution, then you must specify the same Type that you specify for the attribute in the extensible flexfield. For example, if you use SalesPersonId of type Long in the descriptive flexfield, then you must also specify SalesPersonId as Long on the extensible flexfield. In a different scenario, if you use an attribute named packingInstructions of type String on your descriptive flexfield, then you must also set packingInstructions to String on your extensible flexfield. If you don't use the same data type, then your service mapping might not fail when compiling but will fail in your runtime environment. The runtime failure might be difficult to troubleshoot because you might not encounter any errors but your set up might not work as you expected.

6. Set up the sources.
    - Click **Sources**.
    - In the InvoiceSources Details area, on the Entity Mappings tab, click **View > Columns**, then add a check mark to each value.
        - Joined Entity
        - Joined Entity Attribute
        - Use Existing View Object
    - Click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	HeaderSalesPersonEFF_Custom
Type	View Object
View Object	`HeaderEffBHeaderContext1privateVO`

Attribute	Value
Query Type	Join
Query Attribute	HeaderId
Use Existing View Object	Contains a check mark.
Joined Entity	Header It might be necessary to click Save to get the list of values to populate.
Joined Entity Attribute	HeaderId

- On the Attribute Mappings tab, add attributes, then click **Save**.

Attribute	View Object Attribute
HeaderId_Custom	HeaderId
SalesPersonId_Custom	`_H1AttributeNum1` Copy the value from your spreadsheet.

The attribute View Object Attribute is required for each mapping you create in this topic.

- On the Entity Mappings tab, click the **row** that contains the value.

Attribute	View Object Attribute
Entity	ARInterfaceLine

- In the ARInterfaceLine Details area, click **Actions > Add Row**, set the value, then click **Save**.

Attribute	Value
Attribute	ResourceSalesrepId

7. Set up the services.
    - Click **Services**.
    - In the InvoiceService Details area, click **Actions > Add Row**, then set the value.

Attribute	Value
Entity	HeaderSalesPersonEFF_Custom

    - In the HeaderSalesPersonEFF_Custom - Entities area, add attributes.

Attribute	Alias
HeaderId_Custom	Optional
SalesPersonId_Custom	Optional.

      The alias is optional. You can add a value or leave it empty. If you add an alias, make sure it begins with an upper case character, such as My Alias. Don't begin the alias with a lower case character, such as my alias.
    - In the InvoiceService Details area, click the **row** that contains the value.

Attribute	Value
Entity	Header

    - In the Header Entities area, add an attribute.

Attribute	Alias
HeaderId	Leave empty.

    - In the InvoiceService Details area, click the **row** that contains the value.

Attribute	Value
Entity	ARInterfaceLine

Attribute	Value

- In the ARInterfaceLine Entities area, add the attribute, then click **Save and Close**.

Attribute	Alias
ResourceSalesrepId	Leave empty.

Create an Integration Algorithm

1. Click **Tasks**, then, under Order Management Configuration, click **Manage Algorithms**.
2. On the Manage Algorithms page, click **Actions > Create**.
3. On the Create Algorithm page, set the value, then click **Save**.

Attribute	Value
Name	Accounts Receivable Mapping Custom

4. Click **Variables > Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Name	ARIntegration
Data Type	Data Object
Input and Output	Input and Output
Internal Service Schema	FulfillmentIntegration.InvoiceService

5. Click **Algorithm > Add Step > Conditional Action**, set the values, then click **Save**.

Attribute	Value
Name	Map Salesperson Id
Description	Map the primary salesperson from the extensible flexfield that resides on the order header.

6. In the Data Sets area, add the data sets, then click **Save**.

Name	Variable Path	Cardinality	Data Set Join
ARLine	ARIntegration.ARInterfaceLine	Leave empty	Not applicable
FLine	ARIntegration.FulfillLine	Zero or one	`[FulfillLineId: {ARLine.FulfillLineId}]`
HdrEFF	ARIntegration.HeaderSalesPersonEFF_Custom	Zero or one	`[HeaderId_Custom: {FLine. HeaderId}]`

where

Code	Description
ARIntegration.ARInterfaceLine	Specifies to use entity ARInterfaceLine of the service mapping to get order details, then store them in ARIntegration, which is the variable you added to this integration algorithm earlier in this topic. This integration uses ARIntegration to temporarily store the data that this integration algorithm will communicate through the service mapping. The Data Set Join is empty in primary data set ARLine because the secondary data set populates ARLine.
ARIntegration.FulfillLine	Filters the results in ARIntegration according to the fulfillment line. The join `[FulfillLineId: {ARLine.FulfillLineId}]` uses the FulfillLineId attribute of the FulfillLine entity from the service mapping to filter the primary ARLine data set so it contains only the fulfillment line that FulfillLineId references.
ARIntegration.HeaderSalesPersonEFF_Custom	Filters the results in ARIntegration according to order header. The join `[HeaderId_Custom: {FLine. HeaderId}]` uses the HeaderId attribute of the HeaderSalesPersonEFF_Custom entity to filter the **`FLine`** data set so it contains only order headers.

Note

- In the Variable Path, make sure you use the exact entity names from the service mapping, such as ARInterfaceLine, FulfillLine, and HeaderSalesPersonEFF_Custom.
- To make ARLine the primary, enable the Primary option.
- Each group of data sets includes one primary data set and one or more secondary data sets.

- Cardinality specifies that there is zero or one records in the secondary data set to one or many records in the primary data set. It determines whether the join is an inner join or outer join.
- Data Set Join defines the constraint that this step uses to filter records that the secondary data set saves in the primary.
- FLine represents a row from entity FulffillLine that matches the join condition. The name is case-sensitive. Make sure you use FLine, not fline.
- HdrEFF represents a row from entity HeaderSalesPersonEFF_Custom that matches the join condition.
- ARInterfaceLine and FufillLine come predefined.
- You create HeaderSalesPersonEFF_Custom in the service mapping.
- ARInterfaceLine comes predefined with attribute ResourceSalesRepId. Your set up will populate ResourceSalesRepId with the value from attribute SalesPerson_Custom in entity HeaderSalesPersonEFF_Custom that you define. You define HeaderSalesPersonEFF_Custom in the join conditions for the data set so it uniquely identifies the row that your set up uses to do the mapping.

7. In the Execute Condition area, click **Add Condition > Default Action**, add the value, then click **Save and Close**.

Attribute	Value
Action	`ARLine.ResourceSalesrepId=HdrEFF.SalesPerson_Custom`

where

- **ARLine.ResourceSalesrepId**. Specifies to store the result in the ResourceSalesrepId attribute of the ARInterfaceLine entity on the service mapping.

8. Test your algorithm. For details, see *Set Up an Integration Algorithm*.
9. On the Manage Algorithms page, click **Actions > Publish**.
10. Verify that the Status displays Published, then click **Done**.
11. Navigate to the Edit Service Mapping page.
12. Click **Services**, set the values, then click **Save and Close**.

Attribute	Value
Service	InvoiceService
Implementation Type	Algorithm
Implementation	Accounts Receivable Mapping Custom

where

- Accounts Receivable Mapping Custom is the name of the integration algorithm you created.

What If I Can't Find an Attribute I Can Use

You can't add an attribute to the data model, but you can reference an attribute that you aren't using. Most applications come predefined with a set of attributes you can use for your own, specific purpose, such as string attributes Attribute1 through Attribute20 with drop ship, or string attributes SrcAttribute1 through SrcAttribute20 in shipping. Get a list of the attributes. For details, see *Entities and Attributes That You Can Use When You Integrate Order Management with Other Oracle Applications*.

Related Topics

- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Overview of Integrating Order Management with Accounts Receivable
- Set Up an Integration Algorithm
- Create Your Service Mapping

Map Extensible Flexfields to Descriptive Flexfields

Use a service mapping to map an extensible flexfield to a descriptive flexfield on the purchase requisition header in Procurement.

1. Make sure you have the privileges that you need to administer Order Management.
2. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Go to the Pricing Administration work area.
4. Click **Tasks**, then, under Order Management Configuration, click **Manage Service Mappings**.
5. On the Manage Service Mappings page, click **FulfillmentIntegration**.
6. On the Edit Service Mapping page, click **Sources**, then, in the Source column, click **PurchaseRequestSource**.
7. In the Details area, on the Entity Mappings tab, in the Entity column, click **PurchaseRequestHeader**.
8. On the Attribute Mappings tab, click **Actions > Add Row**, then set the values.

Attribute	View Object Attribute
AttributeCategory	RequestingBusinessUnitName

The rest of the setup is similar to using a descriptive flexfield. For details see, Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications.

Related Topics

- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Create a Sandbox So You Can Edit Service Mappings

Reference

Entities and Attributes That You Can Use When You Integrate Order Management with Other Oracle Applications

Use a variety of entities and attributes when you integrate Order Management with some other Oracle application.

Summary

Here's a summary of the entities, sources, and services you can use in a service mapping or integration algorithm.

Object	Value
Common	Header FulfillLine FilfillLineDetails
ShipmentSource	ShipmentRequestHeader ShipmentRequestLine
InvoiceSource	ARChargeInterfaceLine ARInterfaceLine Charge ChargeComponent TransactionInterfaceGdf TransactionInterfaceHeaderDff TransactionInterfaceLinkToDff TransactionInterfaceLinkToDff TransactionLineInterfaceGdf
PurchaseRequestSource	PurchaseRequestHeader PurchaseRequestLine PurchaseRequestConfig
ReceiptSource	ReceiptAdvice ReceiptAdviceLine ReceiptAdviceLineLot ReceiptAdviceLineLotWithSerials

Entities You Can Use with Other Oracle Applications

Entity	Description
Header	Sales order header.

Entity	Description
FulfillLine	Fulfillment line on the sales order.
FulfillLineDetails	Fulfillment line details on the sales order.
ARInterfaceLine	Account receivables on the interface line.
TransactionInterfaceGdf,	Flexfield that stores receivable transactions that are available globally.
TransactionInterfaceLinkToDff,	Flexfield that stores link-to details for receivables transactions.
TransactionLineDff,	Flexfield that stores receivable invoice lines.
TransactionLineInterfaceGdf,	Flexfield that stores receivable transaction lines that are available globally.
PurchaseRequestHeader	Purchase request header.
PurchaseRequestLine	Purchase request lines.
PurchaseRequestConfig	Details of a configured item for a purchase request line.
TransactionSalesCreditDff	Flexfield that stores details about sales credits.
ShipmentRequestHeader	Shipment header.
ShipmentRequestLine	Shipment line.
ReceiptAdvice	Receipt advice.
ReceiptAdviceLine	Receipt advice line.
ReceiptAdviceLineLot	Lot number details for the receipt advice line.
ReceiptAdviceLineLotWithSerials	Serial number and lot number details for the receipt advice line.
TransactionInterfaceHeaderDff	Descriptive flexfield that stores details about the receivables transaction header.
ARChargeInterfaceLine	Account receivables on the interface line. You use this object primarily for charges, discounts, or freight.
Charge	Charges on the fulfillment line.

Entity	Description
ChargeComponent	Charge components on the fulfillment line.

You can map the attributes that the ARInterfaceLine entity contains.

Attributes You Can Use With Drop Ship

Purchase Request Header

You can use these attributes in the PurchaseRequestHeader element.

Attribute Name	Type
ApproverEmail	string
ApproverId	long
AssessableAmount	decimal
Attribute1	string
Attribute10	string
Attribute11	string
Attribute12	string
Attribute13	string
Attribute14	string
Attribute15	string
Attribute16	string
Attribute17	string
Attribute18	string
Attribute19	string
Attribute2	string
Attribute20	string

Attribute Name	Type
Attribute3	string
Attribute4	string
Attribute5	string
Attribute6	string
Attribute7	string
Attribute8	string
Attribute9	string
AttributeCategory	string
AttributeDate1	date
AttributeDate10	date
AttributeDate2	date
AttributeDate3	date
AttributeDate4	date
AttributeDate5	date
AttributeDate6	date
AttributeDate7	date
AttributeDate8	date
AttributeDate9	date
AttributeNumber1	decimal
AttributeNumber10	decimal
AttributeNumber2	decimal

Attribute Name	Type
AttributeNumber3	decimal
AttributeNumber4	decimal
AttributeNumber5	decimal
AttributeNumber6	decimal
AttributeNumber7	decimal
AttributeNumber8	decimal
AttributeNumber9	decimal
AttributeTimestamp1	dateTime
AttributeTimestamp10	dateTime
AttributeTimestamp2	dateTime
AttributeTimestamp3	dateTime
AttributeTimestamp4	dateTime
AttributeTimestamp5	dateTime
AttributeTimestamp6	dateTime
AttributeTimestamp7	dateTime
AttributeTimestamp8	dateTime
AttributeTimestamp9	dateTime
Description	string
DocumentStatusCode	string
EmergencyPONumber	string
HeaderId	Long

Attribute Name	Type
InterfaceSourceLineId	long
Justification	string
RequisitionNumber	string
SoldToLegalEntityId	long
SoldToLegalEntityName	string
TaxationCountryCode	string
TaxationTerritory	string

You can't use these attributes in the PurchaseRequestHeader element.

Attribute Name	Type
AttachmentCategory	string
AttachmentEntityName	string
AttachmentPrimaryKey1	string
AttachmentPrimaryKey2	string
AttachmentPrimaryKey3	string
AttachmentPrimaryKey4	string
AttachmentPrimaryKey5	string
DocumentFiscalClassification	string
DocumentFiscalClassificationCode	string
ExternallyManagedFlag	boolean
InterfaceHeaderKey	string

Attribute Name	Type
PreparerEmail	string
PreparerId	long
RequisitioningBUId	long
RequisitioningBUName	string

Purchase Request Line

You can use these attributes in the PurchaseRequestLine element.

Attribute Name	Type
Amount	decimal
Attribute1	string
Attribute10	string
Attribute11	string
Attribute12	string
Attribute13	string
Attribute14	string
Attribute15	string
Attribute16	string
Attribute17	string
Attribute18	string
Attribute19	string
Attribute2	string
Attribute20	string

Attribute Name	Type
Attribute3	string
Attribute4	string
Attribute5	string
Attribute6	string
Attribute7	string
Attribute8	string
Attribute9	string
AttributeCategory	string
AttributeDate1	date
AttributeDate10	date
AttributeDate2	date
AttributeDate3	date
AttributeDate4	date
AttributeDate5	date
AttributeDate6	date
AttributeDate7	date
AttributeDate8	date
AttributeDate9	date
AttributeNumber1	decimal
AttributeNumber10	decimal
AttributeNumber2	decimal

Attribute Name	Type
AttributeNumber3	decimal
AttributeNumber4	decimal
AttributeNumber5	decimal
AttributeNumber6	decimal
AttributeNumber7	decimal
AttributeNumber8	decimal
AttributeNumber9	decimal
AttributeTimestamp1	dateTime
AttributeTimestamp10	dateTime
AttributeTimestamp2	dateTime
AttributeTimestamp3	dateTime
AttributeTimestamp4	dateTime
AttributeTimestamp5	dateTime
AttributeTimestamp6	dateTime
AttributeTimestamp7	dateTime
AttributeTimestamp8	dateTime
AttributeTimestamp9	dateTime
AutosourceFlag	boolean
BaseModelNumber	string
BaseModelPrice	decimal
BuyerEmail	string

Attribute Name	Type
BuyerId	long
BuyerName	string
CategoryId	long
CategoryName	string
ChangeActionReason	string
CurrencyCode	string
CustomerItemDescription	string
DaysEarlyReceiptAllowed	decimal
DeliverToCustomerContactEmail	string
DeliverToCustomerContactId	long
DeliverToCustomerContactName	string
DeliverToCustomerContactNumber	string
DeliverToCustomerId	long
DeliverToCustomerLocationCLLICode	string
DeliverToCustomerLocationOriginalSystem	string
DeliverToCustomerName	string
DeliverToCustomerNumber	string
DeliverToLocationCode	string
DeliverToLocationId	long
DestinationSubinventory	string
FulfillLineId	Long

Attribute Name	Type
GroupCode	string
HazardClass	string
HazardClassId	long
ItemDescription	string
ItemRevision	string
LineType	string
LineTypeId	long
NegotiatedByPreparerFlag	boolean
NegotiationRequiredFlag	boolean
NewSupplierFlag	boolean
NoteToBuyer	string
NoteToReceiver	string
NoteToSupplier	string
Price	decimal
ProcurementBUId	long
ProcurementBUName	string
Rate	decimal
RateDate	date
RateType	string
RequesterEmail	string
RequesterId	long

Attribute Name	Type
SecondaryQuantity	decimal
SecondaryUOM	string
SecondaryUOMCode	string
ShipToCustomerContactEmail	string
ShipToCustomerLocationOriginalSystemRe	string
ShipToLocationCode	string
ShipToLocationId	long
SourceAgreementId	long
SourceAgreementLineId	long
SourceAgreementLineNumber	decimal
SourceAgreementNumber	string
SupplierContactEmail	string
SupplierContactFax	string
SupplierContactId	long
SupplierContactName	string
SupplierContactPhone	string
SupplierItemNumber	string
UNNumber	string
UNNumberId	long
UrgentRequisitionLineFlag	boolean

You can't use these attributes in the PurchaseRequestLine element.

Attribute Name	Type
AssessableAmount	decimal
AttachmentCategoryItem	string
AttachmentCategoryShip	string
AttachmentEntityName	string
AttachmentPrimaryKey1	string
AttachmentPrimaryKey2	string
AttachmentPrimaryKey3	string
AttachmentPrimaryKey4	string
AttachmentPrimaryKey5	string
BackToBackFlag	boolean
BaseModelId	long
Carrier	string
CarrierId	long
ConfiguredItemFlag	boolean
CustomerItemNumber	string
CustomerPOLineNumber	string
CustomerPONumber	string
CustomerPOScheduleNumber	string
DaysLateReceiptAllowed	decimal
DeliverToCustomerLocationId	long
DeliverToOrganizationCode	string

Attribute Name	Type
DeliverToOrganizationId	long
DestinationTypeCode	string
FirstPartyTaxRegistrationId	long
FirstPartyTaxRegistrationNumber	string
InterfaceLineKey	string
ItemId	long
ItemNumber	string
LineIntendedUse	string
LineIntendedUseId	long
LineIntendedUseName	string
LocationOfFinalDischargeCode	string
LocationOfFinalDischargeId	long
ModeOfTransportCode	string
OrchestrationCode	string
OverReceiptTolerancePercent	decimal
ProductCategory	string
ProductCategoryName	string
ProductFiscalClassification	string
ProductFiscalClassificationId	long
ProductFiscalClassificationName	string
ProductType	string

Attribute Name	Type
ProductTypeName	string
Quantity	decimal
RequestedDeliveryDate	date
RequestedShipDate	date
SalesOrderLineNumber	string
SalesOrderNumber	string
SalesOrderScheduleNumber	string
ServiceLevelCode	string
ShipToCustomerContactId	long
ShipToCustomerContactName	string
ShipToCustomerContactNumber	string
ShipToCustomerId	long
ShipToCustomerLocationCLLICode	string
ShipToCustomerLocationId	long
ShipToCustomerName	string
ShipToCustomerNumber	string
SupplierId	long
SupplierName	string
SupplierSiteId	long
SupplierSiteName	string
TaxClassificationCode	string

Attribute Name	Type
TaxClassificationName	string
ThirdPartyTaxRegistrationId	long
ThirdPartyTaxRegistrationNumber	string
TransactionBusinessCategory	string
TransactionBusinessCategoryName	string
UnitOfMeasure	string
UnitOfMeasureCode	string
UserDefinedFiscalClassification	string
UserDefinedFiscalClassificationName	string
WorkOrderId	long
WorkOrderNumber	string
WorkOrderOperationId	long
WorkOrderOperationSequence	decimal
WorkOrderProduct	string
WorkOrderSubTypeCode	string

Purchase Request Requisition Distribution

You can't use any attributes in the PurchaseRequestInputReqDistInterface element.

Purchase Request Configuration

You can use these attributes in the PurchaseRequestConfig element.

Attribute Name	Type
Description	string
FulfillLineId	long

Attribute Name	Type
ItemRevision	string
Price	decimal
SupplierItemNumber	string

You can't use these attributes in the PurchaseRequestConfig element.

Attribute Name	Type
AttachmentCategory	string
AttachmentEntityName	string
AttachmentPrimaryKey1	string
AttachmentPrimaryKey2	string
AttachmentPrimaryKey3	string
AttachmentPrimaryKey4	string
AttachmentPrimaryKey5	string
BaseModelId	long
BaseModelNumber	string
ComponentLineId	long
ComponentLineNumber	long
InterfaceConfigurationComponentKey	string
ItemId	long
ItemNumber	string
ItemType	string

Attribute Name	Type
ParentComponentLineId	long
Quantity	decimal
UnitOfMeasure	string
UnitOfMeasureCode	string

Attributes You Can't Use in Any Element with Drop Ship

You can't use these attributes in any element with drop ship.

Attribute Name	Type
interfaceSourceCode	string
requisitioningBUId	long
requisitioningBUName	string
groupBy	string
nextRequisitionNumber	int
initiateApprovalAfterRequisitionImport	string
maximumBatchSize	int
errorProcessingLevel	string

Attributes You Can Use With Oracle Receiving

Receipt Advice

You can use these attributes in the ReceiptAdvice element.

Attribute Name	Type
Comments	string
CurrencyConversionDate	date

Attribute Name	Type
CurrencyConversionRate	decimal
CurrencyConversionType	string
CustomerSiteId	long
DocumentRevisionDate	dateTime
DocumentRevisionNumber	string
FreightCarrierId	long
FreightCarrierName	string
FreightTerms	string
HeaderId	long
NoteToReceiver	string
OutsourcerContactId	long
OutsourcerContactName	string
OutsourcerPartyId	long
OutsourcerPartyName	string
ShipFromLocationCode	string
ShipFromLocationId	long
ShipToLocationCode	string
ShipToLocationId	long
ShipToOrganizationCode	string
ShipToOrganizationId	long
VendorId	long

Attribute Name	Type
VendorName	string
VendorSiteId	long
VendorSiteName	string

You can't use these attributes in the ReceiptAdvice element.

Attribute Name	Type
CustomerPartyName	string
ActionCode	string
BUId	long
CurrencyCode	string
CustomerId	long
DocumentCreationDate	dateTime
DocumentLastUpdateDate	dateTime
DocumentNumber	string
ReceiptAdviceNumber	string
ReceiptAdviceOriginalSystemReference	string
ReceiptSourceCode	string
SourceDocumentType	string
SourceSystemId	long

ReceiptAdviceLine

You can use these attributes in the ReceiptAdviceLine element.

Attribute Name	Type
AllowSubstituteReceipt	string
Comments	string
CountryOfOriginCode	string
DaysEarlyReceiptAllowed	decimal
DaysLateReceiptAllowed	decimal
DocumentScheduleNumber	string
EnforceShipToLocCode	string
FulfillLineId	long
ItemCategory	string
ItemRevision	string
NoteToReceiver	string
OrigSourceFulfilLineNumber	string
OverReceiptExceptionCode	string
OverReceiptTolerance	decimal
ReceiptDaysExceptionCode	string
RoutingCode	string
RoutingHeaderId	long
SecondaryQuantityExpected	decimal
SecondaryUnitOfMeasure	string
SecondaryUOMCode	string
ShipFromLocationCode	string

Attribute Name	Type
ShipFromLocationId	long
ShipToLocationCode	string
ShipToLocationId	long
SourceLineNumber	string
SubstituteItemId	long
SubstituteItemNumber	string
VendorItemNumber	string

You can't use these attributes in the ReceiptAdviceLine element.

Attribute Name	Type
AssessableValue	decimal
CustomerItemId	long
CustomerItemNumber	string
DefaultTaxationCountry	string
DocumentFiscalClassification	string
DocumentLineCreationDate	dateTime
DocumentLineLastUpdateDate	dateTime
DocumentLineNumber	string
ExpectedReceiptDate	dateTime
FinalDischargeLocationId	long
FirstPtyRegId	long
IntendedUseClassifId	long

Attribute Name	Type
ItemDescription	string
ItemId	long
ItemNumber	string
OrigSalesOrderLineNumber	string
OrigSalesOrderNumber	string
OrigSourceHeaderNumber	string
OrigSourceLineNumber	string
ProductCategory	string
ProductFiscClassId	long
ProductType	string
QuantityExpected	decimal
ReceiptAdviceLineNumber	string
ReceiptSourceCode	string
ShipToOrganizationCode	string
ShipToOrganizationId	long
SourceFulfillmentLineNumber	string
SourceHeaderNumber	string
TaxClassificationCode	string
TaxInvoiceDate	date
TaxInvoiceNumber	string
TaxShipFromLocationId	long

Attribute Name	Type
ThirdPtyRegId	long
TransactionDate	dateTime
TrxBusinessCategory	string
UnitOfMeasure	string
UOMCode	string
UserDefinedFiscClass	string

Lot for the Receipt Advice Line

You can use these attributes in the ReceiptAdviceLineLot element.

Attribute Name	Type
FulfillLineId	long
GradeCode	string
LotExpirationDate	dateTime
ParentLotNumber	string
SecondaryTransactionQuantity	decimal
SecondaryTransactionQuantityUOMCode	string
TransactionQuantityUOMCode	string

You can't use these attributes in the ReceiptAdviceLineLot element.

Attribute Name	Type
LotNumber	string
TransactionQuantity	decimal

Serials for the Receipt Advice Line

You can use these attributes in the ReceiptAdviceLineSerials element.

Attribute Name	Type
FulfillLineId	long

You can't use these attributes in the ReceiptAdviceLineSerials element.

Attribute Name	Type
FromSerialNumber	string
ToSerialNumber	string

Lot for Receipt Advice Lines That Include Serials

You can use these attributes in the ReceiptAdviceLineLotWithSerials element.

Attribute Name	Type
FulfillLineId	long
GradeCode	string
LotExpirationDate	dateTime
ParentLotNumber	string
SecondaryTransactionQuantity	decimal
SecondaryTransactionQuantityUOMCode	string
TransactionQuantityUOMCode	string

You can't use these attributes in the ReceiptAdviceLineLotWithSerials element.

Attribute Name	Type
TransactionQuantity	decimal

Lot for Receipt Advice Line When Serials Control the Lot

You can't use any attributes in the ReceiptAdviceLineLotSerials element. Specifically, you can't use these attributes.

Attribute Name	Type
FromSerialNumber	string
ToSerialNumber	string

Attributes You Can Use With Oracle Shipping

Shipment Request Header

You can use these attributes in the ShipmentRequestHeader element.

Attribute Name	Type
CarrierPartyId	long
CarrierPartyName	string
CarrierPartyNumber	string
FobCode	string
FreightTermsCode	string
HeaderId	Long
ModeOfTransport	string
OutsourcerPartyId	long
OutsourcerPartyName	string
OutsourcerPartyNumber	string
ServiceLevel	string
ShipFromLocationId	long

You can't use these attributes in the ShipmentRequestHeader element.

Attribute Name	Type
ActionType	string
BillToContactId	long
BillToContactName	string
BillToContactNumber	string
BillToPartyId	long
BillToPartyName	string
BillToPartyNumber	string
BillToPartySiteId	long
BillToPartySiteNumber	string
ConversionDate	dateTime
ConversionRate	decimal
ConversionType	string
CurrencyCode	string
DocumentNumber	string
OrganizationCode	string
OrganizationId	long
OrganizationName	string
SalesOrderNumber	string
ShipToContactId	long
ShipToContactName	string
ShipToContactNumber	string

Attribute Name	Type
ShipToPartyId	long
ShipToPartyName	string
ShipToPartyNumber	string
ShipToPartySiteId	long
ShipToPartySiteNumber	string
SoldPartyName	string
SoldToContactId	long
SoldToContactName	string
SoldToContactNumber	string
SoldToPartyId	long
SoldToPartyNumber	string
SourceDocumentType	string
SourceHeaderNumber	string
SourceSystemId	long
SourceSystemName	string

Shipment Request Line

You can use these attributes in the ShipmentRequestLine element.

Attribute Name	Type
ArrivalSetName	string
CurrencyCode	string
CustomerItemNumber	string

Attribute Name	Type
DoNotShipAfterDate	dateTime
DoNotShipBeforeDate	dateTime
EarliestDropoffDate	dateTime
EarliestPickupDate	dateTime
EndAssemblyItemNumber	string
FinalDischargeLocCode	string
FulfillLineId	long
InitialDestinationLocationId	long
IntendedUse	string
ItemDescription	string
LatestDropoffDate	dateTime
LatestPickupDate	dateTime
OrderedQuantity2	decimal
OrderedQuantityUomCode2	string
OrderedQuantityUomName2	string
PreferredGrade	string
ShipToleranceAbove	decimal
ShipToleranceBelow	decimal
SourceLineUpdateDate	dateTime
SourceShipmentId	long
SrcAttribute1	string

Attribute Name	Type
SrcAttribute10	string
SrcAttribute11	string
SrcAttribute12	string
SrcAttribute13	string
SrcAttribute14	string
SrcAttribute15	string
SrcAttribute16	string
SrcAttribute17	string
SrcAttribute18	string
SrcAttribute19	string
SrcAttribute2	string
SrcAttribute20	string
SrcAttribute3	string
SrcAttribute4	string
SrcAttribute5	string
SrcAttribute6	string
SrcAttribute7	string
SrcAttribute8	string
SrcAttribute9	string
SrcAttributeCategory	string
SrcAttributeDate1	date

Attribute Name	Type
SrcAttributeDate2	date
SrcAttributeDate3	date
SrcAttributeDate4	date
SrcAttributeDate5	date
SrcAttributeNumber1	decimal
SrcAttributeNumber10	decimal
SrcAttributeNumber2	decimal
SrcAttributeNumber3	decimal
SrcAttributeNumber4	decimal
SrcAttributeNumber5	decimal
SrcAttributeNumber6	decimal
SrcAttributeNumber7	decimal
SrcAttributeNumber8	decimal
SrcAttributeNumber9	decimal
SrcAttributeTimestamp1	dateTime
SrcAttributeTimestamp2	dateTime
SrcAttributeTimestamp3	dateTime
SrcAttributeTimestamp4	dateTime
SrcAttributeTimestamp5	dateTime
TradeComplianceScreeningDate	dateTime
TradeComplianceScreeningReason	string

Attribute Name	Type
TradeComplianceScreeningStatus	string
TransportationPlanningStatus	string
TransportationShipment	string
TransportationShipmentLine	string

You can't use these attributes in the ShipmentRequestLine element.

Attribute Name	Type
AssessableValue	decimal
BaseItemId	long
BaseItemNumber	string
BillToContactId	long
BillToContactName	string
BillToContactNumber	string
BillToPartyId	long
BillToPartyName	string
BillToPartyNumber	string
BillToPartySiteId	long
BillToPartySiteNumber	string
BusinessUnitName	string
CarrierPartyId	long
CarrierPartyName	string
CarrierPartyNumber	string

Attribute Name	Type
CategoryId	long
CategoryName	string
CustPoNumber	string
DateRequested	dateTime
DateScheduled	dateTime
DefaultTaxationCountry	string
DocumentSubType	string
ExemptCertificateNumber	string
ExemptReasonCode	string
FinalDischargeLocationId	long
FirstPtyNumber	string
FirstPtyRegId	long
FobCode	string
FreightTermsCode	string
FromSubinventoryCode	string
IntendedUseClassifId	decimal
InventoryItemId	long
ItemNumber	string
LineActionType	string
ModeOfTransport	string
OrderedQuantity	decimal

Attribute Name	Type
OrderedQuantityUomCode	string
OrderedQuantityUomName	string
OrganizationCode	string
OrganizationId	long
OrganizationName	string
OrgId	long
PackingInstructions	string
ParentInventoryItemId	long
ParentItemNumber	string
ParentSourceShipmentId	long
ProductCategory	string
ProductType	string
ReleaseLock	string
RequestDateTypeCode	string
SalesOrderLineNumber	string
SalesOrderShipmentNumber	string
SellingPrice	decimal
ServiceLevel	string
ShipmentPriorityCode	string
ShippingInstructions	string
ShipSetName	string

Attribute Name	Type
ShipToContactId	long
ShipToContactName	string
ShipToContactNumber	string
ShipToPartyId	long
ShipToPartyName	string
ShipToPartyNumber	string
ShipToPartySiteId	long
ShipToPartySiteNumber	string
SoldToContactId	long
SoldToContactName	string
SoldToContactNumber	string
SoldToPartyId	long
SoldToPartyName	string
SoldToPartyNumber	string
SourceLineNumber	string
SourceShipmentNumber	string
TaxClassificationCode	string
TaxInvoiceDate	date
TaxInvoiceNumber	string
ThirdPtyNumber	string
ThirdPtyRegId	long

Attribute Name	Type
TrxBusinessCategory	string
UnitPrice	decimal
UserDefinedFiscClass	string

Order Line for a Shipment Request Hold

You can't use any attributes in the ShipmentRequestOrderLineHold element. Specifically, you can't use these attributes.

Attribute Name	Type
HoldActionCode	string
HoldId	long
HoldReasonCode	string
HoldReason	string

Attributes You Must Not Use

You must not use these fulfillment line attributes in any integration.

- Unit List Price
- Unit Selling Price
- Extended Amount

Related Topics

- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Use Descriptive Flexfields to Integrate Order Management with Other Oracle Applications
- Specify Transaction Types When You Integrate Order Management with Accounts Receivable
- Pricing Algorithms
- Service Mapping

Web Services

Overview

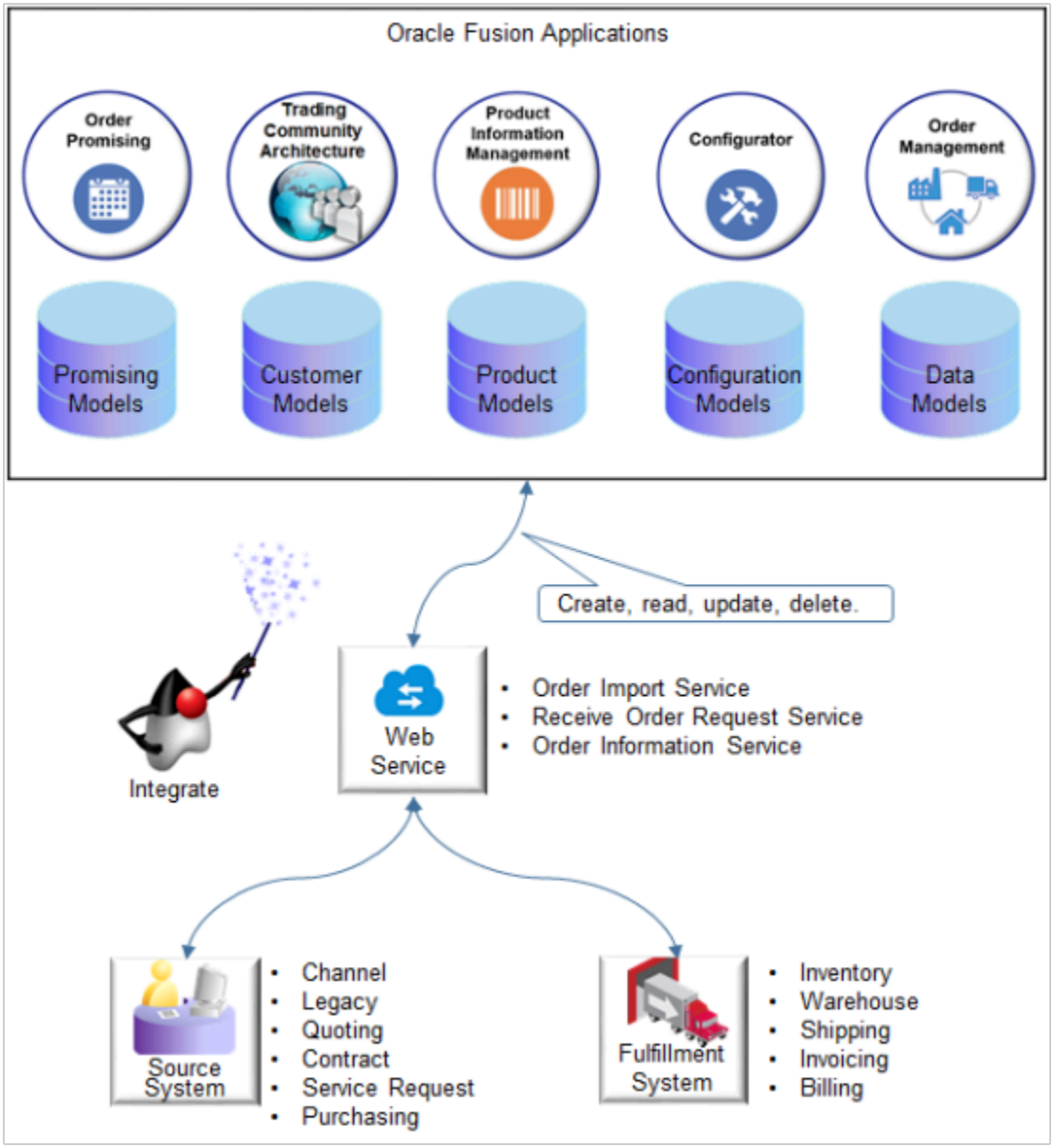
Web Services That You Can Use to Integrate Order Management

Use Order Import Service, Receive Order Request Service, or Order Information Service to integrate Order Management with some other system.

Here are some examples of other systems that you can integrate.

- Channel
- Legacy
- Quoting
- Contract
- Service request
- Purchasing
- Inventory
- Warehouse
- Shipping
- Invoicing
- Billing

Create, read, update, and delete the same data that Oracle Applications use.



For example:

Oracle Application	Data
Order Promising	Promising model
Trading Community Architecture	Customer model
Product Information Management	Product model
Configurator	Configuration model

Oracle Application	Data
Order Management	Data model

Terminology

- **Source system.** An order capture system or application that sends the Order Request object to the Order Import Service web service. For brevity, this document mentions only source system.
- **Internal.** An action, process, or object that resides in Order Management. A cross-reference that resides in Order Management and Planning Repository is an example of an internal object.
- **External.** An action, process, or object that resides outside of Oracle Order Management. A source order that a source system creates in an order capture system is an example of an external object.
- **Source order.** An order that a source system creates in an order capture system.
- **Sales order.** A source order that Order Management converts into a sales order so it can fulfill the source order.

Should I Use REST API, Order Import Service, or Receive Order Request Service?

We recommend that you use REST API instead of Order Import Service or Receive Order Request Service.

- If you're already using Order Import Service, we recommend that you migrate that usage to REST API. Order Management continues to support Order Import Service, but will retire it and no longer support it in a future update.
- Order Management continues to support Receive Order Request Service in Update 12, but will retire it and no longer support it in subsequent updates.
- Use REST API instead of Receive Order Request Service.
- Use REST API instead of Receive Order Request Service to create a sales order.
- Order Management no longer supports the Create operation or Submit operation of Receive Order Request Service.
- You can use the pause task, apply hold, release hold, and check availability operations of Receive Order Request Service.

Order Import Service

Use the Order Import Service web service to create an integration that sends order requests from your upstream system to Order Management. This web service processes the request, then creates a sales order in Order Management. You can also use it to submit a draft sales order to fulfillment.

Order Import Service is a SOAP (Simple Object Access Protocol) service that uses Order Request Object as the payload. Learn about the operations you can use and the attributes you reference. For details, go to *SOAP Web Services for Oracle SCM*. Expand **Business Object Services > Import Sales Orders**.

Here are the details.

Details	Description
Type	Technical
Required	No
Service Name	OrderImportService

Details	Description
Description	This Service receives source orders from different channel systems.
WSDL	http://host:port/fomImportOrdersService/OrderImportService?wsdl
`Input`	OrderImportServiceRequest
Output	OrderImportServiceResponse

Use these operations with Order Import Service.

Operation	Input Message	Output Message
createOrders	createOrders	createOrdersResponse
createOrdersAsync	createOrdersAsync	createOrdersAsyncResponse
stageOrders	stageOrders	stageOrdersResponse
stageOrdersAsync	stageOrdersAsync	stageOrdersAsyncResponse
SubmitDraftOrder	SubmitDraftOrder	SubmitDraftOrderResponse
SubmitDraftOrderAsync	SubmitDraftOrderAsync	SubmitDraftOrderAsyncResponse

Create Orders Operation

Use the createOrders operation to import source orders from different channels into Order Management. Create a sales order in draft mode or submit mode according to the SubmitFlag attribute in the payload.

Stage Orders Operation

Use the stageOrders operation to import orders from different channels to staging tables. You can then run a scheduled process that imports data from interface tables, processes them, then imports each interface record into Order Management as a sales order.

Submit Draft Order Operation

Use the SubmitDraftOrder operation to submit a draft sales order to order fulfillment.

Receive Order Request Service

Use the Receive Order Request Service web service to do a pause task, apply a hold, release a hold, or check availability.

Receive Order Request Service is a SOAP (Simple Object Access Protocol) service that uses Order Request Object as the payload.

Use the Order Import Service web service instead of Receive Order Request Service to create a sales order.

Here are the details.

Details	Description
Type	Technical
Required	No
Service Name	ReceiveOrderRequestService
Description	This service receives source orders from different channel systems.
WSDL	http: //host: port/ soa-infra/ services/ default/ DooDecompReceiveOrderExternalComposite/ ReceiveOrderRequestServiceWSDL
`Input`	ReceiveOrderServiceRequestMessage
Output	ReceiveOrderServiceResponseMessage

Related Topics

- Overview of Importing Orders Into Order Management

Guidelines for Using Web Services to Integrate Order Management

Apply guidelines to make sure your integration to other systems with Order Management goes smoothly.

Plan

Plan Your Integration

Requirement	Description
Register source system.	You must register each source system that sends the order request as a source system. More than one source system might create a source order, so the web service stores details that identify the source system on the order so it can track the source system that sends the order. The web service uses the source system during cross-referencing to determine the corresponding internal identifier or value that's specific to the source system.
Plan how you will synchronize customer master data.	You must synchronize customer master data before you import a source order. If the customer on the order doesn't exist in the customer model in Oracle Trading according to the CreateCustomerInformationFlag preference in the service schema, then Order Management can create the customer details.

Requirement	Description
Plan for attributes in Oracle Configure, Price, and Quote Cloud.	If you use Oracle Configure, Price, and Quote Cloud (CPQ), then CPQ sends details about the attributes that Order Management requires to use this functionality, such as FreezePriceFlag, FreezeShippingChargeFlag, FreezeTaxFlag, and so on.
Plan for pricing.	You can send a source order that your upstream system already priced, or the web service can price the source order for you. If you use a predefined implementation of Oracle Configure, Price, and Quote Cloud, then the web service assumes the upstream system already priced the source order and won't allow price changes in Order Management.
Plan how to handle draft sales orders.	Your source system can send a submitted source order or a source order that's in draft status. • If the source order is in draft status, then you must submit the sales order that references the source order. • Order Management won't run the orchestration process that fulfills your sales order until you submit it. • If the web service creates the order in draft status, then you can submit it from the Order Management work area, or you can use operation SubmitDraftOrder in the web service to submit it. • If you use SubmitDraftOrder, then you must add the payload that defines it.

Use Asynchronous or Synchronous Services

Asynchronous or Synchronous?	Description
The integration platform or application that uses the service supports an asynchronous service.	Use the asynchronous operation because asynchronous is more resilient and fault tolerant than synchronous. If the response from the fulfillment system is delayed for some reason, then an asynchronous operation can continue processing but a synchronous operation might time out and go into an error state.
The platform or application doesn't support an asynchronous service, or you prefer not to use asynchronous because its more complex to implement.	Use synchronous as long as the number of lines in the sales order doesn't result in a timeout after 300 seconds. If you use synchronous, and if a sales order times out, then you must make sure you set up your implementation to resubmit the sales order.

Process More Than One Sales Order

Use these operations of Order Import Service.

- **stageOrders.** The only operation that can accept more than one sales order in the payload.
- **createOrders.** Can accept only one sales order in the payload.

The Receive Order Request web service can accept only one sales order in the payload.

Learn about the input messages and output messages that you can use with Order Import Service. For details, go to *SOAP Web Services for Oracle SCM*, then expand **Business Object Services > Import Sales Orders**.

Make Sure Your Payload Uses the Correct Hierarchy

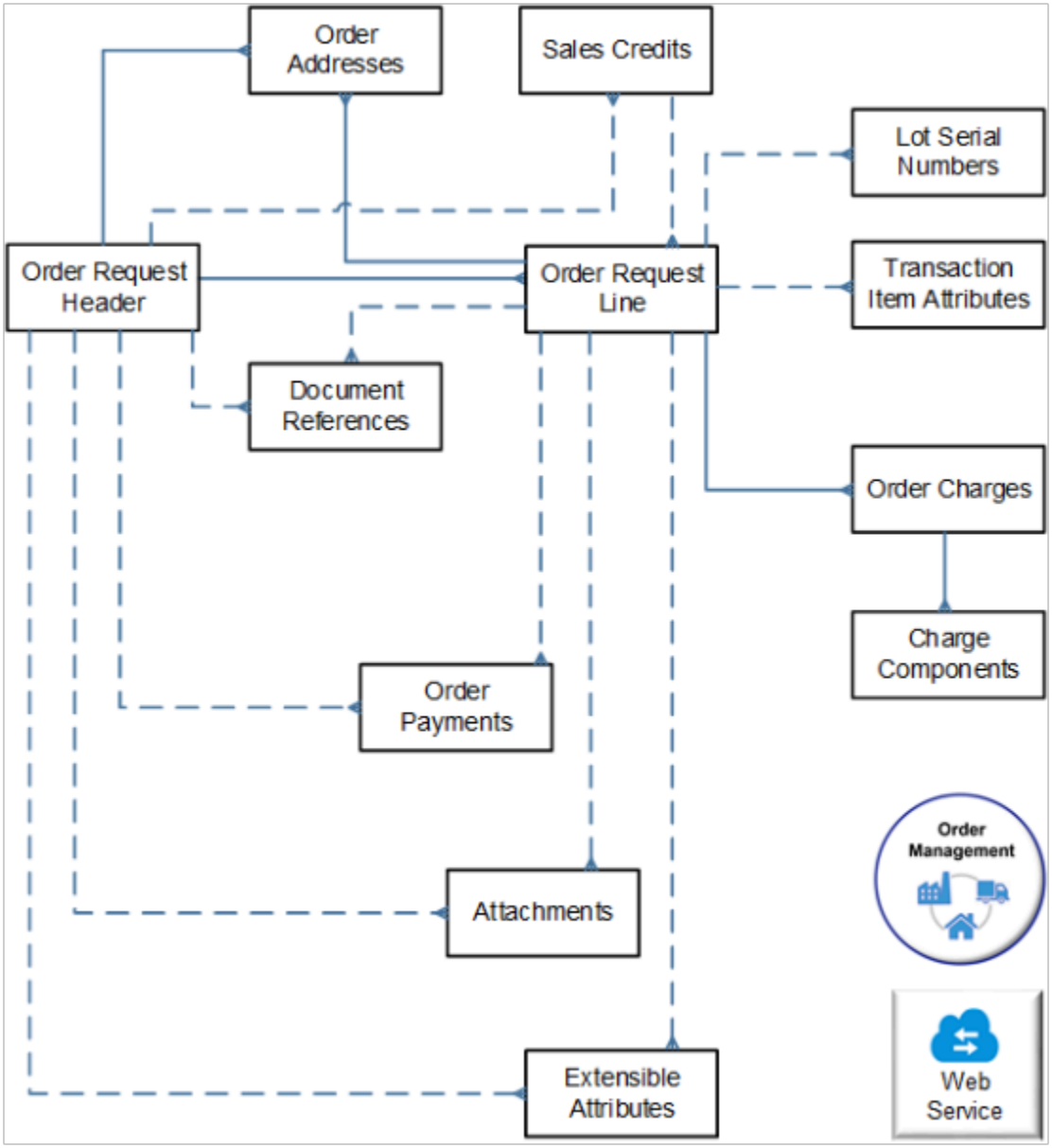
Entity Hierarchy That Web Services Support

Make sure your payload uses this entity hierarchy.

- Header
    - Order Payment
    - Order Sales Credit
    - Order Attachments
    - Order Lines
        - Line LotSerial
        - Line SalesCredit
        - Line Payments
        - Line Document References
        - Line Attachments
        - Line Transactional Attributes
        - Charge
            - Charge Components
    - Order Preferences

Logical Data Model

Make sure your payload can accommodate the data model.

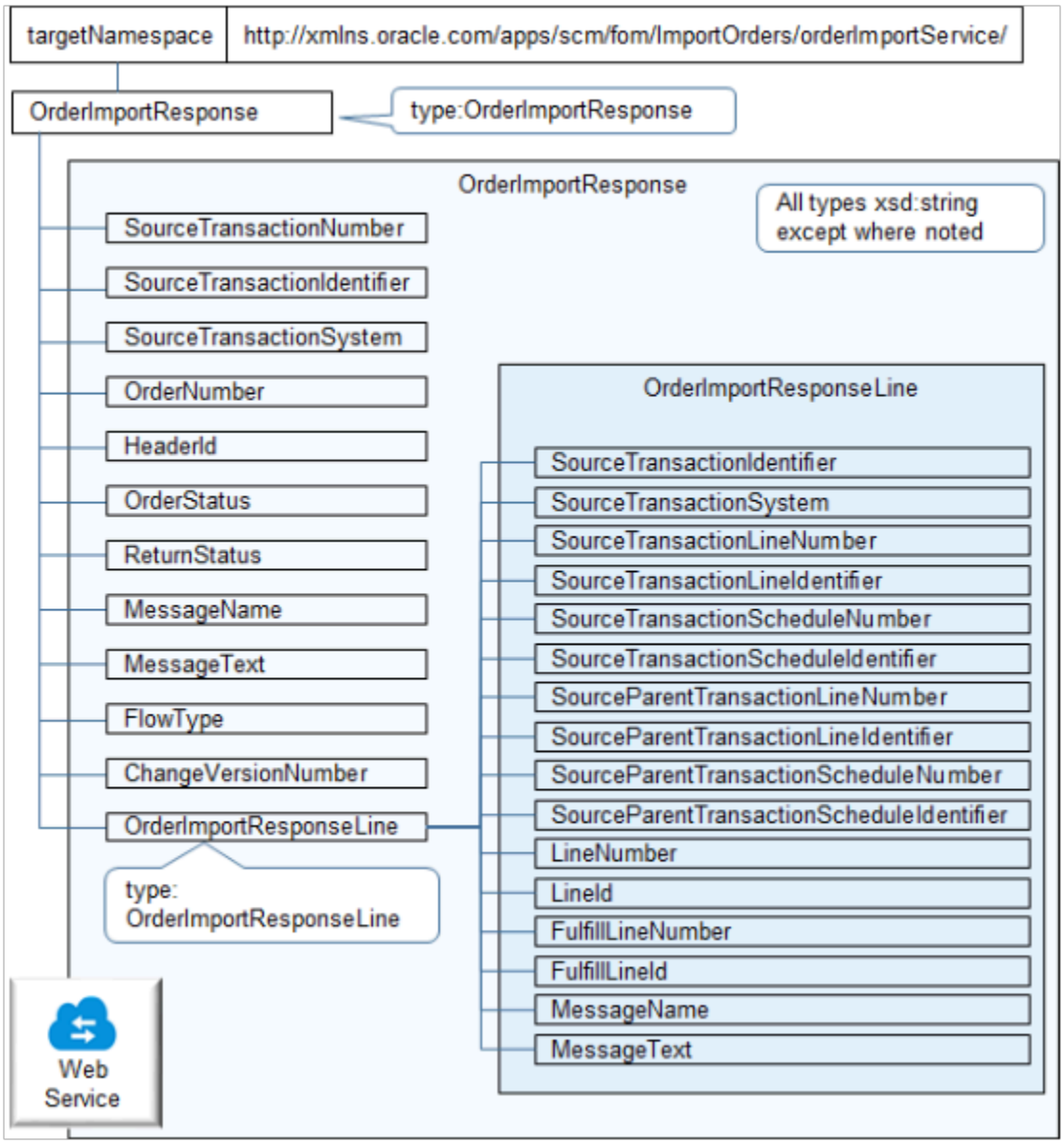


Response Payload

Make sure your integration can accommodate the response that Order Management sends. The response payload returns a status.

Status	Description
SUCCESS	The response includes source keys and Order Management keys. It sends this response after Order Management successfully creates the sales order.
FAILURE	The response includes the first validation error message.

The response payload uses this structure.



Specify Attributes in Payloads

Specify Attributes in Groups

To make sure the payload includes all the required details, the web service processes some attributes as a group. For example, here are the attributes that the web service examines as a group to make sure the payload identifies the buying party.

1. BuyingPartyId
2. BuyingPartyName
3. BuyingPartyNumber

Here's the sequence that the web service uses when it processes each group.

1. Use the attribute that specifies the identifier, such as BuyingPartyId.
2. If the attribute that specifies the identifier is empty, then use the attribute that specifies the number, such as BuyingPartyNumber.
3. If the attribute that specifies the number is empty, then use the attribute that specifies the name, such as BuyingPartyName.

If the attribute that specifies the identifier includes a value, then the web service will always use this value even if the attributes that specify the name or number aren't empty.

If the payload doesn't include a value for the identifier, name, or number, then your order import will likely fail.

Specify Coded Attributes and Their Partners

Some attributes, such as ReturnReasonCode, store an abbreviation for a longer term. The abbreviation is typically text that the user can view to quickly identify the meaning of a lookup value.

You can think of the value of this attribute as coded. A coded attribute typically includes a partner attribute. For example, the ReturnReason attribute is the partner for ReturnReasonCode.

If you provide a value only for the coded attribute in the payload, then Order Management will use the value that the database cross-references from the lookup value to the meaning. For example, it cross-references the RET lookup value to the meaning for RET, which is Return.

Continuing this example, assume you set `ReturnReasonCode=RET` in the payload, and.

- You set ReturnReason="Return Order", then the web service will ignore this value and use the code. This behavior is similar to using Identifier when you don't supply the Name.
- You don't specify a value for ReturnReason, and if the value that the database cross-references to the meaning is Return, then the web service will use Return for the reason.

The web service uses this logic for each of these sets of attributes.

Coded Attribute	Partner Attribute
AccountingRulecode	AccountingRule
CancelReasonCode	CancelReason
ChargeDefinitionCode	ChargeDefinition
ChargeSubtypeCode	ChargeSubType
DemandClassCode	DemandClass
FOBPointcode	FOBPoint
FreightTermsCode	FreightTerms
InvoicingRuleCode	InvoicingRule

Coded Attribute	Partner Attribute
OrderedUOMCode	OrderedUOM
PaymentMethodCode	PaymentMethod
PaymentTerm	PaymentTermCode
RequestedFulfillmentOrganizationCode	RequestedFulfillmentOrganizationName
RequestedSupplierCode	RequestedSupplierName
ShipmentPriorityCode	ShipmentPriority
ShippingCarrierCode	ShippingCarrier
ShippingModeCode	ShippingMode
ShippingServiceLevelCode	ShippingServiceLevel
SubInventoryCode	Subinventory
SubstitutionReasonCode	SubstitutionReason
TaxExemptReasonCode	TaxExemptReason
TransactionalCurrencyCode	TransactionalCurrencyName
TransactionLineTypeCode	TransactionLineType

Include Identifiers and Values

Use an attribute that includes the word Identifier in the name to send the identifier, such as Requesting Business Unit Identifier. If you send the identifier and the value for the identifier, then the web service uses the identifier.

Master data includes customers and items. The source system can send different details, depending on whether it uses the same master data and references data that Order Management uses.

- **Uses the same data.** The source system can send the Oracle identifier or the values.

- **Doesn't use the same data.** The source system can send the identifiers and values that it contains. Order Import Service uses them as keys to look up the cross-reference, depending on whether the key references customer data or product data. If the cross-reference resides in:
    - Oracle Trading Community Model, then resolve it into Oracle customer data
    - Product Information Management, then resolve it into Oracle product data

Each service typically uses a pair of synchronous and asynchronous operations. The service appends the operation name with a value.

- **Sync.** The other operation in the pair is asynchronous.
- **Async.** The other operation in the pair is synchronous.

Process Change Orders and Cancel Orders

Process Change Orders

To modify a sales order, you call a web service with a payload that includes these details:

- Source transaction system
- Source transaction identifier
- Order number and source transaction number of a sales order that Order Management already processed

The web service will process the order as a change order according to the combination of source transaction system and source transaction identifier.

- Use the same web service that you used to create the sales order. The payload structure for a change order is similar to the payload structure for create order.
- Design your payload so it sends the modified value for each attribute.
- Make sure your payload includes all attributes for the order line that you modify.
- If you use Oracle Application Development Framework (ADF), then you must include the entire order line in your payload even if you don't modify any part of the line. However, if you use REST API or FBDI with REST API, then you can exclude that entire line from the payload. For example, if your sales order has 100 lines but you only need to modify order line 1001, then you can use REST API or FBDI with REST API to import only line 1001. For details, see *Use FBDI and REST API to Import Large Volumes of Sales Orders*.

Cancel Sales Orders

To cancel a sales order or order line, you call the same web service that you use to create the sales order.

Here are details to include in your payload.

What You Must Cancel	Description
Cancel the entire sales order.	Set the OperationCode for the order header to CANCEL. You must also identify the source transaction system and include the source transaction identifier.
Cancel the entire order line.	Send the SourceTransactionLineIdentifier and SourceTransactionScheduleIdentifier for the order. Do one of. • Set the OperationCode attribute for the order line to CANCEL.

What You Must Cancel	Description
	• Set the Ordered Quantity attribute to 0.
Cancel part of a shipped order line.	Set the ordered quantity to the quantity that already shipped. For example, if the quantity on the original order line is 10 Each, and if 7 shipped, and if 3 were back ordered, then set the ordered quantity in the payload to 7 Each. Make sure your payload also includes all other attributes from the original order line.

Example of Canceling Part of an Order Line

Consider an example.

- Quantity originally ordered equals 100
- Quantity already shipped equals 40
- Quantity awaiting shipping or backordered equals 60

Use these values.

Scenario	Quantity to Send in Payload for Update Service	Description
Your customer needs a revised total quantity of 55.	55	The quantity of 55 in the payload replaces the original quality. 40 already shipped, so order fulfillment cancels 45 of the 60 that are currently awaiting shipping or backordered, leaving 15 that are still awaiting shipping or backordered.
You must cancel the quantity that hasn't shipped.	40	The quantity of 40 in the payload replaces the original quality. Order fulfillment cancels the 60 that are currently awaiting shipping or backordered.
Assume quantity already shipped is 0, quantity awaiting shipping is 100, and you must cancel the entire quantity.	0	The quantity of 0 in the payload replaces the original quality. Order fulfillment cancels the 100 that are currently awaiting shipping.

Use Security with Web Services

The web service attaches an LPA security policy to the service and the callback. It includes a hybrid policy.

- oracle/wss11_saml_or_username_token_with_message_protection_service_policy

This policy supports five different assertions, including this one.

- oracle/wss_username_token_over_ssl_client_policy

The callback includes an attachment.

- oracle/wss_username_token_over_ssl_client_policy LPA

Use these settings to call the web service only with SSL (Secure Sockets Layer).

Use Files to Import Source Orders

Use the Order Import Template to import source orders. It helps reduce errors and simplifies order import. It contains a structure that the Oracle database requires. For example, it includes a tab for each database table, and it displays these tabs in a specific sequence.

Automate using files to import source orders. Oracle provides a set of web services you can use to upload the completed import template to the server that hosts Oracle WebCenter Content. You then run a scheduled process that imports the uploaded file to the interface tables, processes them, then imports each interface record as a sales order. For details, see *Using the Oracle ERP Cloud Adapter with Oracle Integration*.

Here are the parameters you use when you run the scheduled process.

Parameter	Value
JobDefinitionName	ImportOrdersJob
JobPackageName	oracle/apps/ess/scm/doo/decomposition/receiveTransform/receiveSalesOrder

Related Topics

- Overview of Importing Orders Into Order Management

Reference

Example Web Service Payloads That Integrate Order Management

Get some example payloads in xml files.

To download the xml files, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Example Web Service Payloads attachment.

Example Payloads for the Create Order Operation

Here are some example payloads for the Create Order operation of the Order Import Service. These payloads create sales orders.

Description	Payload
Create or submit draft sales orders.	draft_sales_order.xml submit_draft_order_sync.xml

Description	Payload
Include pricing.	single_line_priced_in_source_system.xml single_line_priced_in_oracle_fusion.xml Note • The single_line_priced_in_oracle_fusion.xml payload includes the text QP, which is an acronym for Quality Pricing. It indicates that Oracle Pricing calculates pricing for the sales order. • The single_line_priced_in_source_system.xml payload includes the text PREPRICED, which means your source system calculates pricing for the sales order.
Include child entities.	child_entity_transactional_item_attribute.xml child_entity_sales_credits child_entity_manual_price_adjustment.xml child_entity_lot_serial.xml
For new customers.	customer_sync_person.xml customer_sync_organization.xml
Include extensible flexfields.	sales_order_with_extensible_flexfield.xml This payload includes the text EFF, which is an acronym for extensible flexfield.
Include billing details.	recurring_billing.xml
Include shipment sets.	shipset_order.xml

Here are some more examples that use the Create Order operation.

Description	Payload
Return sales orders.	return_order_with_reference.xml
Cancel sales orders.	cancel_sales_order.xml
Cancel order line.	cancel_order_lines.xml
Example of a successful operation.	successful_response_for_CreateOrder.xml
Example of a failed operation.	failed_response_for_CreateOrder.xml

Example Payloads for the Stage Order Operation

The payloads for the Stage Order operation are identical to the payloads for the Create Order operation except you use stageOrders at the beginning and at the end of the body. For example, the draft_sales_order.xml payload uses stageOrders on lines 3 and 10 for the stage order operation. To use the Create Orders operation, you replace stageOrders on lines 3 and 10 with createOrders.

successful_response_for_StageOrder.xml is an example of a successful response for the Stage Order operation.

You can't include the Attachment entity in a stage order payload. For example, if you include this content in your payload, it will fail.

```
<ord:Attachment>
 <ord:Title>HeaderTEXTTitle</ord:Title>
 <ord:FileContent></ord:FileContent>
 <ord:FileName></ord:FileName>
 <ord:Data>VGhlIHNvdXJjZSBmaWxlIGlzIGhhbmRsZWQgYXMgYSBiaW5hcnkgZGF0YS4gVGhlIHRleHRib3gg
aXMgaGFuZGxlZCBhcyBhIHN0cmluZyBkYXRhLCBkZWZhdWx0IGNoYXJhY3RlciBzZXQgZm9yIHRo
ZSB0ZXh0Ym94IGlzICdpc28tODg1OS0xJy4gWW91IGNhbiBjaGFuZ2UgdGhlIGNoYXJzZXQgdXNp
bmcgZm9ybSBiZWxsb3cu</ord:Data>
 <ord:Description>HeaderTEXTDesc</ord:Description>
 <ord:url></ord:url>
 <ord:EntityAttributes></ord:EntityAttributes>
 <ord:DataTypeCode>TEXT</ord:DataTypeCode>
</ord:Attachment>
```

Other Examples

Use the ProjectDetail tag to include project details. Here's an example.

```
<ns2:Project xmlns:pjc="http://xmlns.oracle.com/apps/flex/scm/doo/processOrder/pjcDff/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
<ns2:ProjectDetail xsi:type="pjc:OmSalesOrder">
<pjc:projectId>300100010341182</pjc:projectId>
<pjc:taskId>300100010341193</pjc:taskId>
<pjc:expenditureItemDate>2019-03-20</pjc:expenditureItemDate>
<pjc:expenditureTypeId>300100036998309</pjc:expenditureTypeId>
<pjc:organizationId>204</pjc:organizationId>
</ns2:ProjectDetail>
</ns2:Project>
```

Attributes That You Can Use with Web Services

Get details about the attributes you can include in the request payload when you use the Receive Order Request web service.

Learn more about the objects and operations that you can use. Go to *SOAP Web Services for Oracle SCM*, then expand the table of contents to get details about the service you need.

- **Business Objects Services > Import Sales Orders**. Get details about the attributes you must include in the order header and other child entities when you use the Order Import service.
- **Business Objects Services > Order Fulfillment Response Service**
- **Business Objects Services > Order Information Service**
- Business Objects Services > Order Information Service

Attributes for the Order Header

Required Attributes You Must Include in Your Payload

Attribute	Type	Description
BuyingPartyId	Number	Value that identifies the person, company, or organization that placed the source order, sometimes known as the Sold To Customer.
RequestingBusinessUnitIdentifier	Number	Value that identifies the organization that sold the source order.
SourceTransactionIdentifier	VARCHAR2	Value that uniquely identifies the transaction.
SourceTransactionNumber	VARCHAR2	Transaction number that the source system uses.
SourceTransactionRevisionNumber	VARCHAR2	Revision number of the transaction.
SourceTransactionSystem	VARCHAR2	Source system that placed the request for fulfillment.
TransactionalCurrencyCode	VARCHAR2	Currency code for pricing the transaction.
TransactionOn	Timestamp	Date and time that the transaction started. This value identifies the date that the customer committed to purchase the items that this source order contains. Order Management uses this date to measure the time required to fulfill the sales order.

Optional Attributes

Attribute	Type	Description
BillToAccountContactIdentifier	VARCHAR2	Value that identifies the contact for the billing address.
BillToAccountContactName	VARCHAR2	Contact name for the billing address.
BillToAccountContactNumber	VARCHAR2	Contact number for the billing address.
BillToAccountContactOrigSystemReference	VARCHAR2	Cross-reference value of the account contact.
BillToAddress1	VARCHAR2	Address Line 1 of the party that's legally responsible for payment.
BillToAddress2	VARCHAR2	Address Line 2 of the party that's legally responsible for payment.

Attribute	Type	Description
BillToAddress3	VARCHAR2	Address Line 3 of the party that's legally responsible for payment.
BillToAddress4	VARCHAR2	Address Line 4 of the party that's legally responsible for payment.
BillToAddressOrigSystemReference	VARCHAR2	Cross-reference value for the account site.
BillToCity	VARCHAR2	City of the party that's legally responsible for payment.
BillToCounty	VARCHAR2	County of the party that's legally responsible for payment.
BillToCustomerIdentifier	VARCHAR2	Value that identifies the party that's legally responsible for payment.
BillToPartyName	VARCHAR2	Name of the party that's legally responsible for payment.
BillToPartyNumber	VARCHAR2	Number of the party that's legally responsible for payment.
BillToPartySiteIdentifier	VARCHAR2	Value that identifies the party that's legally responsible for payment.
BillToPostalCode	VARCHAR2	Postal Code of the party that's legally responsible for payment.
BillToProvince	VARCHAR2	Province of the party that's legally responsible for payment.
BillToState	VARCHAR2	State of the party that's legally responsible for payment.
BuyingPartyContactId	Number	Value that identifies the person who placed the source order or whose the primary contact for the source order.
BuyingPartyContactName	VARCHAR2	Name of the person who placed the source order or whose the primary contact for the source order.
BuyingPartyContactNumber	VARCHAR2	Contact number for the person who placed the source order or whose the primary contact for the source order.

Attribute	Type	Description
BuyingPartyContactOrigSystemReference	VARCHAR2	Cross-reference value of the person who placed the source order or whose the primary contact for the source order. The customer master in TCA maintains this value.
BuyingPartyName	VARCHAR2	Name of the person, company, or organization that placed the source order, also known as the Sold To Customer.
BuyingPartyNumber	VARCHAR2	Person, company, or organization number that placed the source order, also known as the Sold To Customer.
BuyingPartyOrigSystemReference	VARCHAR2	Cross reference value for the person, company, or organization number that placed the source order. The customer master in the TCA maintains this value.
CancelReason	VARCHAR2	Reason for the cancel.
CancelReasonCode	VARCHAR2	Brief text that identifies the cancel reason.
Comments	VARCHAR2	Text that describes the transaction.
CurrencyConversionDate	Date	Currency conversion date.
CurrencyConversionRate	Number	The exchange rate that Order Management must use if it converts the value from one currency to another currency.
CurrencyConversionType	VARCHAR2	Conversion type for foreign currency.
CustomerPONumber	VARCHAR2	The purchase order number that the customer sends to identify the source order.
FreezePriceFlag	VARCHAR2	Set to true or false. • **true**. The source system prices the sales order, prices are frozen so Oracle Pricing won't price it, and you must provide data for charges and charge components in the import payload. • **false**. Prices aren't frozen, Oracle Pricing will price it, and you don't need to provide values for charges and charge components in the import payload.
FreezeShippingChargeFlag	VARCHAR2	Set to true or false.

Attribute	Type	Description
		• **true**. The source system calculates shipping charges for the sales order, and prices are frozen so Oracle Pricing won't calculate shipping charges. • **false**. Shipping charges aren't frozen and Oracle Pricing will calculate them.
FreezeTaxFlag	VARCHAR2	Set to true or false. • **true**. The source system calculates tax charges for the sales order, and prices are frozen so Oracle Pricing won't calculate tax charges. • **false**. Tax charges aren't frozen and Oracle Pricing will calculate them.
OperationMode	VARCHAR2	A value of CANCEL for this attribute indicates to cancel the sales order. You can use only CANCEL.
OrigSysDocumentReference	VARCHAR2	Document number from the source system.
PartialShipAllowedFlag	VARCHAR2	Specify how to ship the items that the source order contains. • **true**. Ship items in more than one shipment, if necessary. • **false**. Ship all items in a single shipment.
PricedOn	Date	The date when the document is priced. Order Management populates this attribute with the system date at the beginning of a pricing process.
RequestCancelDate	Date	Time and day when the user requested the cancel.
RequestingLegalUnit	VARCHAR2	Name of the legal entity that formed a contract with the customer.
RequestingLegalUnitIdentifier	Number	Value that identifies the legal entity that formed a contract with the customer.
ShipToAddress1	VARCHAR2	Address Line 1 of the ship-to destination.
ShipToAddress2	VARCHAR2	Address Line 2 of the ship-to destination.
ShipToAddress3	VARCHAR2	Address Line 3 of the ship-to destination.
ShipToAddress4	VARCHAR2	Address Line 4 of the ship-to destination.

Attribute	Type	Description
ShipToAddressOrigSystemReference	VARCHAR2	Cross-reference value of the address for the ship-to destination.
ShipToCity	VARCHAR2	City of the ship-to destination.
ShipToContactPartyIdentifier	VARCHAR2	Value that identifies the contact for the shipping address.
ShipToContactPartyName	VARCHAR2	Contact name for the shipping address.
ShipToContactPartyNumber	VARCHAR2	Contact number for the shipping address.
ShipToContactPartyOrigSystemReference	VARCHAR2	Cross-reference value of the party contact.
ShipToCounty	VARCHAR2	County of the ship-to destination.
ShipToPartyIdentifier	VARCHAR2	Value that identifies the party that must receive the goods.
ShipToPartyName	VARCHAR2	Name of the party that must receive the goods.
ShipToPartyNumber	VARCHAR2	Number that identifies the party that must receive the goods.
ShipToPartySiteIdentifier	VARCHAR2	Value that identifies the party site of the ship-to destination, such as 1036.
ShipToPostalCode	VARCHAR2	Postal code of the ship-to destination.
ShipToProvince	VARCHAR2	Province of the ship-to destination.
ShipToState	VARCHAR2	State of the ship-to destination.
TransactionalCurrencyName	VARCHAR2	Currency name that Order Management must use to price the transaction.
TransactionDocumentTypeCode	VARCHAR2	Specify the type of request. • Sales order • Purchase order • Internal material transfer

Attributes for Order Lines

Required Attributes

Attribute	Type	Description
OrderedQuantity	Number	Quantity of the item that the source system requested.
OrderedUOMCode	VARCHAR2	Code for the unit of measure that the source system requested, such as Unit or Kg.
ProductIdentifier	Number	Value that uniquely identifies the item that Order Management processes to fulfill the sales order.
SourceTransactionIdentifier	VARCHAR2	Value that uniquely identifies the transaction.
SourceTransactionLineIdentifier	VARCHAR2	Value that uniquely identifies the transaction line.
SourceTransactionLineNumber	VARCHAR2	Line number of the transaction line. Order Management uses this value to sort transaction lines.
SourceTransactionNumber	VARCHAR2	Transaction number that resides in the source system.
SourceTransactionRevisionNumber	VARCHAR2	Revision number of the transaction.
SourceTransactionScheduleIdentifier	VARCHAR2	Value that uniquely identifies the source transaction schedule number.
SourceTransactionScheduleNumber	VARCHAR2	Line number of a schedule line, shipment line, or subline. The source system assigned this value when it captured the source order in the source system.
SourceTransactionSystem	VARCHAR2	Name of the source system that placed the request for fulfillment.

Optional Attributes

Attribute	Type	Description
AccountingRule	VARCHAR2	Name of the accounting rule that determines the accounting period to use when recording the revenue distribution for an invoice line.

Attribute	Type	Description
AccountingRulecode	VARCHAR2	Brief text that identifies that identifies the accounting rule.
AssessableValue	Number	Price that a tax authority uses to value an item.
BillToAccountContactIdentifier	VARCHAR2	Value that identifies the bill-to account contact.
BillToAccountContactName	VARCHAR2	Name of the contact who resides at the billing address.
BillToAccountContactNumber	VARCHAR2	Number for the contact who resides at the billing address.
BillToAccountContactOrigSystemReference	VARCHAR2	Cross-reference value for the account contact.
BillToAccountSiteUseIdentifier	VARCHAR2	Value that identifies the party site whose legally responsible for payment.
BillToAddress1	VARCHAR2	Address Line 1 of the party whose responsible for payment.
BillToAddress2	VARCHAR2	Address Line 2 of the party whose responsible for payment.
BillToAddress3	VARCHAR2	Address Line 3 of the party whose responsible for payment.
BillToAddress4	VARCHAR2	Address Line 4 of the party whose responsible for payment.
BillToAddressOrigSystemReference	VARCHAR2	Cross-reference value for the account site.
BillToCity	VARCHAR2	City of the party whose responsible for payment.
BillToCounty	VARCHAR2	County of the party whose responsible for payment.
BillToCustomerIdentifier	VARCHAR2	Value that identifies the bill-to customer.
BillToCustomerName	VARCHAR2	Name of the party whose legally responsible for payment.
BillToCustomerNumber	VARCHAR2	Number of the party whose legally responsible for payment.

Attribute	Type	Description
BillToPostalCode	VARCHAR2	Postal code of the party whose responsible for payment.
BillToProvince	VARCHAR2	Province of the party whose responsible for payment.
BillToState	VARCHAR2	State of the party whose responsible for payment.
BusinessUnitIdentifier	Number	Value that identifies the business unit.
BusinessUnitName	VARCHAR2	Name of the organization that fulfills the sales order.
CancelReason	VARCHAR2	Reason for the cancel request.
CancelReasonCode	VARCHAR2	Brief text that identifies the cancel reason.
Comments	VARCHAR2	Text that the user can use to add details that are related to the order line.
ComponentIdPath	VARCHAR2	Path to the inventory item identifier for the parent of this line. The configurator populates and uses this attribute.
ConfigHeaderId	Number	Header Id of the configuration. The configurator populates and uses this attribute.
ConfigRevisionNumber	Number	Revision number of the configuration. The configurator populates and uses this attribute.
ConfiguratorPath	VARCHAR2	Runtime representation of the path to the Inventory Item Id for the component. The configurator populates and uses this attribute.
CreditChkAuthExpDate	VARCHAR2	Date when the credit authorization expires.
CreditChkAuthNumber	NUMBER	Value that uniquely identifies the authorization that Credit Management creates for the requested amount for the customer. Use CreditChkAuthNumber only if the Credit Check attribute contains Authorization or Reauthorization.
CustomerPOLineNumber	VARCHAR2	Line number from the purchase order that the buying party provides.
CustomerPONumber	VARCHAR2	Purchase order number that the buying party provides.

Attribute	Type	Description
CustomerPOScheduleNumber	VARCHAR2	Schedule number from the purchase order that the buying party provides.
CustomerProductDescription	VARCHAR2	Description of an item.
CustomerProductidentifier	Number	Value that identifies the customer item number.
CustomerProductNumber	VARCHAR2	Number that identifies the item that the customer ordered. This number resides in the customer item table.
DefaultTaxationCountry	VARCHAR2	Name of the country that Order Management uses to calculate tax.
DemandClass	VARCHAR2	Name of the demand class. A demand class can represent. • A group of customers, such as government customers or commercial customers • Sales channels • Regions • Different sources of demand, such as retail, mail order, or wholesale
DemandClassCode	VARCHAR2	Brief text that identifies the demand class.
DestinationShippingLocationIdentifier	Number	Value that identifies the shipment destination.
DestinationShipppingOrganizationIdentifier	Number	Value that identifies the organization that receives the shipment.
DocumentSubType	VARCHAR2	Name of the subtype that Order Management uses to calculate tax and tax reporting, depending on the requirements of different countries.
EarliestAcceptableArrivalDate	Date	Don't use this attribute.
EarliestAcceptableShipDate	Date	Date that specifies the earliest time that the customer is willing to ship the sales order.
ExtendedAmount	Number	Monetary value for the fulfill line quantity.
ExtendedQuantity	Number	Total quantity for a configured item. This value is a sum of the requested quantity of the

Attribute	Type	Description
		components that a configured item contains. The configurator populates and uses this attribute.
FinalDischargeLocationIdentifier	Number	Final destination location for the purchases that the customer makes.
FirstpartyTaxRegistration	Number	Registration number that Order Management sends to the supplier. The supplier uses this number to tax the transaction.
FOBPoint	VARCHAR2	Location where the seller is willing to pay for transportation of the goods to the port of shipment, plus loading costs.
FOBPointcode	VARCHAR2	Brief text that identifies the FOB point.
FreightTerms	VARCHAR2	Terms for paying freight charges, such as paid by shipper, collect, prepaid, and so on.
FreightTermsCode	VARCHAR2	Brief text that identifies the freight terms.
FulfillmentLineIdentifier	Number	Value that uniquely identifies a fulfillment line. Order Management can use this value to identify the fulfillment line that its referencing in the context of the change.
IntendedUseClassificationIdentifier	Number	Identifies the intended use. For tax purposes.
InventoryOrganization	VARCHAR2	Name of the inventory organization that owns the item.
InventoryOrganizationIdentifier	Number	Value that identifies the inventory organization identifier.
InvoicingRule	VARCHAR2	Name of the invoicing rule that determines when to recognize the receivable so that Order Management can invoice it.
InvoicingRuleCode	VARCHAR2	Value that identifies the invoicing rule.
IsValidConfiguration	VARCHAR2	Specify whether configuration is valid. • **true**. Configuration is valid. • **false**. Configuration isn't valid. The configurator expects that an application that calls the configurator won't submit the sales order.

Attribute	Type	Description
		The web service populates this attribute only for the root order line. The configurator populates and uses this attribute.
LatestAcceptableArrivalDate	Date	Date that specifies the latest time that the customer is willing to receive the sales order at the ship-to address.
LatestAcceptableShipDate	Date	Date that specifies the latest time that the customer is willing to ship the sales order.
OperationMode	VARCHAR2	A value of CANCEL for this attribute indicates to cancel the order line. You can use only CANCEL.
OrderedUOM	VARCHAR2	Unit of measure of the requested item, such as Unit or `Kgs`.
OriginalProductDescription	VARCHAR2	SKU (stock keeping unit) that identifies the item that Order Management fulfills.
OriginalProductIdentifier	VARCHAR2	Value that identifies the item that the customer requested or ordered. Order Management subsequently substituted this item with some other item.
OriginalProductNumber	VARCHAR2	Display name of the item that Order Management fulfills.
OverrideScheduleDateFlag	VARCHAR2	Set it to TRUE or FALSE. TRUE: you can manually override the schedule date that Global Order Promising calculates when it promises the item. FALSE: you can't do this.
PackingInstructions	VARCHAR2	Comment text for packing instructions.
ParentLineReference	VARCHAR2	Value that identifies the line that's the parent of this line in a configured item, or in any other parent and child relationship.
PartialShipAllowedFlag	VARCHAR2	Specify whether Order Management ships the items that the sales order contains in more than one shipment. • **true**. Ship in more than one shipment, if necessary. • **false**. Ship in one shipment.
PaymentTerm	VARCHAR2	The payment terms to use for this payment.
PaymentTermCode	VARCHAR2	Brief text that identifies the payment term.

Attribute	Type	Description
ProductCategory	VARCHAR2	Classifies the item for tax purposes. If your deployment doesn't use Oracle Inventory to classify an item for tax purposes, then Order Management uses the product category.
ProductDescription	VARCHAR2	Display name of the item.
ProductFiscalCategoryIdentifier	Number	Fiscal category that the tax authority uses to tax the item.
ProductNumber	VARCHAR2	SKU (stock keeping unit) that identifies the item to fulfill.
ProductType	VARCHAR2	Specify the type of transaction line. • Goods • Services
PromiseArrivalDate	Date	Date that the fulfillment process promised to the customer that the item would arrive at the ship-to address.
PromiseShipDate	Date	Date that the fulfillment process promised to the customer that the item would ship.
RequestCancelDate	Date	Date when the customer requested to cancel the source order.
RequestedArrivalDate	Date	Date that the customer specified to deliver the goods.
RequestedFulfillmentOrganizationCode	VARCHAR2	Code that identifies the requested fulfillment organization.
RequestedFulfillmentOrganizationIdentifier	Number	Value that identifies the requested fulfillment organization.
RequestedFulfillmentOrganizationName	VARCHAR2	Name of the organization that shipped the sales order.
RequestedShipDate	Date	Date that the customer specified to ship the goods.
RequestedSupplierCode	VARCHAR2	Brief text that identifies the supplier name.
RequestedSupplierName	VARCHAR2	Name of the supplier whose responsible for shipping the item. You can specify a supplier

Attribute	Type	Description
		according to a contractual obligation in a drop ship flow.
RequestedSupplierNumber	VARCHAR2	Number that identifies the supplier whose responsible for shipping the item. You can specify a supplier according to a contractual obligation in a drop ship flow.
RequestedSupplierSiteCode	VARCHAR2	Brief text that identifies the requested supplier site.
RequestingBusinessUnitIdentifier	Number	Value that identifies the requesting business unit.
RequestingBusinessUnitName	VARCHAR2	Internal organization that started or captured the transaction.
ReturnReason	VARCHAR2	Reason why the customer must return the item.
ReturnReasonCode	VARCHAR2	Brief text that identifies the return reason.
RootParentLineReference	VARCHAR2	Value that identifies the line that resides at the root of the configured item' hierarchy.
ScheduleArrivalDate	Date	Date that Order Management scheduled to deliver the goods.
ScheduleShipDate	Date	Date that Order Management scheduled to ship the goods.
ShipmentPriority	VARCHAR2	Priority that specifies the urgency to use when shipping the item.
ShipmentPriorityCode	VARCHAR2	Brief text that identifies the shipment priority.
ShippingCarrier	VARCHAR2	Name of the carrier who delivered the goods.
ShippingCarrierCode	VARCHAR2	Brief text that identifies the shipping carrier.
ShippingInstructions	VARCHAR2	Comment text for shipping instructions.
ShippingMode	VARCHAR2	Mode that Order Management used to deliver the shipment.
ShippingModeCode	VARCHAR2	Brief text that identifies the shipping mode.

Attribute	Type	Description
ShippingServiceLevel	VARCHAR2	Level of service to use when delivering the item.
ShippingServiceLevelCode	VARCHAR2	Brief text that identifies the shipping service level.
ShipSetName	VARCHAR2	Name of the shipment set.
ShipToAddress1	VARCHAR2	Address Line 1 of the ship-to party.
ShipToAddress2	VARCHAR2	Address Line 2 of the ship-to party.
ShipToAddress3	VARCHAR2	Address Line 3 of the ship-to party.
ShipToAddress4	VARCHAR2	Address Line 4 of the ship-to party.
ShipToAddressOrigSystemReference	VARCHAR2	Cross-reference value for the party site.
ShipToCity	VARCHAR2	City of the ship-to party.
ShipToContactPartyIdentifier	VARCHAR2	Value that identifies the shipping address of the ship-to contact.
ShipToContactPartyName	VARCHAR2	Name of the contact who resides at the shipping address.
ShipToContactPartyNumber	VARCHAR2	Number of the contact for the shipping address.
ShipToContactPartyOrigSystemReference	VARCHAR2	Cross-reference value for the party contact.
ShipToCounty	VARCHAR2	County of the ship-to party.
ShipToPartyIdentifier	VARCHAR2	Value that identifies the ship-to party name.
ShipToPartyName	VARCHAR2	Name of the party that receives the goods.
ShipToPartyNumber	VARCHAR2	Number that identifies the party that receives the goods.
ShipToPartySiteIdentifier	VARCHAR2	Value that identifies the party site that receives the goods, such as 1036.
ShipToPostalCode	VARCHAR2	Postal Code of the ship-to party.

Attribute	Type	Description
ShipToProvince	VARCHAR2	Province of the ship-to party.
ShipToRequestRegion	VARCHAR2	Identifies the region of the ship-to request. Global Order Promising uses this attribute to process a sales order according to region.
ShipToState	VARCHAR2	State of the ship-to party.
SourceSystemProductReference	VARCHAR2	Cross-reference value of the item.
Subinventory	VARCHAR2	Subinventory. You can update this attribute through a web service but not in the Order Management work area. Order Management doesn't use it during processing but does send it to Oracle Shipping. Global Order Promising doesn't consider it during scheduling. Order Management doesn't send it to Oracle Inventory Management for reservations.
SubInventoryCode	VARCHAR2	Brief text that identifies the subInventory.
SubstitutionAllowedFlag	VARCHAR2	Specify whether Order Management substitutes items during fulfillment. • **true**. Substitute items. • **false**. Don't substitute.
SubstitutionReason	VARCHAR2	Reason for the substitution.
SubstitutionReasonCode	VARCHAR2	Brief text that identifies the substitution reason.
SupplierAddressCity	VARCHAR2	City of the organization that supplies and ships the items.
SupplierAddressCountry	VARCHAR2	Country of the organization that supplies and ships the items.
SupplierAddressCounty	VARCHAR2	County of the organization that supplies and ships the items.
SupplierAddressLine1	VARCHAR2	Address Line 1of the organization that supplies and ships the items.
SupplierAddressLine2	VARCHAR2	Address Line 2 of the organization that supplies and ships the items.

Attribute	Type	Description
SupplierAddressLine3	VARCHAR2	Address Line 3 of the organization that supplies and ships the items.
SupplierAddressLine4	VARCHAR2	Address Line 4 of the organization that supplies and ships the items.
SupplierAddressPostalCode	VARCHAR2	Postal code of the organization that supplies and ships the items.
SupplierAddressProvince	VARCHAR2	Province of the organization that supplies and ships the items.
SupplierAddressState	VARCHAR2	State of the organization that supplies and ships the items.
TaxClassificationCode	VARCHAR2	Brief text that identifies the tax classification.
TaxExemptFlag	VARCHAR2	Specify whether to exempt the transaction from taxation. • **true**. Exempt from taxation. • **false**. Don't exempt.
TaxExemptionCertificateNumber	VARCHAR2	Number that identifies the tax exemption certificate that a taxing authority grants for a customer whose tax exempt.
TaxExemptReason	VARCHAR2	The reason to grant and take a tax exemption.
TaxExemptReasonCode	VARCHAR2	Brief text that identifies the tax exempt reason.
ThirdpartyTaxRegistration	Number	Registration number that the customer specifies in the purchase order.
TransactionBusinessCategory	VARCHAR2	Category of a transaction that a tax authority might require. For tax purposes.
TransactionCategoryCode	VARCHAR2	Brief text that identifies the transaction category. • **ORDER**. Process a new source order. • **RETURN**. Process a return of an existing sales order.
TransactionLineType	VARCHAR2	Type of line or action that resides in the source system. Some example values include Buy, Replace with, Return For Credit, Upgrade, and so on.

Attribute	Type	Description
TransactionLineTypeCode	VARCHAR2	Code that identifies the transaction line type.
UnitListPrice	Number	List price of the item prior to any discounts and adjustments.
UnitSellingPrice	Number	Selling price of the item with discounts and adjustments applied.
UserDefinedFiscClass	VARCHAR2	Classification of the transaction into different categories for tax purposes.

Other Attributes

Attribute	Description
Fulfillment Line Standardized Value	Equals the Ordered Quantity multiplied by the Unit Selling Price multiplied by the Unit Currency Conversion Rate.
Fulfillment Line Transactional Value	Equals the Ordered Quantity multiplied by the Unit Selling Price.

Attributes for Order Preferences

Attribute	Type	Description
CreateCustomerInformationFlag	VARCHAR2	Specify whether to create details for missing attributes. • **true**. Create the missing attribute. For example, if the Sold-to attribute and the Ship-to attribute are each missing, then create a Sold-to attribute and a Ship-to attribute. • **false**. Don't create the missing attribute.

Other Optional Attributes You Can Include in the Request Payload

Source Transaction Details

You can include these optional attributes in the request payload for the source transaction.

Attribute	Type	Description
SourceTransactionIdentifier	VARCHAR2	Value that uniquely identifies the transaction.

Attribute	Type	Description
SourceTransactionLineIdentifier	VARCHAR2	Value that uniquely identifies the transaction line.
SourceTransactionNumber	VARCHAR2	Transaction number.
SourceTransactionRevisionNumber	VARCHAR2	Revision number of the transaction.
SourceTransactionScheduleIdentifier	VARCHAR2	Value that uniquely identifies the transaction for a schedule, shipment, or subline. The source system assigns this value.
SourceTransactionSystem	VARCHAR2	Name of the source system that placed the request for fulfillment.
SubmitFlag	VARCHAR2	• **true**. Submit the sales order to fulfillment after you import it. Use `true` when you don't need to modify the order after you import it into Order Management and instead want to submit it directly to fulfillment. The default value is `true`. If you don't send any value for SubmitFlag, then we assume it's `true`.If `true`, and if the import fails during submit, then we don't submit the sales order but instead save it in Draft status. If you're importing a sales order revision, then you must use `true`. • **false**. Create a draft of the sales order that you're importing but don't submit it. Use `false` when you need to modify the order in Order Management after you import and before you submit it to fulfillment. You can use the SubmitDraftOrder operation in a subsequent import to submit the draft order to fulfillment. Use `false` only for a new sales order. Don't use `false` when you revise a sales order.

You typically include these attributes when you specify one of these objects.

- Lot Numbers and serial numbers
- Sales credits
- Payments
- Attachments
- Document references
- Transaction items
- Charges
- Charge components

Lot Numbers and Serial Numbers

You can include these optional attributes in the request payload to define lot numbers and serial numbers.

Attribute	Type	Description
ItemRevisionNumber	VARCHAR2	Number that identifies the revision.
LocatorIdentifier	Number	Value that identifies the locator where Order Management ships the item from or received into.
LotNumber	VARCHAR2	Number assigned to a quantity of items for identification purposes. A lot number is an identification number that a manufacturer assigns to a quantity of material, typically for quality control. Some manufacturers combine the lot number with the serial number to form an identification number.
SerialNumberFrom	VARCHAR2	Starting serial number of a range of serial numbers.
SerialNumberTo	VARCHAR2	Ending serial number of a range of serial numbers.
SourceTransactionLotIdentifier	VARCHAR2	Value that uniquely identifies the lot. The source system assigns this value.

Sales Credits

You can include these optional attributes in the request payload to define sales credits.

Attribute	Type	Description
Percent	Number	Number that specifies the sales credit percentage for a salesperson.
SalesCreditTypeCode	VARCHAR2	Brief text that identifies the sales credit type.
SalesCreditTypeReference	VARCHAR2	Foreign key that references the sales credit type. This type is Revenue or `Non Revenue`.
SalesPersonIdentifier	Number	Value that uniquely identifies the salesperson. The source system assigns this value.
SourceTransactionSalesCreditIdentifier	VARCHAR2	Value that uniquely identifies the sales credit. The source system assigns this value.

Payments

You can include these optional attributes in the request payload to define payments.

Attribute	Type	Description
PaymentMethod	VARCHAR2	Payment method that's associated with the payment instrument for the customer account.
PaymentMethodCode	VARCHAR2	Brief text that identifies the payment method.
PaymentSetIdentifier	Number	Value that uniquely identifies a group of payments that belong to one prepaid order. If the set identifies a prepayment, then the foreign key references billing.
PaymentTransactionIdentifier	Number	Value that identifies the payment details. The source system contains this value.
SourceTransactionPaymentIdentifier	VARCHAR2	Value that uniquely identifies the internal payment. The source system assigns this value.

Attachments

You can include these optional attributes in the request payload to define attachments.

Attribute	Type	Description
Data	BlobDomain	Data that the attachment contains.
DataTypeCode	VARCHAR2	Type of attachment.
Description	VARCHAR2	Description of the attachment.
FileContent	VARCHAR2	Mime type for BLOB (Binary Large OBject) attachment.
FileName	VARCHAR2	File name of the attached document. If the attachment is a URL, then Order Management doesn't use this attribute.
Title	VARCHAR2	Title of the document.
URL	VARCHAR2	URL.

Document References

You can include these optional attributes in the request payload to define document references. The web service currently accepts only the original sales order or order line reference when creating a return line for the document reference entity.

Attribute	Type	Description
DocumentAdditionalIdentifier	VARCHAR2	Value that identifies more qualifiers for the ID. Used when multipart keys are present.
DocumentAdditionalLineIdentifier	VARCHAR2	Value that identifies more qualifiers for the document line. Used when multipart keys are present.
DocumentAdditionalLineNumber	VARCHAR2	Number that identifies the document line. You can use it as another way to identify the object instance. You can use it to capture more identifying details, as necessary.
DocumentAdditionalNumber	VARCHAR2	Number that identifies the document. You can use it as another way to identify the object instance. You can use it to capture more identifying details, as necessary.
DocumentAdditionalSubLineIdentifier	VARCHAR2	Value that identifies more qualifiers for the subline. Used when multipart keys are present.
DocumentAdditionalSubLineNumber	VARCHAR2	Number that identifies the document subline. You can use it as another way to identify the object instance. You can use it to capture more identifying details, as necessary.
DocumentIdentifier	VARCHAR2	Value that uniquely identifies the document.
DocumentLineIdentifier	VARCHAR2	Value that uniquely identifies the document line. Order Management creates this value.
DocumentLineNumber	VARCHAR2	User-friendly number that identifies the document line, such as the line number in a sales order, or the line number in a purchase order.
DocumentNumber	VARCHAR2	User-friendly number that identifies the document, such as asset number, sales order number, or purchase order number.
DocumentReferenceType	VARCHAR2	Type of business document or object that the source order references, such as asset, sales order, or purchase order.
DocumentSubLineIdentifier	VARCHAR2	Value that uniquely identifies the document subline. Order Management creates this value.

Attribute	Type	Description
DocumentSubLineNumber	VARCHAR2	User-friendly number that identifies the subline.

Transaction Items

You can include these optional attributes in the request payload to define transaction items.

Attribute	Type	Description
CharacterValue	VARCHAR2	Item attribute value of type character.
DateValue	Date	Item attribute value of type date.
NumberValue	Number	Item attribute value of type number.
TimestampValue	Timestamp	Item attribute value of type time.
TransactionAttributeName	VARCHAR2	Item attribute name.

Charges

You can include these optional attributes in the request payload to define charges.

Attribute	Type	Description
ApplyTo	long	Specify whether to apply a charge to the item, shipping, or return.
BatchIdentifier	Number	Number that identifies the batch.
ChargeDefinition	VARCHAR2	Value for the charge definition entity. A charge definition defines the price type, charge type, and the charge subtype. Order Management typically denormalizes these objects on this entity.
ChargeDefinitionCode	VARCHAR2	Brief text that identifies the charge definition.
ChargeSubType	VARCHAR2	Type of charge, defined for this configuration to aggregate totals. • Goods sale • Service sale

Attribute	Type	Description
		• Financing compared to lease • Shipping charges • Restocking penalties • Special charges The Pricing Engine returns the charge value for each line.
ChargeSubtypeCode	VARCHAR2	Brief text that identifies the charge subtype.
ParentEntityCode	VARCHAR2	Parent entity of the charge. • Line • Line Coverage
ParentEntityId	Number	ID of the parent entity of the charge.
PricedQuantity	Number	The priced quantity. This quantity equals Line.RequestedQuantity for the item.
PricedQuantityUOMCode	VARCHAR2	Brief text that identifies the UOM for the priced quantity. For example, Ton. Values for this attribute come from PIM (Product Information Management), or a similar service that Order Management provides.
PriceTypeCode	VARCHAR2	Price type of a charge. • One-time • Recurring
PrimaryFlag	string	Specify whether this charge is the primary charge. • **true**. Primary charge. • **false**. Not the primary charge.
RollupFlag	VARCHAR2	Specify whether this charge is a rollup charge or an aggregate charge. • **true**. Rollup charge. • **false**. Aggregate charge.
SourceChargeIdentifier	VARCHAR2	Value that uniquely identifies the charge. The order capture system assigns this value.

Charge Components

You can include these optional attributes in the request payload to define charge components.

Attribute	Type	Description
ChargeCurrencyCode	VARCHAR2	Brief text that identifies the currency that the charge component uses. Order Management uses this code to standardize service.
ChargeCurrencyUnitPrice	Number	Price or adjustment for each unit in the charge currency for the UOM that the order line uses.
HeaderCurrencyCode	VARCHAR2	Brief text that identifies the currency that the header uses. Order Management uses this code to standardize service.
HeaderCurrencyExtendedAmount	Number	Extended amount in the header currency.
HeaderCurrencyUnitPrice	Number	Price or adjustment for each unit in the charge currency for the UOM that the header uses.
PriceElementCode	Number	Brief text that identifies the price element, such as LISTPRICE, NETPRICE, and so on.
PriceElementUsageCode	VARCHAR2	Brief text that specifies how to use the charge component. A QP lookup provides the values for this attribute. Order Management comes predefined to use one of these values. • List Price • Invoice Price
RollupFlag	VARCHAR2	Specifies a charge component as a rollup value or aggregate value for the element code of the charge component price.
SequenceNum	Number	Sequence number for the charge component.
SourceChargeComponentIdentifier	VARCHAR2	Value that uniquely identifies the charge component. The source system assigns this value.
SourceChargeIdentifier	VARCHAR2	Value that uniquely identifies the charge. The source system assigns this value.
SourceParentChargeComponentId	VARCHAR2	Identifier for the charge component for the contributing charge. Order Management uses this attribute only for a charge component where the parent charge is a rollup charge.

Related Topics

- Manage Lookups in Order Management

Operations and Attributes You Can Use with the Receive Order Request Service

Get details about operations and attributes you can use with the Receive Order Request Service when you integrate Order Management with other systems.

Operation	Description	Input Payload
ProcessOrderRequest	Submit a sales order to start a transformation.	Not applicable
SubmitDraftOrder	Create a sales order in a draft status, or create and submit a sales order according to the submitFlag in the SDO.	Not applicable
RequestHold	Request to put the sales order or fulfillment process on hold.	RequestHoldProcessRequest
ReleaseHold	Release a hold that's currently holding a sales order or fulfillment process.	ReleaseHoldProcessRequest
GetAvailabilityCheck	Get the supply of an item that's currently available in an organization or supplier.	GetAvailabilityCheckProcessRequ
ReleasePausedTasks	Release paused tasks according to a combination of search parameters.	ReleasePausedEventTaskRequest
CheckAvailability	Allow an order capture system to get, view, and analyze the availability of a sales order item and the promising options for this item.	CheckAvailabilityInput
ProcessOrderRequestSync	Submit a sales order to start transformation.	Not applicable
SubmitDraftOrderSync	Create a sales order in a draft status, or create and submit a sales order according to the submitFlag in the SDO.	Not applicable
GetAvailabilityCheckSync	Get the supply of an item that's currently available in an organization or supplier.	GetAvailabilityCheckProcessRequ

ReleasePausedTasks Operation

The ReleasePausedTasks operation releases paused tasks according to a combination of search parameters. You can use it only as an asynchronous web service. You must include these attributes in a request that uses ReleasePausedTasks.

Attribute	Type	Required	Description
SourceSystem	String	Yes	Source system that provides the release pause request.

Attribute	Type	Required	Description
EventName	String	Yes	Name of the pause event to release.
InventoryItemId	Long	Yes	Item identifier.
OrderNumber	String	Yes	Sales order number.
LineNumber	Long	Yes	Order line number.
FulfillLineNumber	Long	Yes	Number of the order fulfillment line.
SoldToCustomerId	Long	Yes	Customer identifier.
FulfillOrgId	Long	Yes	Warehouse identifier.
PauseTaskId	Long	Yes	Paused task identifier.
FromOrderDate	DateTime	Yes	Filter sales orders that happen on or after FromOrderDate.
ToOrderDate	DateTime	Yes	Filter sales orders that happen on or before ToOrderDate.
FromScheduledShipDate	DateTime	Yes	Filter sales orders that are scheduled to ship on or after FromScheduledShipDate.
ToScheduledShipDate	DateTime	Yes	Filter sales orders that are scheduled to ship on or before ToScheduledShipDate.
FromPauseWaitUntilDate	DateTime	Yes	Filter pause tasks that are scheduled to release on or after FromPauseWaitUntilDate.
ToPauseWaitUntilDate	DateTime	Yes	Filter pause tasks that are scheduled to release on or before ToPauseWaitUntilDate.
SourceOrderSystem	String	Yes	Source system that provides the source order.
SourceOrderNumber	String	Yes	Order number in the source system.

Attribute	Type	Required	Description
TaskInstanceId	Long	Yes	Task Instance Identifier.

ReleasePausedTasks provides this response.

Attribute	Type	Description
NumberOfTasksReleased	Long	Number of tasks released.
ErrorMessage	String	Error message if the service doesn't complete successfully.
ReturnStatus	String	Return status.

CheckAvailability Operation

The CheckAvailability operation allows your order capture system to get, view, and analyze the availability of a sales order item and the promising options for this item. You can use it only as an asynchronous web service. An asterisk (*) in the Required column indicates a group of attributes. You must include at least one attribute from the group.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
CheckAvailabilityInput SourceOrderSystem	SourceOrderSystem	String	Yes	Source system.
CheckAvailabilityInput PromisingSet	Promising Set	Group	No	Promising set.
CheckAvailabilityInput PromisingLine	Promising Line	Group	No	List of promising line attributes in the shipment set. Each promising line contains the attributes of the promising line attribute.
CheckAvailabilityInput PromisingModel	PromisingModel	Group	No	Groups promising lines into a configure-to order (CTO) model.
CheckAvailabilityInput PromisingSet	Promising Set	Group	No	Groups promising lines into a shipment set.
CheckAvailabilityInput PromisingSet SetName	SetName	String	Conditional	Name of the shipment set. Required only if you also use promising set.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
CheckAvailabilityInput PromisingSet PromisingLine	Promising Line	Group	No	List of promising line attributes that the shipment set contains. Each promising line contains the attributes of the promising line attribute.
CheckAvailabilityInput PromisingModel	PromisingModel	Group	No	Groups promising lines into a configure-to order (CTO) model.
CheckAvailabilityInput PromisingModel RootParentFulfillId	RootParentFulfillId	String	Conditional	Root parent line of the model. Required only if you also use the promising model.
CheckAvailabilityInput PromisingModel ModelType	ModelType	String	Conditional	Type of model. Valid values include ATO (assemble-to order) or PTO-SMC. Required only if you also use the promising model.
CheckAvailabilityInput PromisingModel IncludedItemsFlag	IncludedItemsFlag	Boolean	No	Indicates whether to provide items of the pick-to order (PTO) model as input. Valid values include True or False. If False, Order Management determines the items. Default value is True.
CheckAvailabilityInput PromisingModel PromisingLine	Promising Line	Group	No	List of promising line attributes that the model contains. Each promising line includes the same set of attributes that the promising line attribute contains.
CheckAvailabilityInput PromisingLine	PromisingLine	Group	No	Groups the list of attributes that you can specify for the check availability service.
CheckAvailabilityInput PromisingLine PromisingLineIdentifier	PromisingLineIdentifier	String	Yes	Unique identifier for the order line.
CheckAvailabilityInput PromisingLine MasterOrganization	MasterOrganization	String	No	Not used.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
CheckAvailabilityInput PromisingLine InventoryOrganization	InventoryOrganization	String	Yes	Item validation organization.
CheckAvailabilityInput PromisingLine ProductName	ProductName	String	No	Not used.
CheckAvailabilityInput PromisingLine RequestedItemId	RequestedItemId	String	Yes	Requested Item.
CheckAvailabilityInput PromisingLine PromisingType	PromisingType	String	Yes	Used with the requested date to determine whether its the Requested Ship Date from the warehouse or the Requested Delivery Date for the ship-to address. Valid values include Ship or Arrival.
CheckAvailabilityInput PromisingLine RequestedDate	RequestedDate	Date	Yes	Date when the item is requested to ship or deliver.
CheckAvailabilityInput PromisingLine RequestedQuantity	RequestedQuantity	Double	Yes	Requested quantity of the item.
CheckAvailabilityInput PromisingLine RequestedQuantityUOM	RequestedQuantityUOM	String	Yes	Unit of measure in the item quantity that's requested.
CheckAvailabilityInput PromisingLine DemandClass	DemandClass	String	No	Demand class of the order line.
CheckAvailabilityInput PromisingLine DeliveryLeadTime	DeliveryLeadTime	Double	No	Default value for the delivery lead time to use when calculating the ship date or arrival date. This value applies only if you don't specify Carrier, Mode, or Service Level. Used in conjunction with DeliveryCostPerUnit.
CheckAvailabilityInput PromisingLine DeliveryCostPerUnit	DeliveryCostPerUnit	Double	No	Delivery cost for each unit of the delivered item. This value applies only if you don't specify Carrier, Mode, or Service Level.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
				Used in conjunction with DeliveryLeadTime.
CheckAvailabilityInput PromisingLine UnitPrice	UnitPrice	Double	Yes	Unit price of the requested item.
CheckAvailabilityInput PromisingLine EarliestAcceptableDate	EarliestAcceptableDate	Date	No	Don't use this attribute.
CheckAvailabilityInput PromisingLine LatestAcceptableDate	LatestAcceptableDate	Date	No	Latest date when the item can ship or deliver.
CheckAvailabilityInput PromisingLine RequestedCarrier	RequestedCarrier	String	No	Carrier to use for the shipment.
CheckAvailabilityInput PromisingLine RequestedServiceLevel	RequestedServiceLevel	String	No	Level of service of the shipment.
CheckAvailabilityInput PromisingLine RequestedMode	RequestedMode	String	No	Mode of transport of the shipment.
CheckAvailabilityInput PromisingLine SubstitutionsAllowed	SubstitutionsAllowed	Boolean	Yes	Determines whether to allow substitutions. Valid values include True or False.
CheckAvailabilityInput PromisingLine SplitsAllowed	SplitsAllowed	Boolean	Yes	Determines whether the line can split into smaller quantities across dates or source of supply for fulfilling the request. Valid values include True or False.
CheckAvailabilityInput PromisingLine GenerateAlternateAvailability	GenerateAlternateAvailability	Boolean	Yes	Determines whether to create alternative availability options. Valid values include True or False.
CheckAvailabilityInput PromisingLine AlternateAvailabilityBasis	AlternateAvailabilityBasis	String	No	Used to sort the alternate options according to fastest delivery or cost. Valid values include Delivery or Cost. Default value is Delivery.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
CheckAvailabilityInput PromisingLine GeneratePegging	GeneratePegging	Boolean	Yes	Determines whether to create pegging. Valid values include True or False.
CheckAvailabilityInput PromisingLine MaxNumberOfAvailabilities	MaxNumberOfAvailabilities	Integer	No	Maximum number of alternative availability options to provide.
CheckAvailabilityInput PromisingLine MinPromiseQuantity	MinPromiseQuantity	Double	No	Not used currently.
CheckAvailabilityInput PromisingLine MinPromisePercentage	MinPromisePercentage	Integer	No	Not used currently.
CheckAvailabilityInput PromisingLine RequestedDropShipSupplier	RequestedDropShipSupplier	String	No	Supplier selected to fulfill the request. You can specify only one supplier or one warehouse.
CheckAvailabilityInput PromisingLine RequestedDropShipSupplier	RequestedDropShipSupplier	String	No	Site of the supplier. You can specify this value only if you also select the supplier.
CheckAvailabilityInput PromisingLine InternalOrderType	InternalOrderType	String	No	Type of internal order. Valid values include TO or ISO.
CheckAvailabilityInput PromisingLine ParentFulfillId	ParentFulfillId	String	Conditional	Parent line of the current order line. Used only for configure-to-order (CTO) models. If the Promising Line resides in a promising model, then this attribute is required.
CheckAvailabilityInput PromisingLine ConfiguredItem	ConfiguredItem	String	No	Configuration item of an assemble-to-order (ATO) model. Used only for configure-to-order (CTO) models.
CheckAvailabilityInput PromisingLine DestinationOrgId	DestinationOrgId	String	No	Destination organization of an order.
CheckAvailabilityInput PromisingLine RequestedShipFromOrg InstanceId	InstanceId	String	No	Instance where the warehouse is defined.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
CheckAvailabilityInput PromisingLine RequestedShipFromOrg OrgId	OrgId	String	No	Warehouse (inventory organization) selected to fulfill the request. You can specify only one supplier or one warehouse.
CheckAvailabilityInput PromisingLine MinSplitQuantity	MinSplitQuantity	Double	No	Minimum quantity that must be available in the first delivery when splitting the order line.
CheckAvailabilityInput PromisingLine MinSplitPercentage	MinSplitPercentage	Integer	No	Minimum quantity as a percentage of the ordered quantity that must be available in the first delivery when splitting the order line.
CheckAvailabilityInput PromisingLine CustomerAccountId	CustomerAccountId	String	No	Use Party. Don't use Sold-to customer.
CheckAvailabilityInput PromisingLine CustomerShipTo ShipToSiteId	ShipToSiteId	String	No	Use ShipTo PartySite. Don't use Ship-to customer Site.
CheckAvailabilityInput PromisingLine CustomerShipTo RequestedRegion	RequestedRegion	String	Yes*	Region that receives the item. Sold To Party and Ship To Party Site, or RequestedRegion, is required.
CheckAvailabilityInput PromisingLine PartyInfo PartyId	PartyId	String	Yes*	Sold-to party. If you provide Sold-to Party, then you must also provide Ship-to Party Site.
CheckAvailabilityInput PromisingLine PartyInfo ShipToPartySiteId	ShipToPartySiteId	String	Yes*	Ship-to party site. Sold To Party and Ship To Party Site, or RequestedRegion, is required.

CheckAvailability provides this response.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineSetReply	PromisingLineSetReply	Group	Groups the promising result for promising lines in a shipment set.
checkAvailabilityOutput PromisingResult PromisingModelReply	PromisingModelReply	Group	Groups the promising result for promising lines in a configure-to order (CTO) model.
checkAvailabilityOutput PromisingResult PromisingLineReply	PromisingLineReply	Group	Groups the promising result for a promising line.
checkAvailabilityOutput PromisingResult PromisingLineSetReply	PromisingLineSetReply	Group	Groups the promising result for promising lines in a shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply SetId	SetId	String	Name of the shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply NumberOfAltOptions	NumberOfAltOptions	Integer	Number of alternate availability options.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption	PromisingLineSetReply > Default AvailabilitySetOption	Group	Groups the default availability option of the promising result for promising lines in a shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption PromisingLineSetResult ExpectedGroupShipDate	ExpectedGroupShipDate	Date	Expected date when the shipment set will ship.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption PromisingLineSetResult ExpectedGroupArrivalDate	ExpectedGroupArrivalDate	Date	Not used.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption PromisingLineSetResult PromisingLineResult	PromisingLineResult	Group	Promising result for the list of promising line attributes in the shipment set.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions	PromisingLineSetReply AlternateAvailabilitySetOptions	Group	Groups the alternate availability options of the promising result for promising lines in a shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions PromisingLineSetResult ExpectedGroupShipDate	ExpectedGroupShipDate	Date	Date when the shipment set is expected to ship.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption PromisingLineSetResult ExpectedGroupArrivalDate	ExpectedGroupArrivalDate	Date	Not used.
checkAvailabilityOutput PromisingResult PromisingLineSetReply DefaultAvailabilitySetOption PromisingLineSetResult PromisingLineResult	PromisingLineResult	Group	Promising result for the list of promising line attributes in the shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions	PromisingLineSetReply > AlternateAvailabilitySetOptions	Group	Groups the alternate availability options of the promising result for promising lines in a shipment set.
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions PromisingLineSetResult ExpectedGroupShipDate	ExpectedGroupShipDate	Date	Date when the shipment set is expected to ship.
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions PromisingLineSetResult ExpectedGroupArrivalDate	ExpectedGroupArrivalDate	Date	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineSetReply AlternateAvailabilitySetOptions PromisingLineSetResult PromisingLineResult	PromisingLineResult	Group	Promising result for the list of promising line attributes in the shipment set.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingModelReply	PromisingModelReply	Group	Groups the promising result for promising lines in a configure-to order (CTO) model.
checkAvailabilityOutput PromisingResult PromisingModelReply RootParentFulfillId	RootParentFulfillId	String	Root parent of the model.
checkAvailabilityOutput PromisingResult PromisingModelReply NumberOfAltOptions	NumberOfAltOptions	Integer	Number of alternate availability options.
checkAvailabilityOutput PromisingResult PromisingModelReply DefaultAvailabilityModelOption	PromisingModelReply->DefaultAvailabilityModelOption	Group	Groups the default availability option of the promising result for promising lines in a configure-to order (CTO) model.
checkAvailabilityOutput PromisingResult PromisingModelReply DefaultAvailabilityModelOption PromisingLineModelResult RootParentFulfillId	RootParentFulfillId	String	Root parent of the model.
checkAvailabilityOutput PromisingResult PromisingModelReply DefaultAvailabilityModelOption PromisingLineModelResult ErrorCode	ErrorCode	String	Error code to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingModelReply DefaultAvailabilityModelOption PromisingLineModelResult ErrorMessage	ErrorMessage	String	Error message to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingModelReply DefaultAvailabilityModelOption PromisingLineModelResult PromisingLineResult	PromisingLineResult	Group	Promising result for the list of promising line attributes in the configure-to order (CTO) model.
checkAvailabilityOutput PromisingResult PromisingModelReply AlternateAvailabilityModelOptions	PromisingModelReply->AlternateAvailabilityModelOptions	Group	Groups the alternate availability options of the promising result for promising lines in a configure-to order (CTO) model.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingModelReply AlternateAvailabilityModelOptions PromisingLineModelResult RootParentFulfillId	RootParentFulfillId	String	Root parent of the model.
checkAvailabilityOutput PromisingResult PromisingModelReply AlternateAvailabilityModelOptions PromisingLineModelResult ErrorCode	ErrorCode	String	Error code to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingModelReply AlternateAvailabilityModelOptions PromisingLineModelResult ErrorMessage	ErrorMessage	String	Error message to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingModelReply AlternateAvailabilityModelOptions PromisingLineModelResult PromisingLineResult	PromisingLineResult	Group	Promising result for the list of promising line attributes in the configure-to order (CTO) model.
checkAvailabilityOutput PromisingResult PromisingLineReply	PromisingLineReply	Group	Groups the promising result for a promising line.
checkAvailabilityOutput PromisingResult PromisingLineReply NumberOfAltOptions	NumberOfAltOptions	Integer	Number of alternate availability options.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption	PromisingLineReply->DefaultAvailabilityOption	Group	Groups the default availability option of the promising result for a promising line.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption OptionRank	OptionRank	Integer	Rank of the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption OptionSummary	OptionSummary	String	Determines whether the summary of the availability option is available.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult	PromisingLineResult	Group	Groups the alternate availability options of the promising result for a promising line.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult PromisingLineIdentifier	PromisingLineIdentifier	String	Line identifier.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult TotalPrice	TotalPrice	Double	Total price of the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult TotalProfit	TotalProfit	Double	Total profit when using the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult TotalMargin	TotalMargin	Double	Total margin when using the availability.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult TotalValue	TotalValue	Double	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult NumberOfSplits	NumberOfSplits	Integer	Number of split shipments that the availability option recommends.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult NumberOfSubstitutions	NumberOfSubstitutions	Integer	Number of item substitutions that the availability option recommends.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult FillRate	FillRate	Double	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult NumberOfAtpItems	NumberOfAtpItems	Integer	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult NumberOfCtpItems	NumberOfCtpItems	Integer	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult SourcingRule	SourcingRule	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail CustomerAccountId	CustomerAccountId	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail CustomerShipToSiteId	CustomerShipToSiteId	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail RequestedRegion	RequestedRegion	String	Region that requested the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail PartyId	PartyId	String	Sold-to party of the customer.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ShipToPartySiteId	ShipToPartySiteId	String	Ship-to party site of the customer.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail InternalSalesOrderDestOrg InstanceId	InstanceId	String	Instance where the warehouse is defined.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail InternalSalesOrderDestOrg OrgId	OrgId	String	Warehouse that supplies the item that will ship.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail PromisingStatus	PromisingStatus	String	Determines whether the request is met completely or not completely.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail RequestedDate	RequestedDate	Date	Date when the item is requested to ship or deliver.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail PromisingType	PromisingType	String	Determines whether to set the requested date to the Requested Ship Date from the warehouse, or to the Requested Delivery Date to the ship-to address. Valid Values include Ship or Arrival.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedArrivalDate	ExpectedArrivalDate	Date	Expected date to deliver the item to the customer address.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption	ExpectedShipDate	Date	Expected date to ship the item from the warehouse or the supplier.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
PromisingLineResult ResultDetail ExpectedShipDate			
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedPickDate	ExpectedPickDate	Date	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail RequestedItemId	RequestedItemId	String	Item requested.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedAvailableItem	ExpectedAvailableItem	String	Expected item to ship. If an item substitution happens, then this attribute references the item substitution instead of the requested item.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedAvailabilityOnRequestedDa	ExpectedAvailabilityOnRequestedDa	Double	Expected item availability on the request date.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedAvailableQuantity	ExpectedAvailableQuantity	Double	Expected item quantity that's available through the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedAvailableQuantityUOM	ExpectedAvailableQuantityUOM	String	Unit of Measure of the Expected Available Quantity.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedDropShipSupplier	ExpectedDropShipSupplier	String	Supplier that the availability option recommends to supply the item that ships. The availability option recommends only one warehouse or one supplier at one point in time.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedDropShipSupplierSite	ExpectedDropShipSupplierSite	String	Site of the supplier that the availability option recommends to ship the item.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedMode	ExpectedMode	String	Mode of transport that the availability option recommends for the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedService	ExpectedService	String	Level of service that the availability option recommends for the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedCarrier	ExpectedCarrier	String	Carrier who ships the item.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail CarrierCalendar	CarrierCalendar	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedDemandClass	ExpectedDemandClass	String	Demand class.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedTotalFulfilmentCost	ExpectedTotalFulfilmentCost	Double	Expected total fulfillment cost.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedShippingCost	ExpectedShippingCost	Double	Expected shipping cost.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ExpectedMargin	ExpectedMargin	Double	Expected margin.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail Price	Price	Double	Unit price of the item.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail Profit	Profit	Double	Profit projected.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail Value	Value	Double	Value projected.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail LineFillRate	LineFillRate	Double	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ErrorCode	ErrorCode	String	Error code to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption PromisingLineResult ResultDetail ErrorMessage	ErrorMessage	String	Error message to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingLineReply DefaultAvailabilityOption	ExpectedShipFromOrg	String	Warehouse that the availability option recommends to ship the item.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
PromisingLineResult ResultDetail ExpectedShipFromOrg			
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions	PromisingLineReply->AlternateAvailabilityOptions	Group	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions OptionRank	OptionRank	Integer	Rank of the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions OptionSummary	OptionSummary	String	Determines whether the summary of the availability option is available.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult`	PromisingLineResult	Group	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult PromisingLineIdentifier	PromisingLineIdentifier	String	Line identifier.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult TotalPrice	TotalPrice	Double	Total price of the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult TotalProfit	TotalProfit	Double	Total profit when using the availability option.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult TotalMargin	TotalMargin	Double	Total margin when using the availability.
checkAvailabilityOutput PromisingResult	TotalValue	Double	Not applicable

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
PromisingLineReply AlternateAvailabilityOptions PromisingLineResult TotalValue			
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult NumberOfSplits	NumberOfSplits	Integer	Number of split shipments that the availability option recommends.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult NumberOfSubstitutions	NumberOfSubstitutions	Integer	Number of item substitutions that the availability option recommends.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult FillRate	FillRate	Double	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult NumberOfAtpItems	NumberOfAtpItems	Integer	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult NumberOfCtpItems	NumberOfCtpItems	Integer	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult SourcingRule	SourcingRule	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail CustomerAccountId	CustomerAccountId	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply	CustomerShipToSiteId	String	Not applicable

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
AlternateAvailabilityOptions PromisingLineResult ResultDetail CustomerShipToSiteId			
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail RequestedRegion	RequestedRegion	String	Region that requested the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail PartyId	PartyId	String	Sold-to party of the customer.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ShipToPartySiteId	ShipToPartySiteId	String	Ship-to party site of the customer.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail InternalSalesOrderDestOrg InstanceId	InstanceId	String	Instance where the warehouse is defined.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail InternalSalesOrderDestOrg OrgId	OrgId	String	Warehouse that supplies the item that will ship.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail PromisingStatus	PromisingStatus	String	Determines whether the request is met completely or not completely.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail RequestedDate	RequestedDate	Date	Date when the item is requested to ship or deliver.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail PromisingType	PromisingType	String	Determines whether to set the requested date to the Requested Ship Date from the warehouse, or to the Requested Delivery Date to the ship-to address. Valid Values include Ship or Arrival.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedArrivalDate	ExpectedArrivalDate	Date	Expected date to deliver the item to the customer address.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedShipDate	ExpectedShipDate	Date	Expected date to ship the item from the warehouse or the supplier.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedPickDate	ExpectedPickDate	Date	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail RequestedItemId	RequestedItemId	String	Item requested.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedAvailableItem	ExpectedAvailableItem	String	Expected item to ship. If an item substitution happens, then this attribute references the item substitution instead of the requested item.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedAvailabilityOnRequestedDa	ExpectedAvailabilityOnRequestedDa	Double	Expected item availability on the request date.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedAvailableQuantity	ExpectedAvailableQuantity	Double	Expected item quantity that's available through the availability option.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedAvailableQuantityUOM	ExpectedAvailableQuantityUOM	String	Unit of Measure of the Expected Available Quantity.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedDropShipSupplier	ExpectedDropShipSupplier	String	Supplier that the availability option recommends to supply the item that ships. The availability option recommends only one warehouse or one supplier at one point in time.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedDropShipSupplierSite	ExpectedDropShipSupplierSite	String	Site of the supplier that the availability option recommends to ship the item.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedMode	ExpectedMode	String	Mode of transport that the availability option recommends for the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedService	ExpectedService	String	Level of service that the availability option recommends for the shipment.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedCarrier	ExpectedCarrier	String	Carrier who ships the item.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail CarrierCalendar	CarrierCalendar	String	Not applicable
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions	ExpectedDemandClass	String	Demand class.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
PromisingLineResult ResultDetail ExpectedDemandClass			
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedTotalFulfilmentCost	ExpectedTotalFulfilmentCost	Double	Expected total fulfillment cost.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedShippingCost	ExpectedShippingCost	Double	Expected shipping cost.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedMargin	ExpectedMargin	Double	Expected margin.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail Price	Price	Double	Unit price of the item.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail Profit	Profit	Double	Projected profit.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail Value	Value	Double	Projected value.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail LineFillRate	LineFillRate	Double	Not applicable
checkAvailabilityOutput PromisingResult	ErrorCode	String	Error code to use if an error happens.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ErrorCode			
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ErrorMessage	ErrorMessage	String	Error message to use if an error happens.
checkAvailabilityOutput PromisingResult PromisingLineReply AlternateAvailabilityOptions PromisingLineResult ResultDetail ExpectedShipFromOrg	ExpectedShipFromOrg	String	Warehouse that the availability option recommends to ship the item.

GetAvailabilityCheck and GetAvailabilityCheckSync Operations

The GetAvailabilityCheck operation and the GetAvailabilityCheckSync operation gets the supply of an item that's currently available in an organization or supplier. You can use them as a synchronous or asynchronous web service. You must include these required attributes in a request that uses GetAvailabilityCheck or GetAvailabilityCheckSync.

Attribute	Type	Required	Description
SourceOrderSystem	String	Yes	Source order system.
MasterOrganization	String	Yes	Item validation organization.
BusinessUnit	String	Yes	Business unit.
ItemEntry	Group	No	One or more repetitions.
ItemEntry	Group	No	One or more repetitions.
ItemId	String	Yes	Item to reference when determining supply availability.
RequestedDate	Date	Yes	Date when the supply availability is requested.
DestinationOrgId	String	No	Destination organization of a sales order.
SupplierId	String	No	Supplier to reference when determining supply availability.

Attribute	Type	Required	Description
SupplierSiteId	String	No	Supplier site to reference when determining supply availability. You can specify the supplier site only if you also specify the supplier.
OrgInput	Group	No	Zero or more repetitions.
OrgInput	Not applicable	No	Group that captures the warehouse.
OrgId	String	No	Warehouse (inventory organization) that requires the availability of the supply for the item.

GetAvailabilityCheck or GetAvailabilityCheckSync provides this response.

Attribute	Type	Description
ErrorMessage	String	Error message that displays if a problem happens in the data or in the setup.
InvalidItems	Group	One or more repetitions.
PromisingSystem	String	Name of the promising system.
PromisingInstance	String	Instance of the promising system.
PromiseDate	Date	Date when the promise is created.
ItemAvailability	Group	One or more repetitions.
InvalidItems	Group	Group that identifies invalid items.
ItemId	String	Item identifier.
ItemAvailability	Group	Group that includes availability details for one item.
ItemId	String	Identifies the item that this operation references when it determines how much supply is available to fulfill this item.
RequestedDate	Date	Date when supply availability is requested.

Attribute	Type	Description
ErrorCode	String	Error code.
ErrorMessage	String	Error message that displays if a problem happens in the data or in the setup.
InstanceId	String	Instance identifier.
OrgId	String	Warehouse (inventory organization) that stores the item. If the request doesn't include a warehouse or supplier, then this operation calculates the supply availability for all warehouses and suppliers.
ShelfQty	Double	Total supply that's available for the item. Its the cumulative supply minus the cumulative supply that's consumed.
AvailableQty	Double	Total supply that's available for the item and that's not allocated to demand. Its the cumulative supply minus the cumulative demand.

RequestHold Operation

The RequestHold operation put the sales order or fulfillment process on hold. You can use it only as an asynchronous web service. You must include these required attributes in a request that uses RequestHold.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
RequestHoldProcessReques ApplyHoldRequestParams	ApplyHoldRequestParams	Group	No	One or more repetitions.
RequestHoldProcessReques ApplyHoldRequestParams	ApplyHoldRequestParams	Group	No	Not applicable
RequestHoldProcessReques ApplyHoldRequestParams SourceOrderSystem	SourceOrderSystem	string	Yes	Source system that provides the source order.
RequestHoldProcessReques ApplyHoldRequestParams SourceOrderId	SourceOrderId	string	Yes	Identifier of the source order in the source system.
RequestHoldProcessReques ApplyHoldRequestParams SourceLineId	SourceLineId	string	No	Order line identifier in the source system.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
RequestHoldProcessRequest ApplyHoldRequestParams SourceHoldCode	SourceHoldCode	string	Yes	Hold code that's requested.
RequestHoldProcessRequest ApplyHoldRequestParams HoldComments	HoldComments	string	No	Comments for the hold request.

RequestHold provides this response.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
RequestHoldProcessResponse ErrorMessage	ErrorMessage	String	Error message for the call.
RequestHoldProcessResponse ApplyHoldResponseParams	ApplyHoldResponseParams	Group	One or more repetitions.
RequestHoldProcessResponse ApplyHoldResponseParams	ApplyHoldResponseParams	Group	Group that contains the results of the requestHold operation.
RequestHoldProcessResponse ApplyHoldResponseParams SourceOrderSystem	SourceOrderSystem	String	Source system that provides the source order.
RequestHoldProcessResponse ApplyHoldResponseParams SourceOrderId	SourceOrderId	String	Order identifier in the source system.
RequestHoldProcessResponse ApplyHoldResponseParams SourceLineId	SourceLineId	String	Order line identifier in the source system.
RequestHoldProcessResponse ApplyHoldResponseParams SourceHoldCode	SourceHoldCode	String	Hold code that was requested.
RequestHoldProcessResponse ApplyHoldResponseParams RequestStatus	RequestStatus	String	Status of the call.
RequestHoldProcessResponse ApplyHoldResponseParams HoldErrorMessages	HoldErrorMessages	Group	Zero or more repetitions.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
RequestHoldProcessResponse ApplyHoldResponseParams HoldErrorMessages	HoldErrorMessages	Group	Group that contains the order line Id and the error message for this line Id record.
RequestHoldProcessResponse ApplyHoldResponseParams HoldErrorMessages DooLineId	DooLineId	String	Order line identifier in Order Management.
RequestHoldProcessResponse ApplyHoldResponseParams HoldErrorMessages ErrorMessage	ErrorMessage	String	Error message for each line.

ReleaseHold Operation

The ReleaseHold operation releases a hold that's currently holding a sales order or fulfillment process. You can use it only as an asynchronous web service. You must include these required attributes in a request that uses ReleaseHold.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
ReleaseHoldProcessRequest ReleaseHoldRequestParams	ReleaseHoldRequestParams	Group	No	One or more repetitions.
ReleaseHoldProcessRequest ReleaseHoldRequestParams	ReleaseHoldRequestParams	Group	No	Groups the request details for the release hold.
ReleaseHoldProcessRequest ReleaseHoldRequestParams SourceOrderSystem	SourceOrderSystem	String	Yes	Source system that provides the source order.
ReleaseHoldProcessRequest ReleaseHoldRequestParams SourceOrderId	SourceOrderId	String	Yes	Order identifier in the source system.
ReleaseHoldProcessRequest ReleaseHoldRequestParams SourceLineId	SourceLineId	String	No	Order line identifier in the source system.
ReleaseHoldProcessRequest ReleaseHoldRequestParams SourceHoldCode	SourceHoldCode	String	Yes	Hold code to release.
ReleaseHoldProcessRequest ReleaseHoldRequestParams HoldReleaseReasonCode	HoldReleaseReasonCode	String	Yes	Reason code for the release of the hold.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Required	Description
ReleaseHoldProcessRequest ReleaseHoldRequestParams HoldReleaseComments	HoldReleaseComments	String	No	Comments that describe the release reason.

ReleaseHold provides this response.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
ReleaseHoldProcessResponse ErrorMessage	ErrorMessage	String	Error message for the call.
ReleaseHoldProcessResponse ReleaseHoldResponseParams	ReleaseHoldResponseParams	Group	One or more repetitions.
ReleaseHoldProcessResponse ReleaseHoldResponseParams	ReleaseHoldResponseParams	Group	One or more repetitions.
ReleaseHoldProcessResponse ReleaseHoldResponseParams SourceOrderSystem	SourceOrderSystem	String	Source system that provides the source order.
ReleaseHoldProcessResponse ReleaseHoldResponseParams SourceOrderId	SourceOrderId	String	Order identifier in the source system.
ReleaseHoldProcessResponse ReleaseHoldResponseParams SourceHoldCode	SourceHoldCode	String	Hold code that was released.
ReleaseHoldProcessResponse ReleaseHoldResponseParams SourceLineId	SourceLineId	String	Order line identifier in the source system.
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldReleaseStatus	HoldReleaseStatus	String	Status of the call.
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldReleaseDate	HoldReleaseDate	String	Date when the hold was released.
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldErrorMessages	HoldErrorMessages	Group	Zero or more repetitions.

Fully Qualified Name of the Payload Attribute	Attribute	Type	Description
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldErrorMessages	HoldErrorMessages	Group	Group that contains the order line Id and error message for this line Id record.
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldErrorMessages DooLineId	DooLineId	String	Order line identifier in Order Management.
ReleaseHoldProcessResponse ReleaseHoldResponseParams HoldErrorMessages ErrorMessage	ErrorMessage	String	Error message for each order line.

Upstream Source Systems

Integrate Order Management with Source Systems

Set up your source system in Order Management so it can access reference data and master data that your source system uses.

Setting up the source system allows Order Management to identify where the sales order originates and helps to define the characteristics of the source system, such as whether its an order capture system or order fulfillment system, and whether the source system requires Order Management to do cross-referencing when a user creates a sales orders in Order Management. Order Management uses these details to establish cross-reference values for various entities.

If your deployment must integrate with a system that resides outside of Order Management, then you can register a connector that allows Order Management to communicate with it. You must create, deploy, and register the connector. This topic describes how to register the connector and connect Order Management to a source system, such as Oracle Configure, Price, and Quote. You add a connector that uses a web service that communicates order details with the source system.

Summary of the Set Up

1. Create a user credential key.
2. Set up the source system.
3. Administer the source system.
4. Add the connector.
5. Add roles and privileges.
6. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Create Credential Key

You must create a user credential key to integrate Order Management with the service.

The interface uses open access protocols, such as HTTP, so extra security setup is required. You must make sure the user credential is valid in the source system you're integrating, and in the security certificate so the integration can encrypt and decrypt messages. For details, see *Securing SCM*.

Create a user credential key.

1. Use Oracle Wallet Manager to add a user credential key to a credential map.
   You must use the administration privilege and administrator role.
2. In Oracle Wallet Manager, in the Select Map list, select oracle.wsm.security.
3. Enter the user credential key, user name, and password from the service that you're integrating with Order Management.
4. Make sure you have the privileges that you need to administer Order Management.
5. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage External Interface Web Service Details
6. On the Manage Connector Details page, click **Actions > Add Row**, then set the values that you set in steps 1 through 3.

Set Up the Source System

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Customers
   - Task: Manage Trading Community Source Systems
2. On the Manage Trading Community Source Systems page, click **Actions > Create**.
3. On the Create Source System page, set the values.

Attribute	Description
Code	Enter any text that Order Management can use as an abbreviation for the system. Order Management uses this code to identify this system throughout the user interface, such as in lists and logs. For example, assume you work for a company named Vision Corporation, and that your deployment must integrate with a legacy order capture system named Vision Capture. You can enter `VCAP`. The Manage Trading Community Source Systems page comes predefined to use Order Orchestration and Planning (OPS) to orchestrate and plan your sales order. If you use the Order Management work area to create sales orders, then you must not change this behavior, but you can use this page to add the source system you use to import a source order from a channel system.
Name	Enter text that describes the source system, such as `Vision Capture`.

Attribute	Description
Type	Select a value. - **Spoke**. Identifies a spoke system, such as a legacy system. - **Purchased**. Identifies a purchased system, such as data from a third party provider.
Options	Specify the type of data that you will import. - **Enable for Items**. Required. Import data for items. - **Enable for Trading Community Members**. Required. Import data for the trading community. Establishes the Original System Unique Reference `(OSR)` for customer entities. - **Enable for Order Orchestration and Planning**. Required. Import data for Order Orchestration. - **Enable for Assets**. Optional. Import data for assets. For example, if you add a check mark to Enable for Trading Community Members, then you can select the source system as a data source on various pages in the Order Management work area and the Order Orchestration work area.

4. Click **Save and Close > Done**.

Administer the Source System

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Upstream and Fulfillment Source Systems
2. On the Manage Upstream and Fulfillment Source Systems page, click **Actions > Create**.
3. In the Create Source System dialog, set the values.

Attribute	Description
Code	Select the code that you created earlier, such as **VCAP**.
Time Zone	Select the time zone where the server is located.
Version	Select **Other**.
Order Orchestration Type	Select a value. - **Fulfillment**. Specify the source system as a fulfillment system where Order Management sends fulfillment requests and receives fulfillment replies. - **Order Capture**. Specify the source system as an order capture system that sends source orders to Order Management. You typically use Order Capture with the import web service.

Attribute	Description
Collections Allowed	Contains a check mark.
Enable Data Cross-Reference	If the source system. o Expects Order Management to do the cross-reference, then enable this option. o Uses the same values that Oracle Applications use, and you already set up these values in Oracle Applications, then don't enable this option.

4. Click **Save and Close > Done**.

Add the Connector

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage External Interface Web Service Details
2. On the Manage Connector Details page, click **Actions > Add Row**.
3. In the new row, set the values, then click **Save and Close**.

Attribute	Description
Target System	Select the code you created in the Create Source System dialog, such as **VCAP**.
Connector Name	Enter text that describes the connector. For example, enter `Connector_to_VCAP`
Connector URL	Enter the URL that locates the connector service that resides on the source system. In this example, enter the URL that locates the `VCAP system`.
User Name and Password	Enter the values that the Status Update service requires. For example, the user that you specify must be a valid user, and this user must use the privileges that allow this user to run the Status Update service. Order Management uses the credentials you provide so it can communicate with the order capture system when it creates the order, and when it provides a status update.

4. **Optional.** Allow more than one source system instance to communicate with Order Management.
   - Use Trading Community Architecture to add a value to the Target System list.
   - Repeat step 3, except set Target System to the value that you added in Trading Community Architecture.

     For example, assume you work for a telecommunications company. You add one connection to a system named PER_ORA_BM_CPQ for personal phone lines, then add another connection to a system

named BUS_ORA_BM_CPQ for business lines. CPQ is an acronym for Configure, Price and Quote. Order Management will deliver status notifications and billing notifications to any system that contains the string ORA_BM_CPQ.

You can add a prefix, a suffix, a prefix and a suffix, or no prefix or suffix to the string. For example, you can use ABC_ORA_BM_CPQ_XYZ.

5. Verify that Order Management is connected to the source system, and that its communicating sales order data.
    - Use a page in the Order Management work area to verify that it updated the order status. For example, verify that it updated the status from Scheduled to Shipped.
    - Sign into your source system, then verify that it displays the updated status of the sales order that you examined in Order Management. For example, if Order Management updated the status from Scheduled to Shipped on the fulfillment line, then verify that your source system also displays Shipped.

    If Order Management can't connect to your source system, then it might display an error message that indicates it can't connect. For details, see *Fix Connection Problems with Source Systems*.

Connecting to Configure, Price, and Quote

If you connect to Configure, Price, and Quote, then do these steps.

- Set the Target System for the connector to `ORA_BM_CPQ`.
- Use the Manage Business Event Trigger Points page to enable the Fulfillment Line Status Update trigger point.
- Make sure the connector URL references the BM-CPQ status update service. If it doesn't reference this service, then the Business Events Message page will display an error. The URL is different for each BM-CPQ instance. For example, `host:port//BM-CPQ-statusUpdateService`, where you replace **host:port** with your sever address.
- The setup you make for Fulfillment Line Status Update and the corresponding set up in the Edit Status Rule Set area of the Manage Orchestration Process Definitions page doesn't affect how Order Management communicates status values.
- Order Management sends only the following status values. You can't modify this behavior.
    - Scheduled
    - Shipped
    - Awaiting Billing
    - Billed
    - Canceled
    - Closed

Add Roles and Privileges

The user who calls the web service must use an application role with web service privilege Manage Order Orchestration Decomposition Web Service (DOO_MANAGE_ORDER_ORCHESTRATION_DECOMPOSITION_WEB_SERVICE_PRIV).

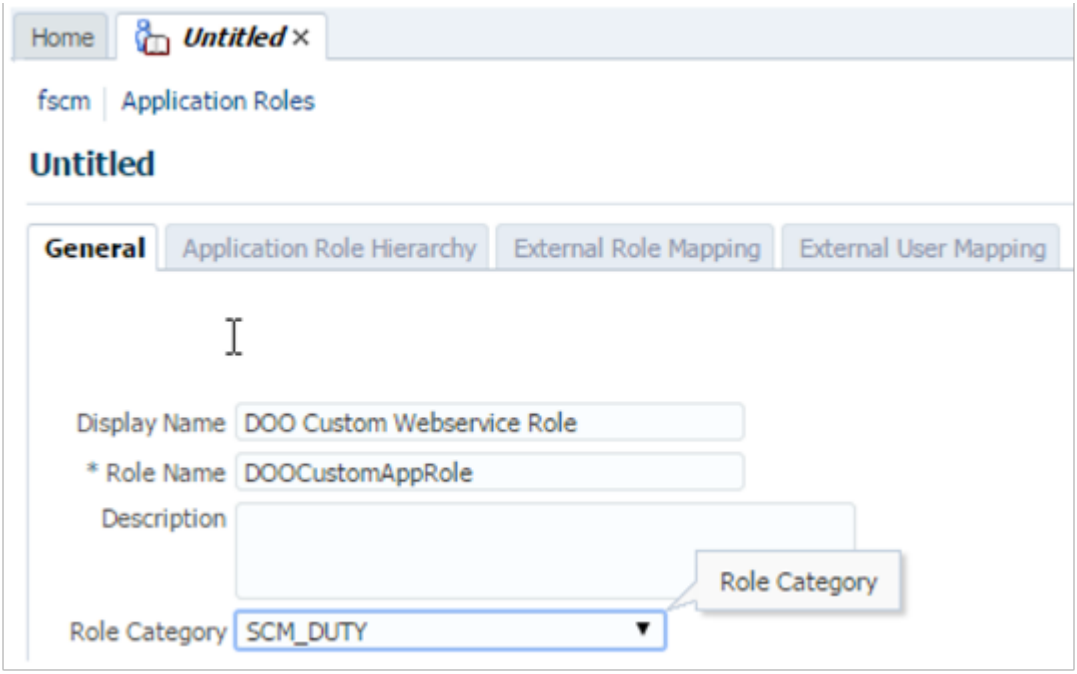
This role and privilege makes sure each service and response request from a source system works correctly when the source system isn't part of Oracle Applications, or when receiving a request from a fulfillment task that isn't in Oracle Applications.

Add roles and privileges.

1. Sign into the security console.
2. Click **Application Roles**.

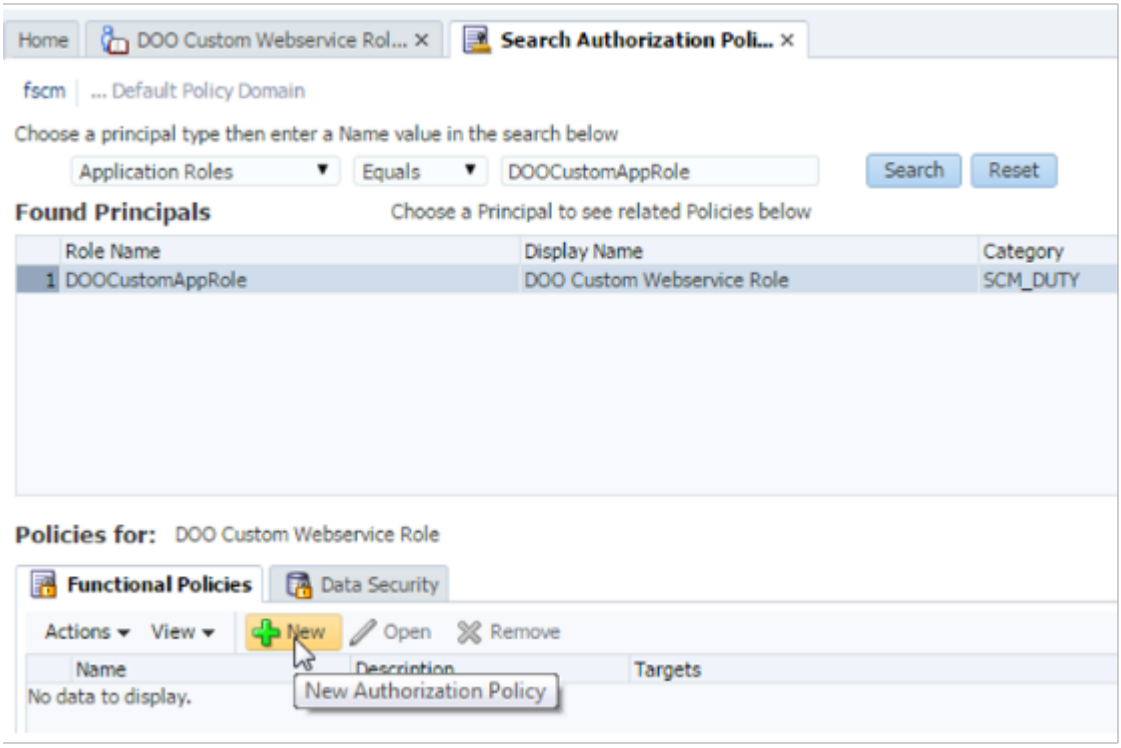
3. On the General tab, set the values, then save your changes.

Attribute	Value
Display Name	DOO Modified Role Service
Role Name	DOOModifiedAppRole
Role Category	SCM Abstract Roles



4. Navigate to the Search Authorization Policies tab, then search for DOOCustomAppRole.

5. In the Functional Policies tab, click **New**.



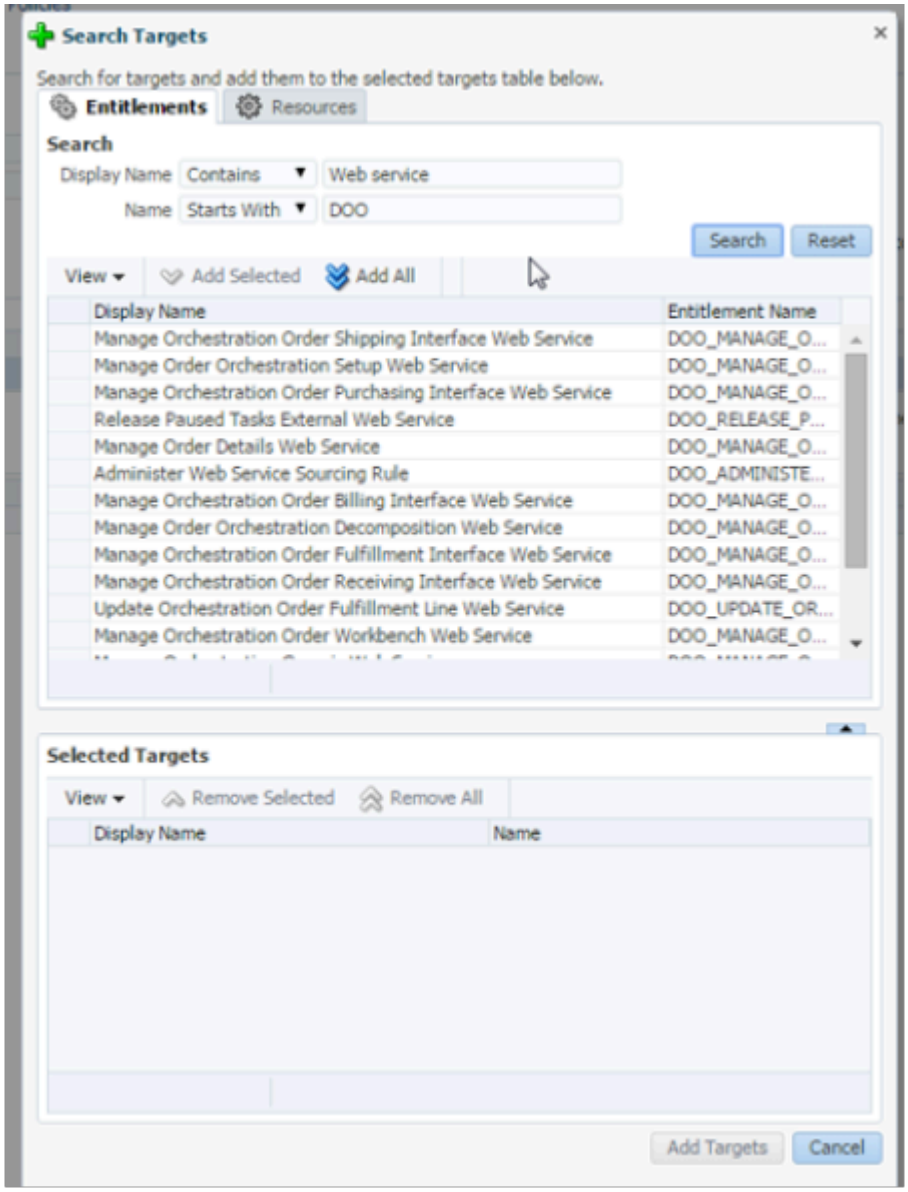
6. In the Untitled tab, set the value, then save your changes.

Attribute	Value
Name	DOOCustomRolePolicy

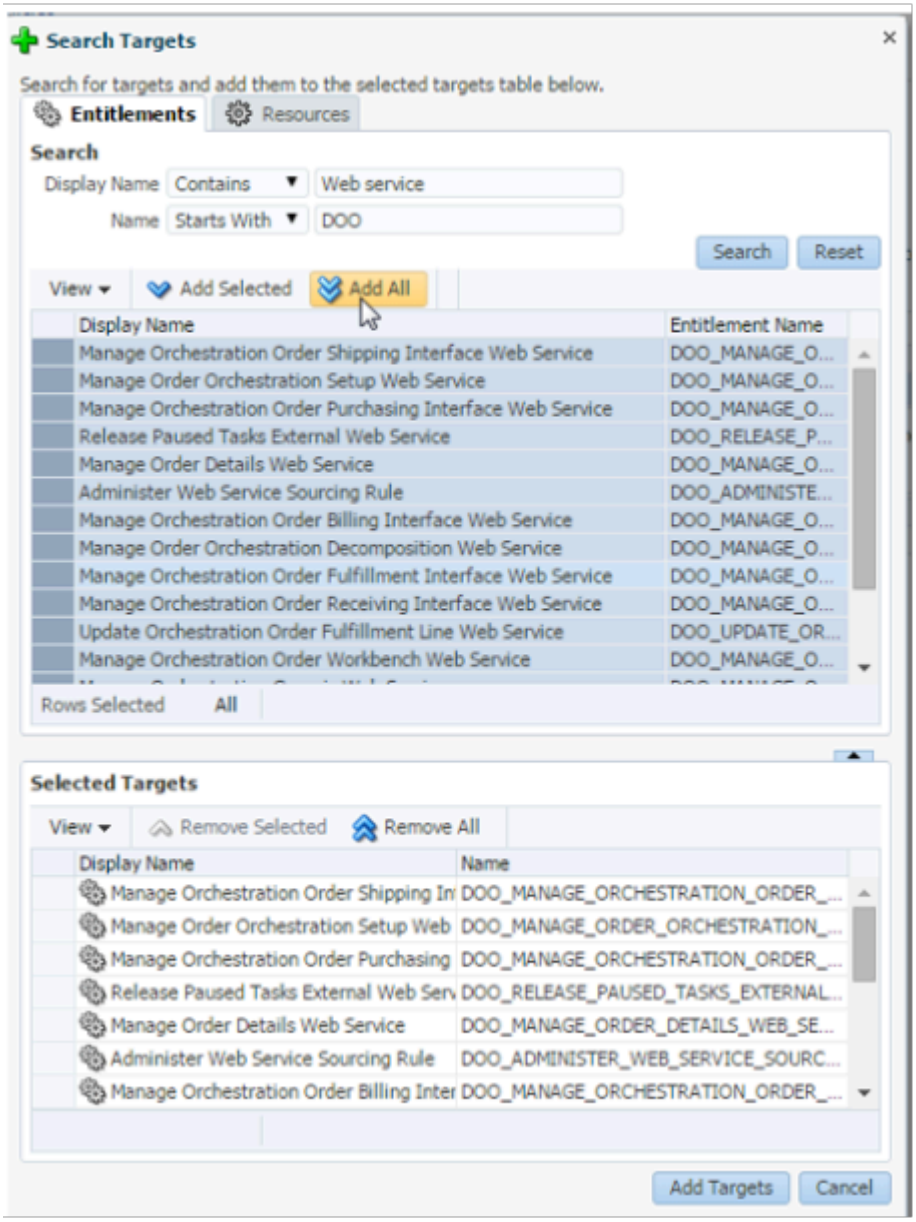
7. In the Targets area, click **Add Target**.
8. In the Search Targets dialog, set values, then click **Search**.

Attribute	Value
Display Name Contains	Web service
Name Starts With	DOO

Attribute	Value

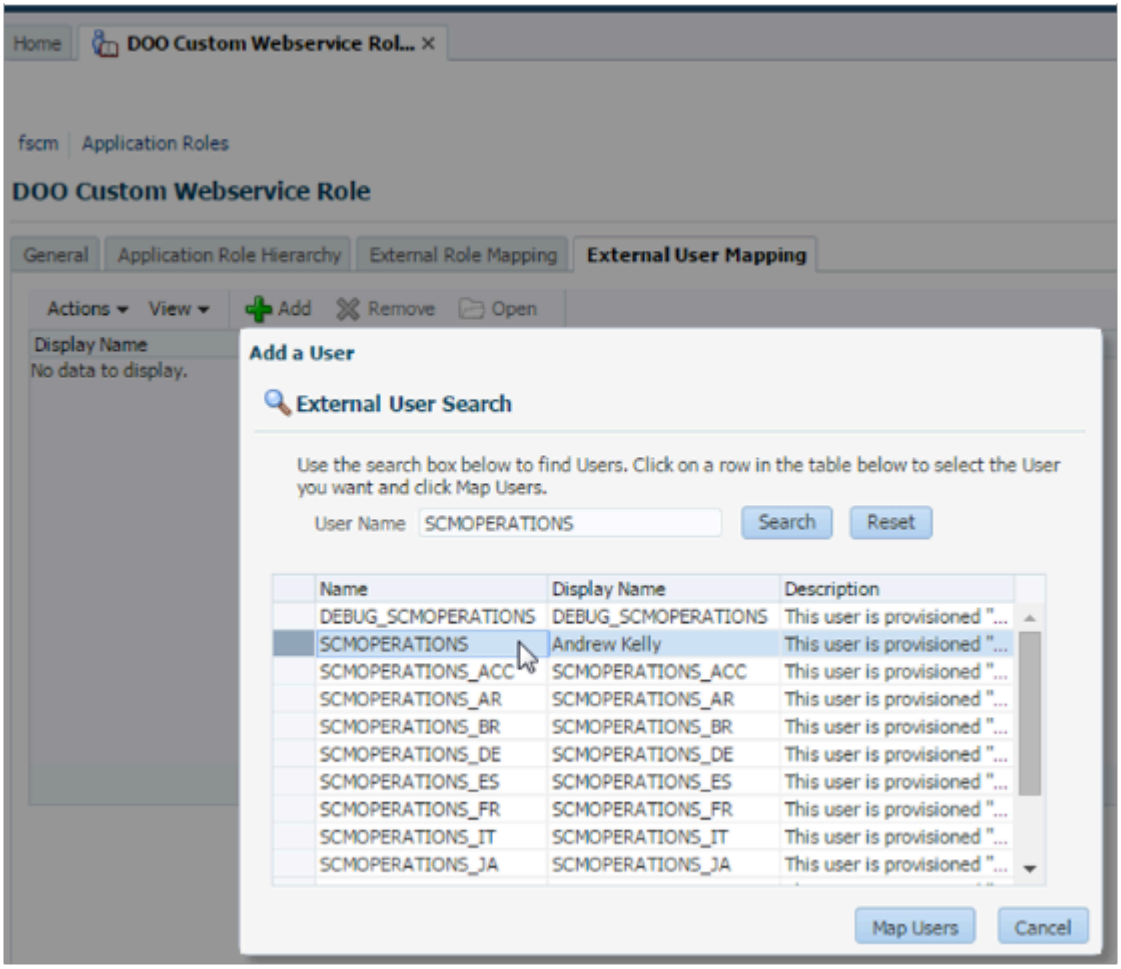


9. Add the privileges you need for each of the web services that you must grant to the user.



10. Navigate to the DOOCustomRolePolicy page.

11. In the `DOO Webservice Role` area, click **Add**.
12. In the Add a User dialog, add a user, then click **Map Users**.



Test Your Set Up

Create a test user in your test environment to make sure your deployment can authenticate the message that Order Management sends.

1. Sign into the Administration Console of the Oracle WebLogic Server.
2. In the Administration Console, click **Create Users**.
3. In the Create a New User area, set the values, then click **OK**.

Attribute	Value
Name	The value you enter must match the name you used when you created the user credential key earlier in this topic.
Description	User name and password to use when sending a message to the test client.
Provider	DefaultAuthentication

Attribute	Value
Password	The value you enter must match the name you used when you created the user credential key earlier in this topic.

4. Test the client in your source system to make sure it can send and receive messages to and from Order Management.

Related Topics

- Fix Connection Problems with Source Systems
- Integrate Order Management with Source Systems

Integrate Names and Codes Between Source Systems and Order Management

If your deployment includes an order capture system, then you must map names and codes from it into Order Management.

An orchestration reference object is an object that resides in the set of objects that an orchestration process processes so it can determine the meaning of a name or the description of a code, such as a payment term name, freight code, or transport code.

An order capture system typically sends sales order data that contains names or codes to an orchestration process, and the orchestration process must display a meaning for the name or a description for the code. You must collect the data that determines these meanings and descriptions into the Order Orchestration and Planning Data Repository.

Assume your order capture system sends sales order data that includes a payment term of 2/10, Net 30 to an orchestration process, and data in the repository includes a payment term of 2/10, Net 30. The orchestration process uses the matching codes to identify the payment term description.

```
2% discount earned if paid within 10 days
```

To get the complete list of orchestration reference objects, you can examine the collected data for them, and view the list of values for the Lookup Type field.

Match Import Data to Order Management Data

Make sure the data you import matches the structure and data type that Order Management uses.

Its important that the data you import or integrate for customer, ship-to, and bill-to attributes is compatible with data in the Order Management database. Use SQL to get data from the Order Management database, then modify your import data to make sure it matches Order Management database requirements for data type and structure.

Get Sold-To Customer

```
SELECT dha.ORDER_NUMBER ,
 dha.source_order_number,
 dha.SOLD_TO_PARTY_ID ,
 dha.STATUS_CODE ,
 hz.PARTY_ID ,
 hz.PARTY_NUMBER ,
 hz.PARTY_NAME
FROM fusion.doo_headers_all dha,
 fusion.HZ_PARTIES HZ
WHERE dha.SOURCE_ORDER_NUMBER = ('&SOURCE_ORDER_NUMBER')
 -- AND status_code <> 'DOO_REFERENCE'
 -- AND Submitted_Flag = 'Y' -- is this the active/submitted version
 and hz.PARTY_ID =dha.SOLD_TO_PARTY_ID
```

Get Ship-To Details on Order Header

```
SELECT SOURCE_ORDER_NUMBER,
 SOLD_TO_CUSTOMER_ID,
 SOLD_TO_PARTY_ID ,
 HZP.PARTy_name
 ||
 ' '
 ||
 HZP.PARTY_NUMBER "Sold to Customer",
 DOA.ADDRESS_USE_TYPE ,
 hza.account_number ,
 hzp_ship_to.party_name ,
 hza.account_name ,
 doa.PARTY_SITE_ID ,
 hzl.ADDRESS1 ,
 hzl.ADDRESS2 ,
 hzl.ADDRESS3 ,
 hzl.ADDRESS4 ,
 hzl.CITY ,
 hzl.POSTAL_CODE ,
 hzl.STATE ,
 hzl.COUNTRY
FROM FUSION.HZ_PARTIES HZP ,
 FUSION.HZ_PARTIES HZP_SHIP_TO ,
 FUSION.DOO_HEADERS_aLL DHA ,
 fusion.DOO_ORDER_ADDRESSES DOA ,
 fusion.HZ_CUST_ACCOUNTS HZA ,
 fusion.HZ_CUST_ACCT_SITES_ALL hzcasa,
 fusion.HZ_PARTY_SITES hzps ,
 fusion.hz_locations HZL
WHERE HZP.PARTY_ID = DHA.SOLD_TO_PARTY_ID
 AND dha.header_id = doa.header_id (+)
 AND
 (
 doa.ADDRESS_USE_TYPE = 'SHIP_TO'
 OR doa.ADDRESS_USE_TYPE IS NULL
 )
 AND doa.party_site_id = hzps.party_site_id (+)
 AND hzcasa.PARTY_SITE_ID (+) = hzps.PARTY_SITE_ID
 AND hzps.party_id = hzp_ship_to.party_id (+)
 AND HZcasa.CUST_ACCounT_ID = hza.CUST_ACCOUNT_ID (+)
 AND hzps.location_id = hzl.location_id (+)
 AND DHA.SOURCE_ORDER_NUMBER = ('&SOURCE_ORDER_NUMBER')
 AND DHA.status_code <> 'DOO_REFERENCE'
 AND DHA.Submitted_Flag = 'Y' -- is this the active/submitted version
```

Get Bill-To Details on Order Header

```
SELECT SOURCE_ORDER_NUMBER,
 SOLD_TO_CUSTOMER_ID,
 SOLD_TO_PARTY_ID ,
 HZP.PARTy_name
 ||
 ' '
 ||
 HZP.PARTY_NUMBER "Sold to Customer",
 DOA.ADDRESS_USE_TYPE ,
 hza.account_number ,
 hzp_ship_to.party_name ,
 hza.account_name ,
 doa.PARTY_SITE_ID ,
 hzl.ADDRESS1 ,
 hzl.ADDRESS2 ,
 hzl.ADDRESS3 ,
 hzl.ADDRESS4 ,
 hzl.CITY ,
 hzl.POSTAL_CODE ,
 hzl.STATE ,
 hzl.COUNTRY
FROM FUSION.HZ_PARTIES HZP ,
 FUSION.HZ_PARTIES HZP_SHIP_TO ,
 FUSION.DOO_HEADERS_aLL DHA ,
 fusion.DOO_ORDER_ADDRESSES DOA ,
 fusion.HZ_CUST_ACCOUNTS HZA ,
 fusion.HZ_CUST_ACCT_SITES_ALL hzcasa,
 fusion.HZ_PARTY_SITES hzps ,
 fusion.hz_locations HZL
WHERE HZP.PARTY_ID = DHA.SOLD_TO_PARTY_ID
 AND dha.header_id = doa.header_id (+)
 AND
 (
 doa.ADDRESS_USE_TYPE = 'SHIP_TO'
 OR doa.ADDRESS_USE_TYPE IS NULL
 )
 AND doa.party_site_id = hzps.party_site_id (+)
 AND hzcasa.PARTY_SITE_ID (+) = hzps.PARTY_SITE_ID
 AND hzps.party_id = hzp_ship_to.party_id (+)
 AND HZcasa.CUST_ACCounT_ID = hza.CUST_ACCOUNT_ID (+)
 AND hzps.location_id = hzl.location_id (+)
 AND DHA.SOURCE_ORDER_NUMBER = ('&SOURCE_ORDER_NUMBER')
 AND DHA.status_code <> 'DOO_REFERENCE'
 AND DHA.Submitted_Flag = 'Y' -- is this the active/submitted version
```

Get Ship-To and Bill-To Details on Order Line

```
SELECT dha.Source_order_number ,
 dha.order_number ,
 dha.submitted_Flag ,
 dfla.SHIP_TO_PARTY_ID ,
 dfla.SHIP_TO_PARTY_SITE_ID,
 dfla.BILL_TO_CUSTOMER_ID ,
 dfla.BILL_TO_SITE_USE_ID
FROM Fusion.DOO_headers_all dha,
 Fusion.DOO_fulfill_lines_all dfla
WHERE dha.header_id = dfla.header_id
 AND dha.source_order_number = '&ENTER SOURCE_ORDER_NUMBER'
```

Related Topics

- Use SQL to Query Order Management Data
- Overview of Importing Orders Into Order Management
- How Order-to-Cash Works with Order Capture Systems
- Roadmap for Setting Up Order-to-Cash

Temporarily Reserve Supply with Order Management

You can use the Temporarily Reserve Supply while Shopping or Quoting feature when you have a lot of sales orders that need the same supply at about the same time. Use it when you place orders through your eCommerce application or web store, and then fulfill them in Oracle Order Management.

If you don't use this feature, then you can confirm supply for these orders only one time, and only when Order Management schedules the order. If you have lots of orders that need the same supply at about the same time, then Order Management can't guarantee that there's enough supply to fulfill all of them.

Example

Assume you're an administrator for Computer Service and Rentals and the AS54888 Desktop Computer is your top selling item.

- Your customer Yu Li browses your web store for the AS54888, finds that there's a quantity of 500 of the AS54888 currently available, adds a quantity of 10 to source order 12345 in the shopping cart, pays for the items, then submits the order.
- The AS54888 is your top selling item, and you are currently processing 50 other orders that also include the AS54888.
- You can't reserve Yu's quantity of 10, so Order Management might or might not be able to fulfill it given all the competing demand from the other 50 orders.

Use this feature and Global Order Promising will temporarily reserve Yu's quantity of 10 until Order Management schedules the order line. Promising uses the DemandSourceLineReference attribute to reserve supply from order 12345 on your web store for the fulfillment line in Order Management. For details, see *Temporarily Reserve Supply While Shopping or Quoting*.

Change Orders

Promising and Order Management use the DemandSourceLineReference only one time to temporarily reserve supply, and use it only up until Promising successfully schedules the order line in Order Management.

- If Promising has successfully scheduled the line, then Order Management treats any change that you make on the line in Order Management or in your source system the same way it treats a change when you don't use this feature.
- If you revise the line in your source system, and if you send a new value in DemandSourceLineReference, then Order Management will ignore the new DemandSourceLineReference.

Let's say you have a promotion where you give a quantity of 1 free AS54888 for each quantity of 10 AS54888s that Yu orders.

- You reserve a quantity of 10 for Yu in the web store, but you send a quantity of 11 to Order Management.
- Or you bring the order into Order Management in draft status, change the quantity from 10 to 11 in the Order Management work area, then submit it.

Promising will use the temporary supply that it reserved for a quantity of 10, will promise the additional quantity of 1 as new demand, and will schedule that new demand according to your Promising rules just like it would any other line that you add in the Order Management work area. For details, see *Assignments and Promising Rules*.

Configured Items

You can use this feature with a standard item, a configured item, or a kit. You can specify a unique value in DemandSourceLineReference for each line that contains the model, option class, or option.

Consider an example.

1. Assume you add a pick-to-order item or a kit to your shopping cart and the item contains included items.

Item	Order Line in Your Web Store	Quantity
Kit Parent	101	2
Included Item1	102	2
Included Item2	103	2
Included Item3	104	2

   You must send a request to reserve supply, and your request must include the FulfillmentLineIdentifier for the parent and your included items. Don't specify DemandSourceLineReference in your import payload. Instead, use the autoSchedule REST API to send a concatenation of the order line and the name of the included item from your web store, such as 101-Included item1. For example:

Oracle Item Number	Order Line in Web Store	Quantity	FulfillmentLineIdentifier in autoSchedule	DemandSourceLineReference
Kit parent	101	2	101	101
Included item1	102	2	101-Included item1	Leave empty.
Included item2	103	2	101-Included item2	Leave empty.
Included item3	104	2	101-Included item3	Leave empty.

2. Submit the order in your shopping cart, then send a request to Order Management to create the sales order. Include the DemandSourceLineReference for the parent, option class, or option, but not for the included items. Order Management will set the DemandSourceLineReference for each included item according to the DemandSourceLineReference value in your request.

For more:

- Go to *REST API for Oracle Supply Chain Management Cloud*, then expand Order Management > Global Order Promising > Automatically Schedule.
- *Overview of Configure-to-Order*

Guidelines

- You can use the Demand Source Line Reference attribute in the Order Management work area only to read a value. You can't use it to set a value.
- You must set up your item number in the Product Information Management work area, and you must reference that number in each interaction with your source system.
- Your eCommerce application or web store isn't part of Oracle Applications. It's a third-party order capture system that you use as a source system for orders that you fulfill in Oracle Order Management.

You can use the DemandSourceLineReference attribute only when you create the order through the Sales Orders for Order Hub Requests REST API, or when you route your file-based data import through REST API. For more:

- Go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management > Sales Orders for Order Hub Requests.
- *Use FBDI and REST API to Import Large Volumes of Sales Orders*

Downstream Fulfillment Systems

Connect and Route

Overview of Connecting Order Management to Your Fulfillment System

Set up a connector so Order Management can communicate with your fulfillment system.

Here's a summary of the set up.

1. Specify how Order Management sends the request to the outbound connector.
    - You specify the URL that locates the outbound connector when you set up the interface.
    - You deploy the outbound connector on a third-party application server.
    - Your information technology group must set up the WSDL for the outbound endpoint.
2. Specify the business conditions that will route the request to the connector.
3. The outbound connector transforms the message into a format that your fulfillment system can understand, then sends the message to your fulfillment system. You can use an integrated development environment, such as Oracle JDeveloper, to specify how to do transformation.
4. The fulfillment system sends a response to the inbound connector.

Response	Description
Immediate	The fulfillment system immediately sends the response.

Response	Description
	A long running task is a task that includes a wait step in the orchestration process. An immediate response doesn't wait to process a fulfillment request that involves a long running task. Instead, it only acknowledges receipt and replies with details that are immediately available. For example, assume the message requests a Boolean value, such as whether the customer is credit worthy or not credit worthy, and the fulfillment system already determined this value. The fulfillment system can respond immediately with the requested details.
Delayed	The fulfillment system sends the response only after a delay so it can finish the long running task. For example, assume the message requests a status update for the Received Date attribute, which is the date that shipping delivered the item to the customer. Shipping is normally a long running task that includes a pause step because shipping a physical item typically requires one or more days to finish. The fulfillment system can't respond immediately with the requested details. It must delay its response while it waits for the shipping task to finish.

Note

- The fulfillment system might also communicate more than one update over time. To process a delayed response, you can add another entry point service in the connector, or you can set up another connector. It isn't necessary to use the interface to set up the connector that receives the message from your fulfillment system and that sends a delayed response.
- This topic assumes you use the same connector to send the delayed response to Order Management.
- For examples of immediate and delayed responses, see *Convert Shipment Costs to Freight Charges*.

5. To send the response, the fulfillment system calls the inbound interface that you set up in Order Management. Order Management uses a single service to accept the response for each task type. You specify the URL that locates the WSDL for the service when you set up the connector.
6. The connector transforms the response into a message that Order Management can understand, then sends it to Order Management.

Here are the task types that the connector uses in Order Management to communicate with your fulfillment system.

Task Type	Description
Reservation	Reserve an item in inventory.
Shipping	Ship the item after you reserve it.
Invoice	Invoice the item after it ships.

For details about the flow you integrate, see *How Data Flows Through Order Management*.

Road Map to Integrating Order Management with Fulfillment Systems

Do the set ups described in *Implementing Order Management*. The steps in bold font are particularly important regarding achieving connectivity.

Step	Details	Description
1	*Create Your Own Task Type*	Specify how to process the sales order and order lines.
2	*Order Management Statuses*	Do. • Set up fulfillment line status. • Specify how to split fulfillment line status. • Update or close fulfillment lines that remain open.
3	*Set Up Orchestration Processes*	Set up your orchestration process to determine status conditions, status values, and fulfillment line status.
4	*Select Fulfillment Lines for Orchestration Process Steps*	Create a business rule that selects fulfillment lines, then specify whether to process them.
5	*Connect Order Management to Your Fulfillment System*	Set up a connector that enables Order Management to communicate with your fulfillment system.
6	*Manage Connector Details Between Order Management and Your Fulfillment System*	Specify the web service that enables Order Management to communicate with your fulfillment system.
7	*Route Requests from Order Management to Fulfillment Systems*	Specify a rule that selects your fulfillment system connector according to sales order, fulfillment line, or orchestration process attribute.
8	*Overview of Setting Up Extensible Flexfields in Order Management*	Optional. If you implement an extensible flexfield, then you must set it up, deploy it, and publish it.

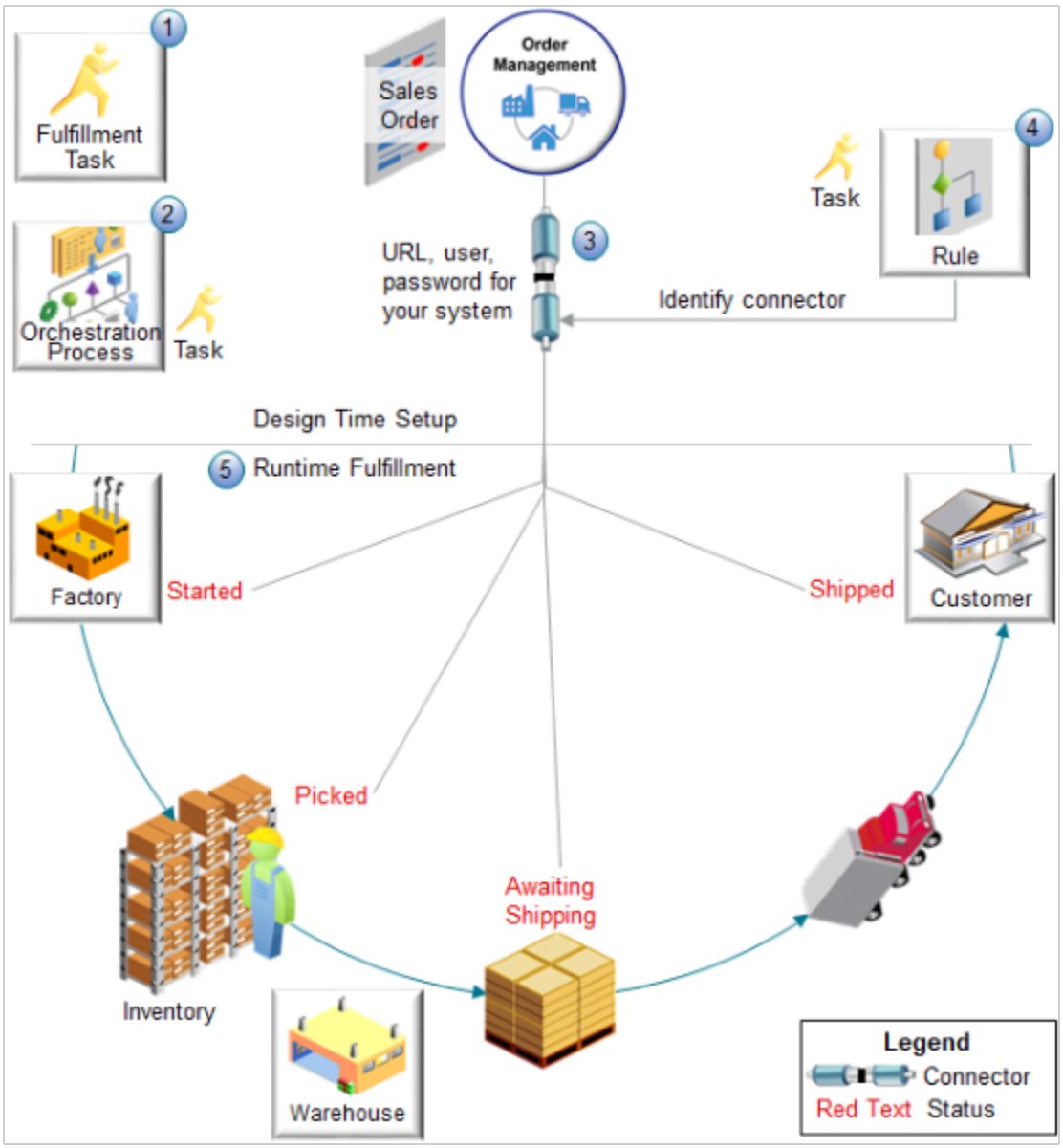
Related Topics

- How Data Flows Through Order Management
- Fulfillment Tasks

Example of Integrating Order Management with Your Fulfillment System

Assume you need to integrate Order Management with your shipment system. Use this topic to learn how.

An external fulfillment system is a fulfillment system that resides outside of Oracle Applications. Here's an example of how you set up the integration.



Note

1. You will specify status values and status conditions on your fulfillment task according to the values that your fulfillment system uses. You also specify the web services to use when communicating status.
2. Set up shipping steps on your orchestration process so they call your fulfillment system through the fulfillment task and task type that you set up for your integration. Modify status values and conditions on the orchestration process so they meet your integration requirements.
3. Set up a connector that includes the URL to your fulfillment system and any sign in requirements, such as user and password. Order Management uses the connector to find your fulfillment system and communicate with it.
4. Set up a routing rule that routes calls from your fulfillment system to the connector according to the task type that you specify.
5. At run time, Order Management and your fulfillment system use the connector to communicate status updates throughout the fulfillment lifecycle, such as Started, Picked, Awaiting Shipping, and Shipped.

Summary of the Steps

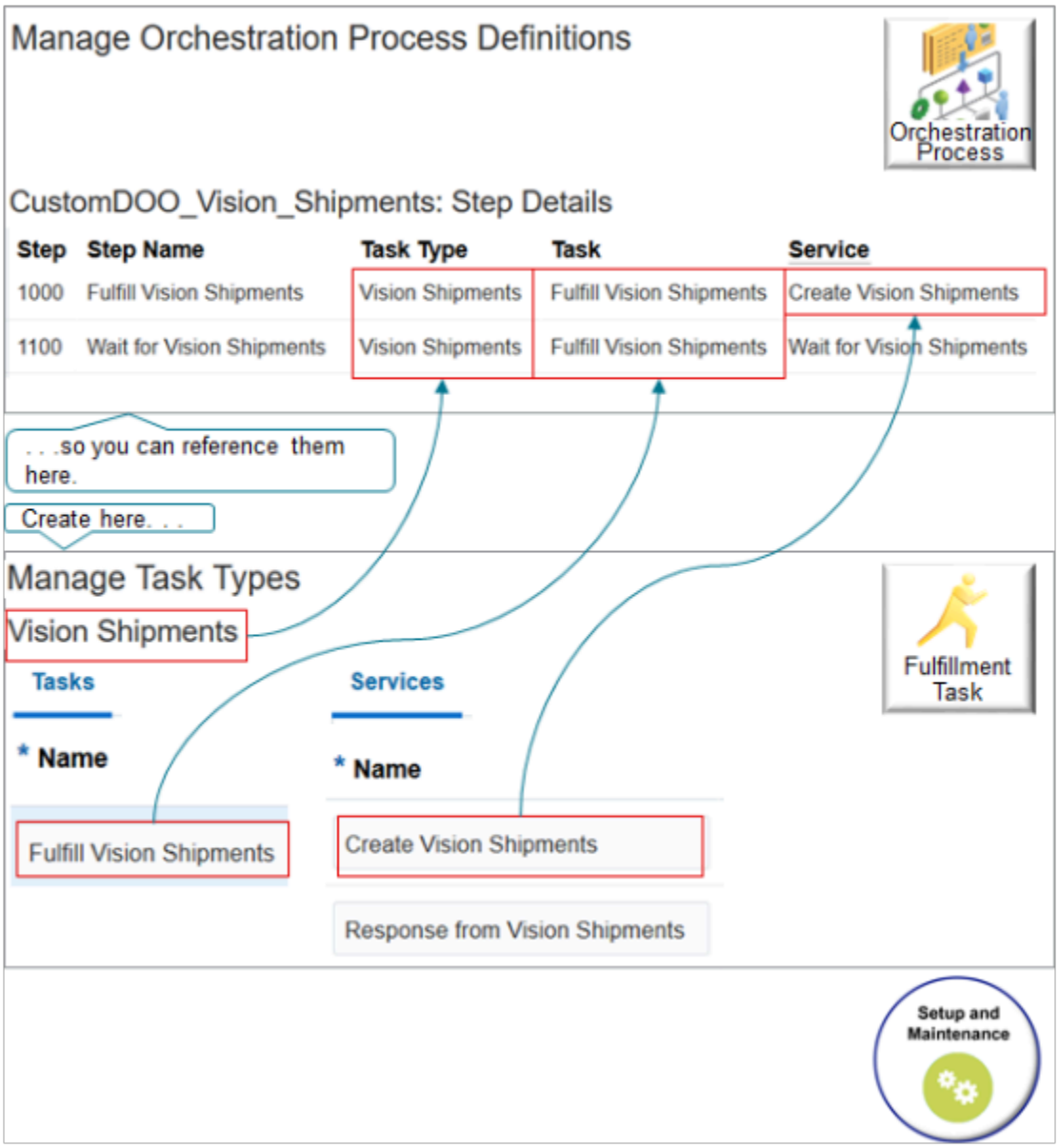
Assume you must integrate your shipping system, Vision Shipments, with Order Management so you can fulfill sales orders that you create in the Order Management work area. Vision Shipments uses status values that are unique to Vision Shipments, such as Awaiting Fulfillment and Awaiting Fulfillment Line Aggregation. You specify them in the Setup and Maintenance work area. Here's a summary of the steps you will do in this topic.

1. Manage task types.
2. Manage status values and status conditions.
3. Create a new orchestration process.
4. Add status values for the fulfillment line.
5. Publish your orchestration process.
6. Set up connector and routing rule.
7. Test your setup.

Note: This example describes one way to integrate Order Management with your shipment system. It is for illustration purposes only. You will need to use different values and procedures, depending on your implementation environment and business requirements.

Manage Task Types

To start, you create task type, task, and services so you can reference them later in this procedure when you set up the orchestration process.



Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Task Types
2. On the Manage Task Types page, click **Actions > Add Custom**, set values, then click **Save**.

Attribute	Value
Task Type	Vision Shipments

Attribute	Value
	You can use any value.
Description	Task type we use to communicate with our Vision Shipments fulfillment system.

3. In the Details area, on the Services tab, add the services you will use to communicate with your Vision Shipments fulfillment system.

Code	Name	Operation Code	Hold Enabled
Create Vision Shipments	Create Vision Shipments	Create	Yes
Vision Shipments Inbound	Response from Vision Shipments	Inbound	-
Cancel Vision Shipments	Cancel Vision Shipments	Cancel	No
Hold Vision Shipments	Hold Vision Shipments	Apply Hold	-
Release Vision Shipments	Release Vision Shipments	Release Hold	-
Update Vision Shipments	Update Vision Shipments	Update	Yes
Wait for Vision Shipments	Wait for Vision Shipments	Wait	-

Its recommend that you use these operations to accommodate the various statuses that Order Management uses when it processes each sales order. This way, the statuses in your fulfillment system will be synchronized with the statues in Order Management.

4. Click **Tasks**, click **Actions > Add Row**, set the values, then click **Save and Close**.

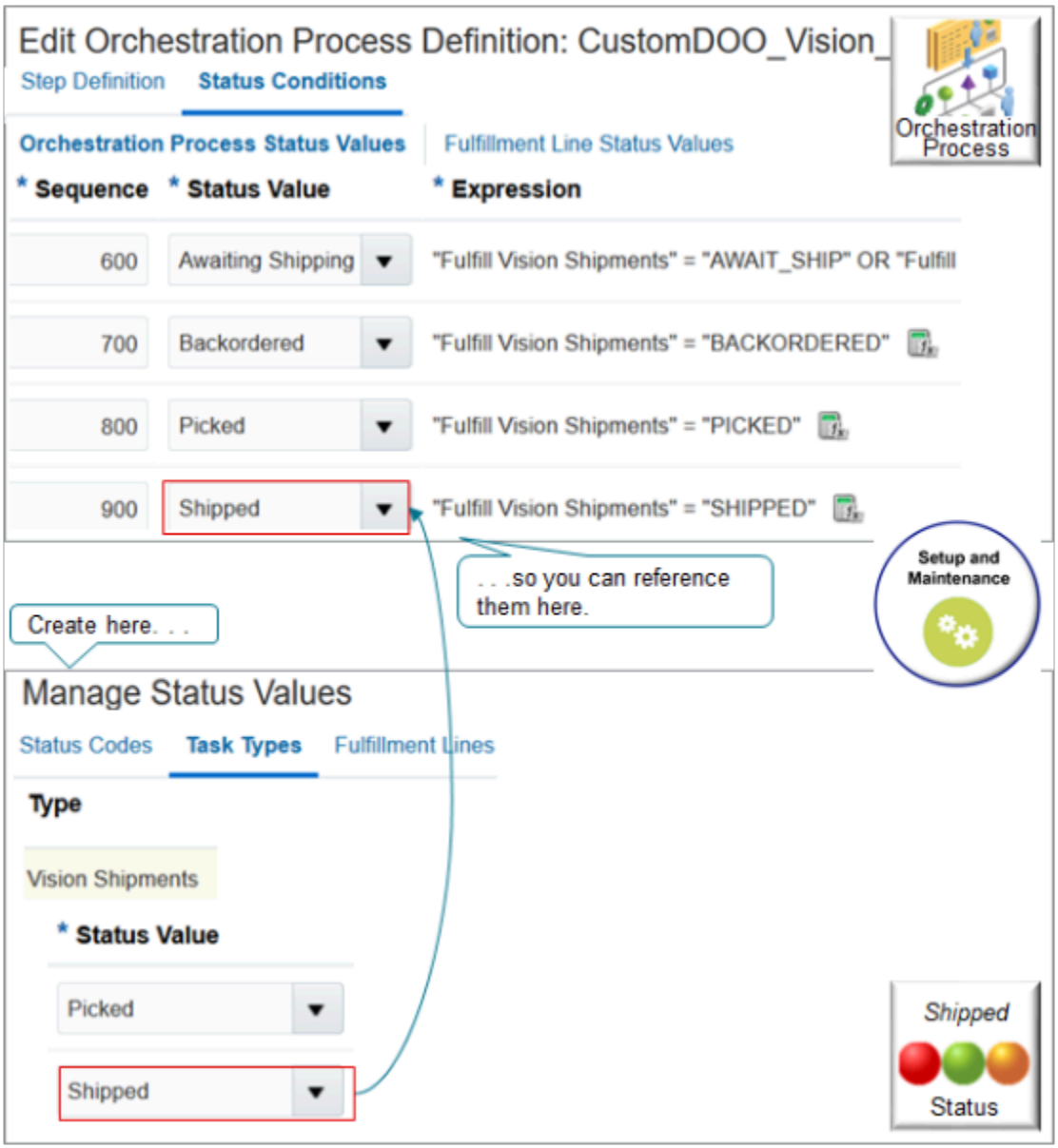
Code	Name	Display Name	Intermediate Replanning
Fulfill_Vision_Shipments	Fulfill Vision Shipments	Fulfill Vision Shipments	Doesn't contain a check mark.

Order Management uses the task to communicate status values with your fulfillment system.

Manage Status Values and Status Conditions

Manage Status Values

Create status values on the Manage Status Values page so you can reference them when you set up the orchestration process.



Note

- Add status values on the Manage Status Values Page.
- Later in this procedure, reference your status value on each orchestration process step.
- For details, see *Fulfillment Tasks*.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Status Values
2. On the Manage Status Values page, click **Task Types**, click **Query by Example**, enter `Vision Shipments` in the Type column, then press the Enter key.
3. In the Status Values area, notice that the Setup and Maintenance work area automatically created a set of status values for you when you created the Vision Shipments task type.

Status Value	Split Priority
Change Pending	10
Not Started	20
Started	30
Various	39
Completed with Various Statuses	50
User Action Required	60
Awaiting Response	110
Not Applicable	900
Cancellation Pending	970
Canceled	990

These values match what Order Management uses. You can add, delete, or modify them to accommodate your fulfillment system, as necessary. In general, we recommend that you keep these statuses so your fulfillment system is synchronized with statuses in Order Management, but you can add more.

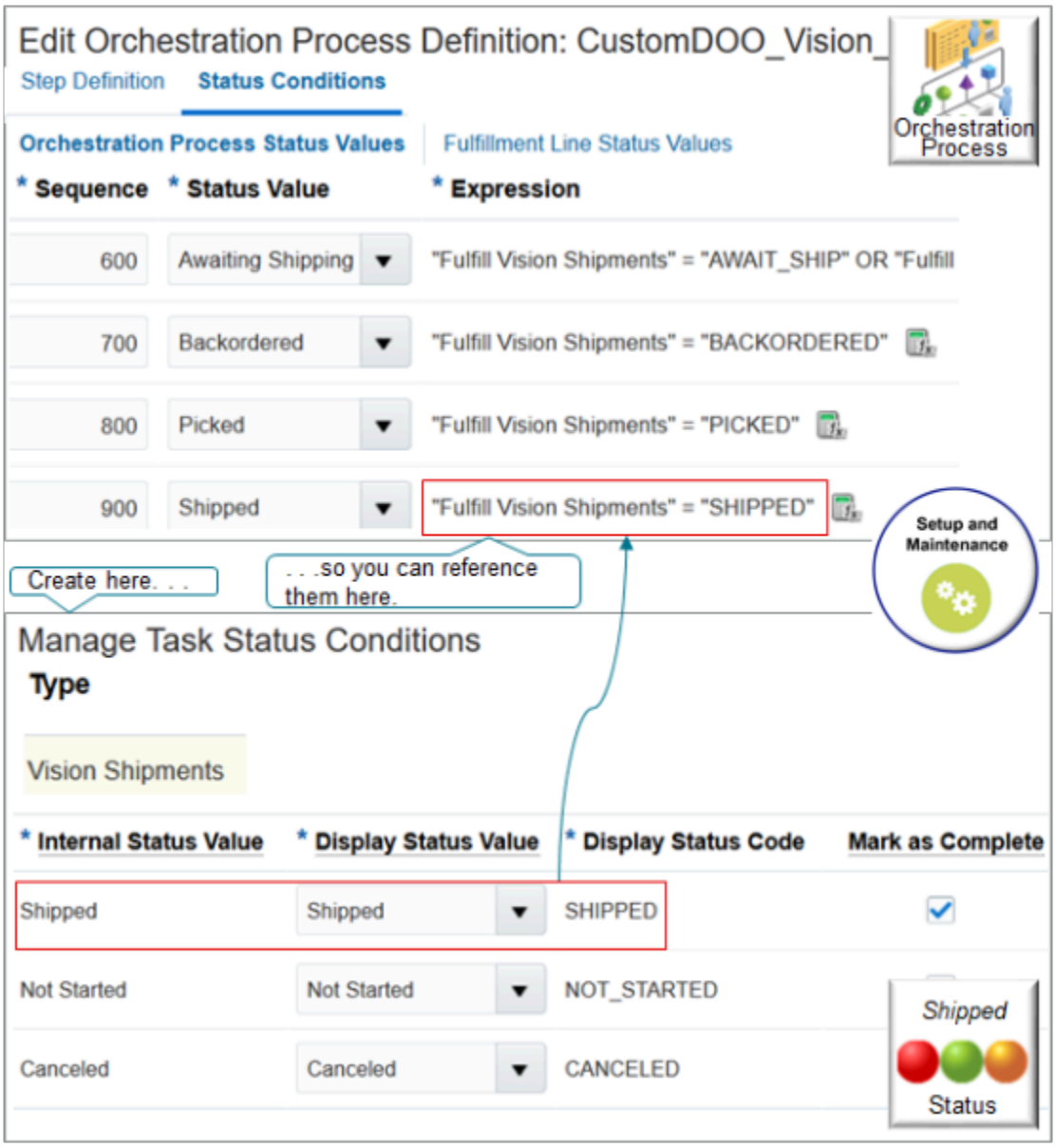
For example, here's the same set of statuses but with a few new ones that Vision Shipments uses. The bold statuses are new. This way, Vision Shipments can communicate its statuses, such as Awaiting Fulfillment or Awaiting Fulfillment Line Aggregation, to Order Management and Order Management can display them in the Order Management work area, or you can reference them in your other set ups, as necessary.

Status Value	Split Priority
Awaiting Fulfillment	2
Awaiting Fulfillment Line Aggregation	5
Change Pending	20
Not Started	30
Started	40
Various	60
Partially Picked	70
Partially Backordered	80
Picked	90
Backordered	100
Shipped	110
Canceled	120
Not Applicable	130
Cancellation Pending	140
Awaiting Shipping	150
Awaiting Response	160
Completed with Various Statuses	170

Status Value	Split Priority
User Action Required	180

Manage Status Conditions

Create status values on the Manage Task Status Conditions page so you can reference them when you set up the orchestration process.



1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders

- Task: Manage Task Status Conditions

2. On the Manage Task Status Conditions page, click **Query by Example**, enter `Vision Shipments` in the Type column, then press the Enter key.
3. In the Status Conditions area, notice that the Setup and Maintenance work area automatically created a set of conditions for you when you created the Vision Shipments task type.

Internal Status Value	Internal Status Code	Display Status Value	Display Status Code
Awaiting Response	AWAIT_RESPONSE	Awaiting Response	AWAIT_RESPONSE
Canceled	CANCELED	Canceled	CANCELED
Cancellation Pending	CANCEL_PENDING	Cancellation Pending	CANCEL_PENDING
Change Pending	CHANGE_PENDING	Change Pending	CHANGE_PENDING
Completed with Various Statuses	COMPLETED_VAR	Completed with Various Statuses	COMPLETED_VAR
Not Applicable	NOT_APPLICABLE	Not Applicable	NOT_APPLICABLE
Not Started	NOT_STARTED	Not Started	NOT_STARTED
Started	STARTED	Started	STARTED
User Action Required	USER_ACTION	User Action Required	USER_ACTION
Various	VARIOUS	Various	VARIOUS

As with the status values, these status conditions match what Order Management uses. You can add, delete, or modify them to accommodate your fulfillment system, as necessary.

Here's the same set with some new conditions. The bold conditions are new. Make sure only the Shipped status contains a check mark in the Mark as Complete column.

Internal Status Value	Internal Status Code	Display Status Value	Display Status Code
Awaiting Fulfillment	AWAIT_FULFILLMENT	Awaiting Fulfillment	AWAIT_FULFILLMENT

Internal Status Value	Internal Status Code	Display Status Value	Display Status Code
Awaiting Fulfillment Line Aggregation	AWAIT_FLINE_AGGREGATE	Awaiting Fulfillment Line Aggregation	AWAIT_FLINE_AGGREGATE
Awaiting Response	AWAIT_RESPONSE	Awaiting Response	AWAIT_RESPONSE
Awaiting Shipping	AWAIT_SHIPPED	Awaiting Shipping	AWAIT_SHIPPED
Backordered	BACKORDERED	Backordered	BACKORDERED
Canceled	CANCELED	Canceled	CANCELED
Cancellation Pending	CANCEL_PENDING	Cancellation Pending	CANCEL_PENDING
Change Pending	CHANGE_PENDING	Change Pending	CHANGE_PENDING
Completed with Various Statuses	COMPLETED_VAR	Completed with Various Statuses	COMPLETED_VAR
Not Applicable	NOT_APPLICABLE	Not Applicable	NOT_APPLICABLE
Not Started	NOT_STARTED	Not Started	NOT_STARTED
Partially Picked	PARTIAL_PICK	Partially Picked	PARTIAL_PICK
Picked	PICKED	Picked	PICKED
Shipped	SHIPPED	Shipped	SHIPPED
Started	STARTED	Started	STARTED
User Action Required	USER_ACTION	User Action Required	USER_ACTION
Various	VARIOUS	Various	VARIOUS

Create a New Orchestration Process

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions

    For details, see *Guidelines for Setting Up Orchestration Processes*.
2. Create your process.
    - On the Manage Orchestration Process Definitions page, do a search.

Attribute	Value
Process Name	DOO_OrderFulfillmentGenericProcess

    - Click **Actions > Duplicate**, modify the attributes, then click **Save**.

Attribute	Value
Process Name	CustomDOO_Vision_Shipments
Process Display Name	Vision Shipments
Description	Custom orchestration process that fulfills sales orders in the Vision Shipments fulfillment system.

3. Modify steps.
    - In the Process Details area, modify values in the row that contains Create Shipment Request in the Step Name attribute.

Attribute	Value
Step Name	Fulfill Vision Shipments
Task Type	Vision Shipments This is the task type you created earlier in this procedure.
Task	Fulfill Vision Shipments

Attribute	Value
	This is the task you created earlier in this procedure.
Service	Create Vision Shipments This is the service you created earlier in this procedure.
Update Service	Update Vision Shipments
Cancel Service	Cancel Vision Shipments

Don't modify any of the other attributes.

- Modify values in the row that contains Wait for Shipment Advice in the Step Name attribute, then click **Save**.

Attribute	Value
Step Name	Wait for Vision Shipments
Task Type	Vision Shipments
Task	Fulfill Vision Shipments
Service	Wait for Vision Shipments
Exit Criteria	Add a check mark to Canceled. Add a check mark to Shipped.
Fulfillment Completion Step	Contains a check mark.
Next Expected Task Status	Shipped

Don't modify any of the other attributes.

Add Status Values for the Orchestration Process

1. Click **Status Conditions**, then, on the Orchestration Process Status Values tab, verify values in the Expression column.

Sequence	Status Value	Expression
600	Awaiting Shipping	"Fulfill Vision Shipments" = "AWAIT_SHIP" OR "Fulfill Vision Shipments" = "VARIOUS"
700	Backordered	"Fulfill Vision Shipments" = "BACKORDERED"
800	Picked	"Fulfill Vision Shipments" = "PICKED"
900	Shipped	"Fulfill Vision Shipments" = "SHIPPED"
1200	Partially Picked	"Fulfill Vision Shipments" = "PARTIAL_PICK"
1300	Partially Backordered	"Fulfill Vision Shipments" = "PARTIAL_BACK"

   Make sure you verify the correct rows. For example, several rows contain the Awaiting Shipping status value, but you must verify only row 600.

   If you don't have these values, then modify the expression, as necessary. For example, if you must modify row 600.

   - Click the **Calculator** icon in the Expression column.
   - In the Expression Builder dialog, delete everything in the Expression window, expand **CustomDOO_Vision_Shipments, Fulfill Vision Shipments**, click **AWAIT_SHIP [Awaiting Shipping]**, then click **Insert Into Expression**.
   - In the Expression window, enter the text OR after the closing bracket.
   - Click **VARIOUS [Various]**, then click **Insert Into Expression**.
   - Verify that the Expression window contains "AWAIT_SHIP" OR "VARIOUS", then click OK.

2. Add a new row.

   - Click **Actions > Add Row**, then set the values.

Attribute	Value
Sequence	1600

Attribute	Value
Status Value	Awaiting Fulfillment

- Click the **Calculator** icon in the Expression column.
- In the Expression Builder dialog, in the Expression window, enter the name of your task, enclose it with double quotation marks, then add a space and equal sign.

  Recall that in this example you created a fulfillment task named Fulfill Vision Shipments, so enter `"Fulfill Vision Shipments" =`.
- Expand **CustomDOO_Vision_Shipments, Fulfill Vision Shipments**, click **AWAIT_FULFILLMENT [Awaiting Fulfillment]**, then click **Insert Into Expression**.
- Verify that the Expression window contains `"Fulfill Vision Shipments" = "AWAIT_FULFILLMENT"`, then click **OK**.

3. Add a new row.
   - Click **Actions > Add Row**, then set the values.

Attribute	Value
Sequence	1700
Status Value	Awaiting Fulfillment Line Aggregation

   - In the Expression column, use the Calculator icon to add `"Fulfill Vision Shipments " = "AWAIT_FLINE_AGGREGATE".`
4. Click **Save**.

Add Status Values for the Fulfillment Line

1. Click **Fulfillment Line Status Values**, then click **Edit Status Rule Set**.
2. On the Edit Status Rule Set page, verify values.

Sequence	Status Value	Expression
1200	Awaiting Shipping	"Fulfill Vision Shipments" = "AWAIT_SHIP" OR "Fulfill Vision Shipments" = "VARIOUS"
1300	Backordered	"Fulfill Vision Shipments" = "BACKORDERED"
1400	Picked	"Fulfill Vision Shipments" = "PICKED"
1500	Shipped	"Fulfill Vision Shipments" = "SHIPPED"

Sequence	Status Value	Expression
1800	Partially Picked	"Fulfill Vision Shipments" = "PARTIAL_PICK"
1900	Partially Backordered	"Fulfill Vision Shipments" = "PARTIAL_BACK"

3. Click **Actions > Add Row**, then set the values.

Sequence	Status Value	Expression
1930	Awaiting Fulfillment	"Fulfill Vision Shipments" = "AWAIT_FULFILLMENT"

4. Click **Actions > Add Row**, set the values, then click **Save and Close**.

Sequence	Status Value	Expression
1960	Awaiting Fulfillment Line Aggregation	"Fulfill Vision Shipments" = "AWAIT_FLINE_AGGREGATE"

Publish Your Orchestration Process

1. On the Edit Orchestration Process Definition page, click **Actions > Validate**.
2. Click **Actions > Release**.
3. Wait for the release to finish, then cancel the Download dialog.
4. Click **Actions > Deploy Process**.

Set Up Connector

1. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage External Interface Web Service Details

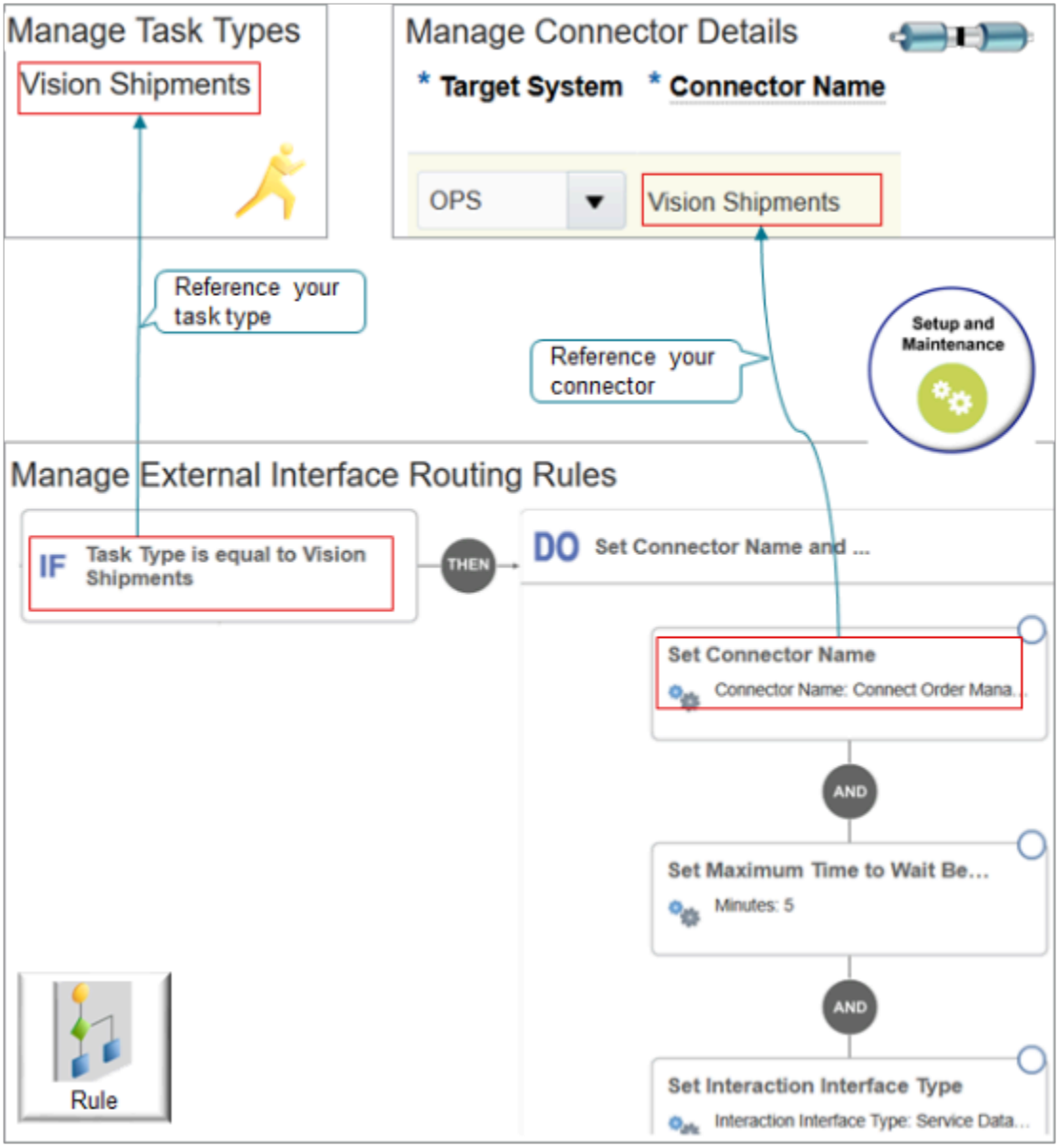
   For details about how to set up a connector, see *Overview of Connecting Order Management to Your Fulfillment System*.
2. On the Manage Connector Details page, click **Actions > Add Row**, set the values, then click **Save and Close**.

Attribute	Value
Target System	OPS

Attribute	Value
	OPS means Order Orchestration and Planning.
Connector Name	Vision Shipments
Connector URL	Enter the URL that locates your fulfillment system.
Connector Description	Connect Order Management to the Vision Shipments fulfillment system.
User Name	Name of a user you have set up on your fulfillment system.
Password	Password you need to access your fulfillment system.
Invocation Mode	Synchronous Service

Set Up Routing Rule

Create a routing rule that routes the fulfillment request to your connector according to task type.



Note

- Reference the task type that you created on the Manage Task Types page when you create the If statement.
- Reference the connector that you created on the Manage Connector Details page when you set the connector name in the Do statement.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Integration Routing Rules for Sales Orders

    For details about how to set up a routing rule, see *Use Visual Information Builder*.
2. On the Manage External Interface Routing Rules page, click **Create New Rule**.
3. Set the values, then click **Save and Close**.

Attribute	Value
Name	Route Sales Orders to the Vision Shipments Fulfillment System

4. Create the If statement.

```
If Task Type is equal to Vision Shipments
```

5. Create the Then statements.

```
Set Connector Name to Vision Shipments
Set Maximum Time to Wait Before Allowing Cancel to 5 Minutes
Set Interaction Interface Time to Service Data Object
```

Test Your Setup

1. Create a sales order in the Order Management work area, then click **Submit**.
2. Monitor the progress of the fulfillment line on the sale order. Verify that it proceeds through the fulfillment statuses as expected.

Related Topics

- Fulfillment Tasks
- Overview of Orchestration Processes
- Use Visual Information Builder
- Overview of Connecting Order Management to Your Fulfillment System

Connect Order Management to Your Fulfillment System

Set up a connector that allows Order Management to communicate with your fulfillment system.

This topic assumes these conditions are true.

- You use Oracle JDeveloper (Java Developer) to create the connector. You can use any similar development tool.
- You create a fulfillment connector that uses the Fulfillment task. You can use a similar approach for other tasks.
- You use a web service to communicate between Order Management and the connector. The connector can use some other technology that you select to communicate with your fulfillment system.

- You use basic security. Your implementation team must incorporate any other security that you need.
- The fulfillment system doesn't understand the Order Management message, so the connector must transform the message.
- The connector and the fulfillment system use security certificates that a certificate authority (CA) publishes.
- You have the privileges that you need to deploy the connector on the fulfillment system, or you licensed Oracle SOA (Service Oriented Architecture) Cloud Service so you can access Oracle JDeveloper on the cloud and integrate with other cloud applications and systems.
- Services in the fulfillment system all use the same user credentials.

Summary of the Set Up

1. Prepare to make connection.
2. Create connector.
3. Add branches for operations.
4. Set up delayed response.
5. Secure your connector and deploy your project.
6. Communicate extensible flexfields to fulfillment system.
7. Deploy connector.

This topic uses example values. You might need different values, depending on your business requirements.

Prepare to Make Connection

1. Get the administrative privileges you need to access the web server where you plan to deploy the connector.
2. Get source system details from Oracle Trading Community Architecture. You will use them to link connectors in the web service details to your fulfillment system.
3. Install the Java runtime environment on the computer you will use to create the connector.

   For details, see *Java Downloads for All Operating Systems*.
4. Open a web browser, then install Java SE Development Kit.

   For details, see *Java SE Development Kit 9 Downloads*.
5. Install Oracle SOA Suite.
   - Go to the *Oracle SOA Suite 12.2.1.3.0 QuickStart Download* page on Oracle Technology Network.
   - Download, then unzip these files.
     - SOA Suite 12.2.1.3 Part 1 of 2
     - SOA Suite 12.2.1.3 Part 2 of 2

     Download them to the bin folder where you installed Java SE Development Kit, such as `C:\Program Files\Java\jdk-9.0.4\bin`.
   - Right-click **Windows Start menu**, then click `Command Prompt (Admin)`.
   - In the DOS command line, navigate to the bin folder where you downloaded the SOA Suite files (Service Oriented Architecture), enter `java.exe -jar fmw_12.2.1.3.0_soa_quickstart.jar`, then press **Enter** on your keyboard.
   - Follow the prompts in the installer until you finish the installation.
6. Open a web browser, then verify your browser can open the URLs you plan to use for your WSDLs. For details about these WSDLs, see the next section in this topic.

Create Connector

1. Open Oracle JDeveloper.

   For details about how to use Oracle JDeveloper, see *Oracle JDeveloper Tutorials*.
2. In the Select Role dialog, select **Studio Developer**, then click **OK**.
3. Click **Application > New**.
4. In the New Gallery dialog, in tree Categories, expand **General**, then click **Applications**.
5. In the Items area, click **SOA Application > OK**.
6. In the Name Your Application dialog, set the values, then click **Next**.

Attribute	Value
Application Name	Enter any value.
Directory	`C:\JDeveloper\mywork\ConnectorService` You can accept the default value or select some other folder. For example: o `C:\JDeveloper\mywork\ConnectorService`
Application Package Prefix	oracle.apps

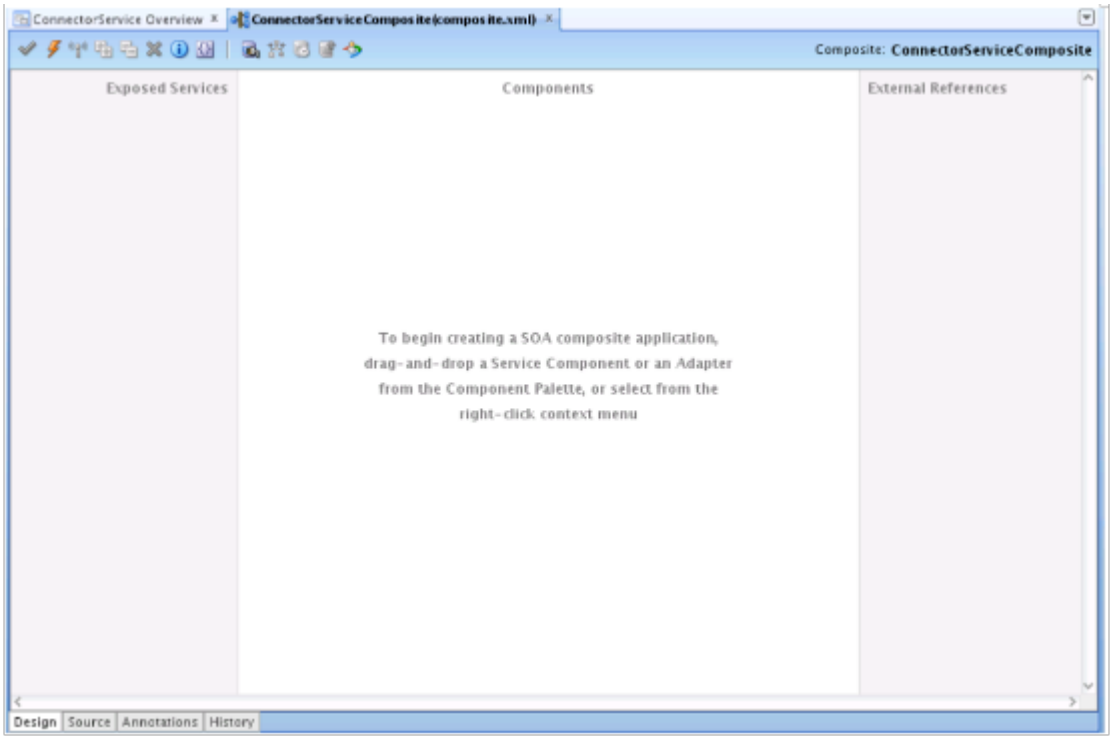
7. In the Name Your Project dialog, set the values, then click **Next**.

Attribute	Value
Project Name	ConnectorServiceComposite You can set any value.
Directory	`C:\JDeveloper\mywork\ConnectorService\ConnectorServiceComposite`

8. In the Configure SOA Settings dialog, set the values, then click **Finish**.

Attribute	Value
Composite Name	ConnectorServiceComposite
Composite Template	Empty Composite

Oracle JDeveloper creates an empty composite.



This composite includes the Exposed Services, Components, and External References panes. You will use these panes during this procedure.

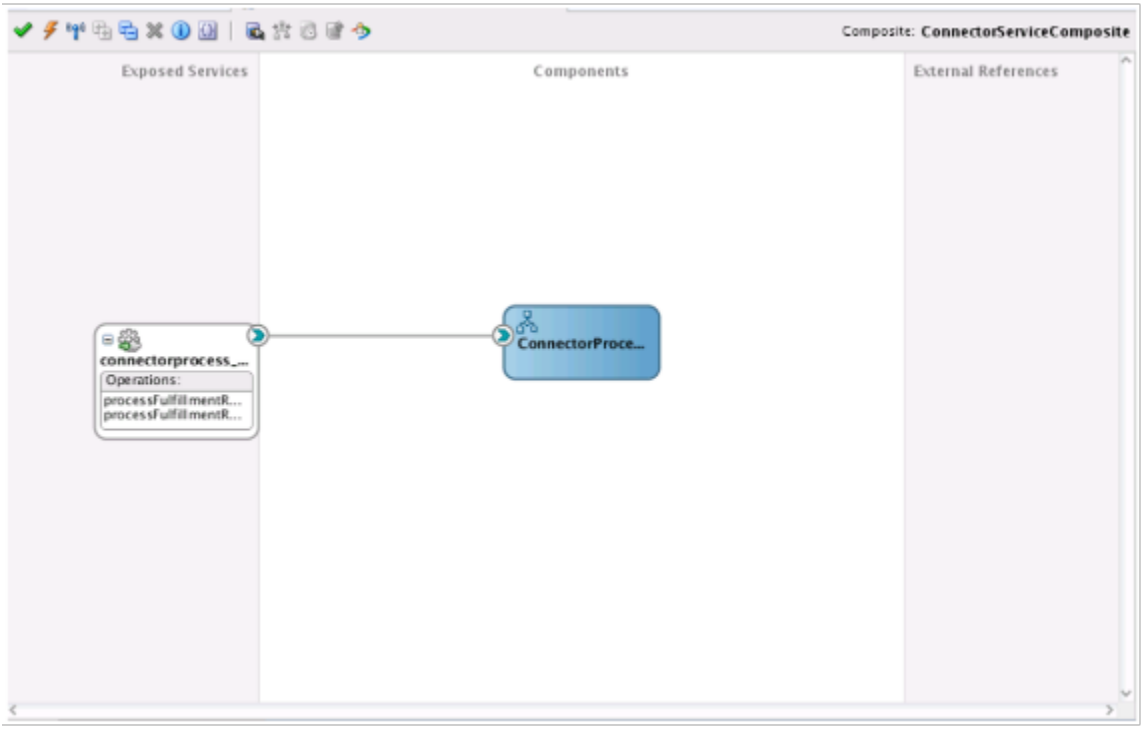
9. Click **File > Save**.
10. Specify the service that will communicate data from your fulfillment system to Order Management.
    - Drag and drop **BPEL Process** from the Component Palette onto the Components pane.
    - In the Create BPEL Process dialog, set the values, then click **OK**. Make sure you set each value in the same sequence that this table displays them.

Attribute	Value
BPEL Specification	BPEL 2.0 Specification
Name	ConnectorProcess
Namespace	`http://xmlns.oracle.com/ConnectorService/ConnectorServiceComposite/ConnectorProcess`
Directory	`C:\JDeveloper\mywork\ConnectorService\ConnectorServiceComposite\SOA\BPEL`
Template Type	Web Service

Attribute	Value
Template	Base on a WSDL
Service Name	connectorprocess_client
Expose as a SOAP Service	Contains a check mark.
WSDL URL	Specify this WSDL. **`https://host:port/soa-infra/services/default/DooTaskExternalInterfaceVirtualPartnersComposite/fulfillmentrequest_client_ep?WSDL`** The FulfillOrderService uses this WSDL. where - **`host`**. Identifies the computer that hosts your Oracle Applications. - **`port`**. Identifies the port that Oracle Applications uses to communicate data. Port is optional. For example, enter this value. **`https://server:port/soa-infra/services/default/DooTaskExternalInterfaceVirtualPartnersComposite/fulfillmentrequest_client_ep?WSDL`** You must identify the host and port. For details, see *Identify Hosts and Ports for Order Management*. This URL locates the WSDL that FulfillOrderService uses. This service communicates data from your fulfillment system to Order Management. It uses these operations and inputs. - The processFulfillmentRequest operation uses the FulfillmentRequestRequestMessage input value. - The processFulfillmentRequestResponseoperation operation uses the FulfillmentRequestResponseMessage input value. If you can't access or use this service for some reason, then see the Use Alternative WSDL Files section in this topic, immediately after this procedure. i. Next to WSDL URL, click **`Find Existing WSDLs`**. ii. In the WSDL Selector dialog, set Location to **`home/user/projects/FulfillmentRequestWSDL`**. where o **`user`** is your user name. iii. In the window below the location you just set, click **`FulfillmentRequest.wsdl`** > **`OK`**. iv. In the Localized Files dialog, add a check mark to **Maintain Original Directory Structure for Imported Files**, then click **OK**.
Port Type	FulfilmentRequest

Attribute	Value
Callback Port Type	FulfilmentRequestCallback

Oracle JDeveloper creates the BPEL process.



11. Create the web service that you use to send the request to your fulfillment system.
    - In the External References pane, right-click, then click **Insert > Direct**.
    - In the Create Direct Binding dialog, set the values, then click **OK**.

Attribute	Value
Name	FulfillmentApplication
Type	Reference
WSDL URL	`FulfillmentApplication/fulfillmentapplication/process_client_ep?WSDL`
Port Type	execute_ptt

Attribute	Value
Callback Port Type	callback_ptt
Copy WSDL	Doesn't contain a check mark.
Transaction Participant	WSDLDriven

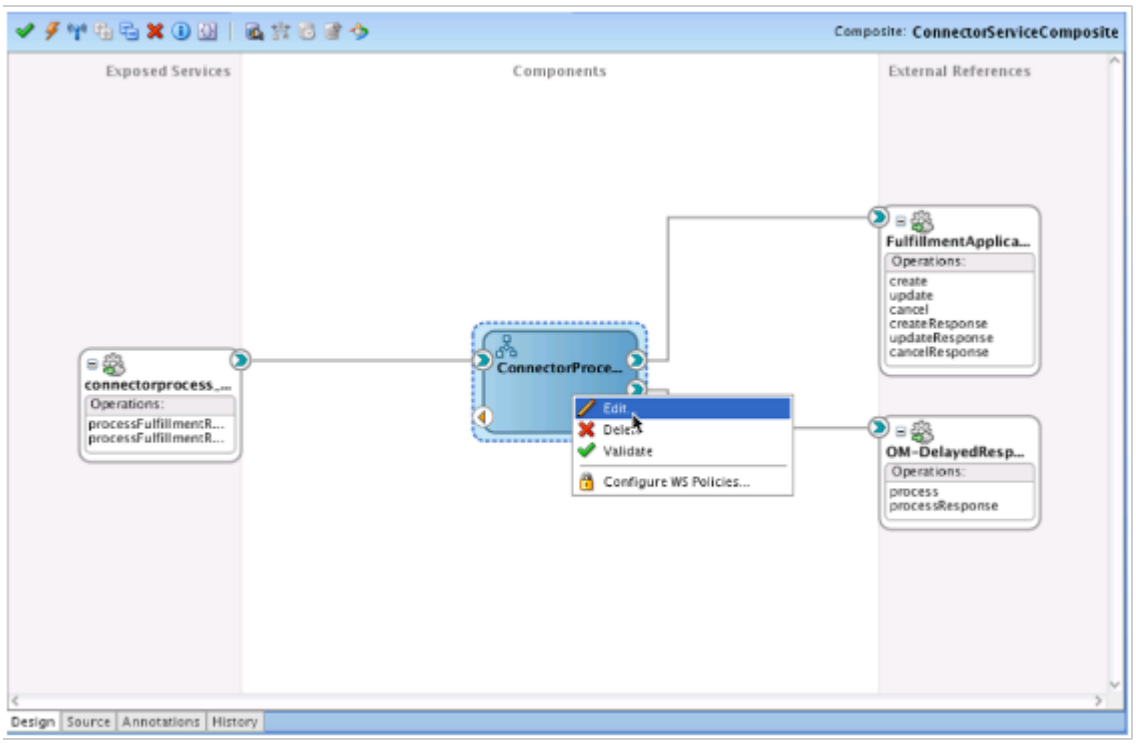
12. Connect the connector to the FulfillmentApplication web service.

    Drag and drop a connection from the Component Palette to create a connection between the ConnectorProcess node in the Components pane and the FulfillmentApplication node in the External References pane.
13. Create the web service you use to send the delayed response back to Order Management.
    - In the External References pane, right-click, then click **Insert > Web Service**.
    - In the Create Web Service dialog, set the values, then click **OK**.

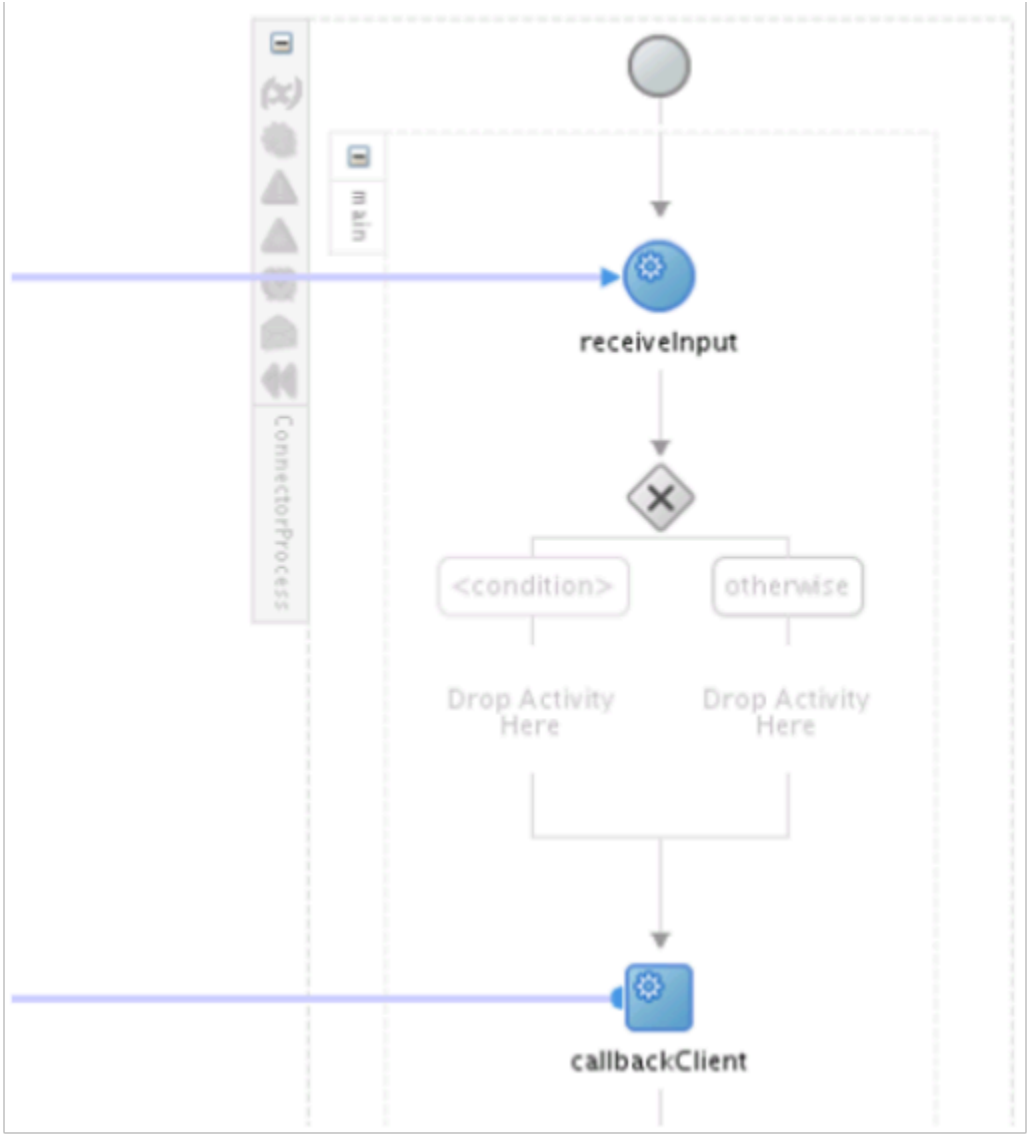
Attribute	Value
Name	OM-DelayedResponse
Type	Reference
WSDL URL	Enter this value. `https://host:port/soa-infra/services/default/DooTaskFulfillOrderResponseInterfaceComposite/fulfillmentresponse?WSDL` where - **host**. Identifies the computer that hosts your Oracle Applications. - **port**. Identifies the port that your Oracle Applications use to communicate data. This URL locates the WSDL that FulfillmentResponseService uses. This service communicates status updates from your fulfillment system to Order Management. It uses these operations and inputs. - The process operation uses the FulfillmentResponseRequestMessage input value. - The processResponse operation uses the FulfillmentResponseResponseMessage input value.
Port Type	FulfillmentResponse
Callback Port Type	FulfillmentResponseCallback

Attribute	Value
Copy WSDL	Doesn't contain a check mark.
Transaction Participant	WSDLDriven

14. Connect the connector to the OM-DelayedResponse web service. Drag and drop a connection from the Component Palette to create a connection between the ConnectorProcess node in the Components pane and the OM-DelayedResponse node in the External References pane.
15. In the Components pane, right-click the **ConnectorProcess** node, click **Edit**.



16. Set up the condition that specifies when to do the Create operation for the service in the fulfillment system.
    - Drag and drop a Switch activity from Component Palette to immediately after the receiveInput activity.

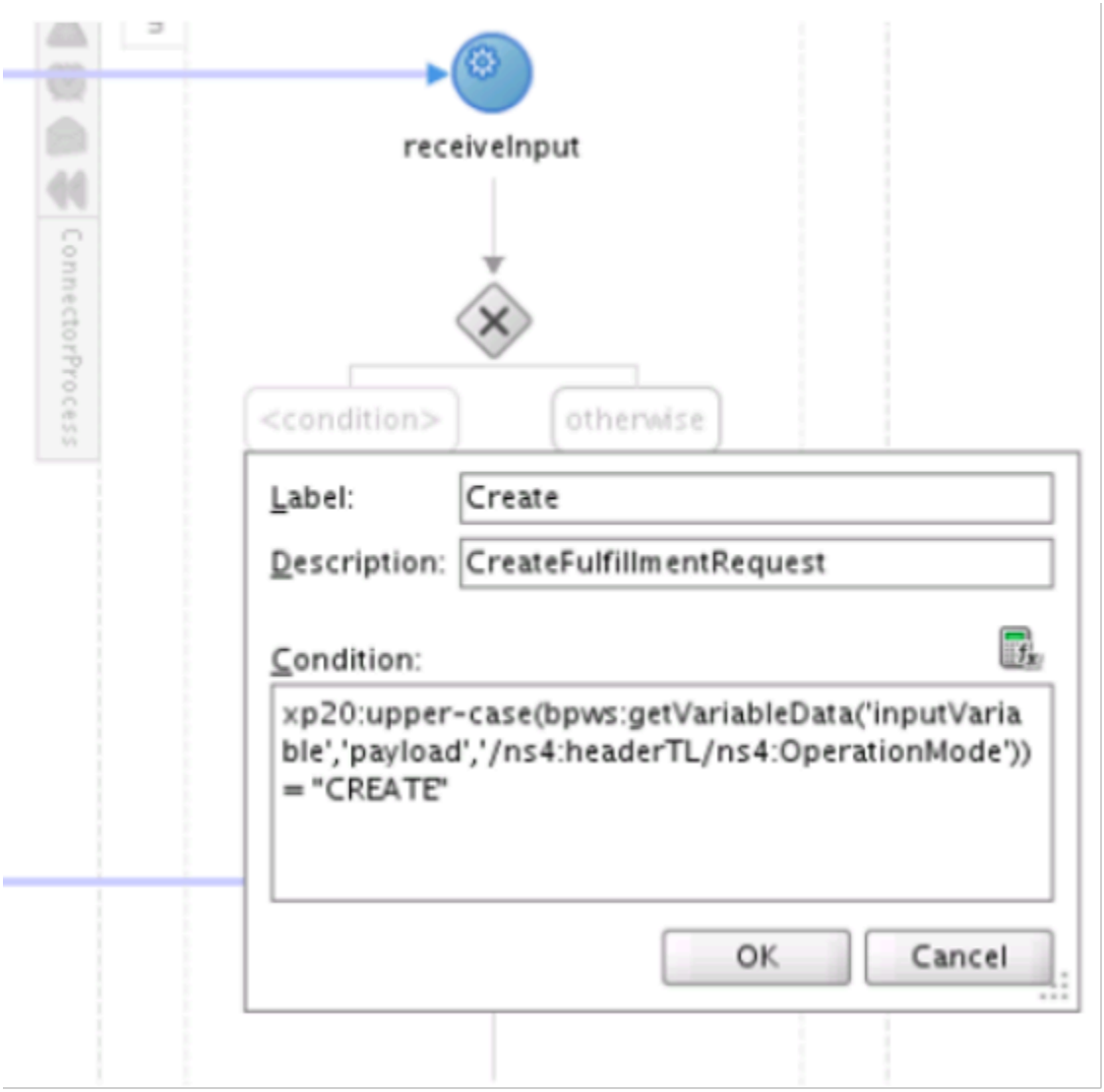


    - Immediately below the switch activity that you just added, click **Condition**, then set the values.

Attribute	Value
Label	Create
Description	CreateFulfillmentReqeuest

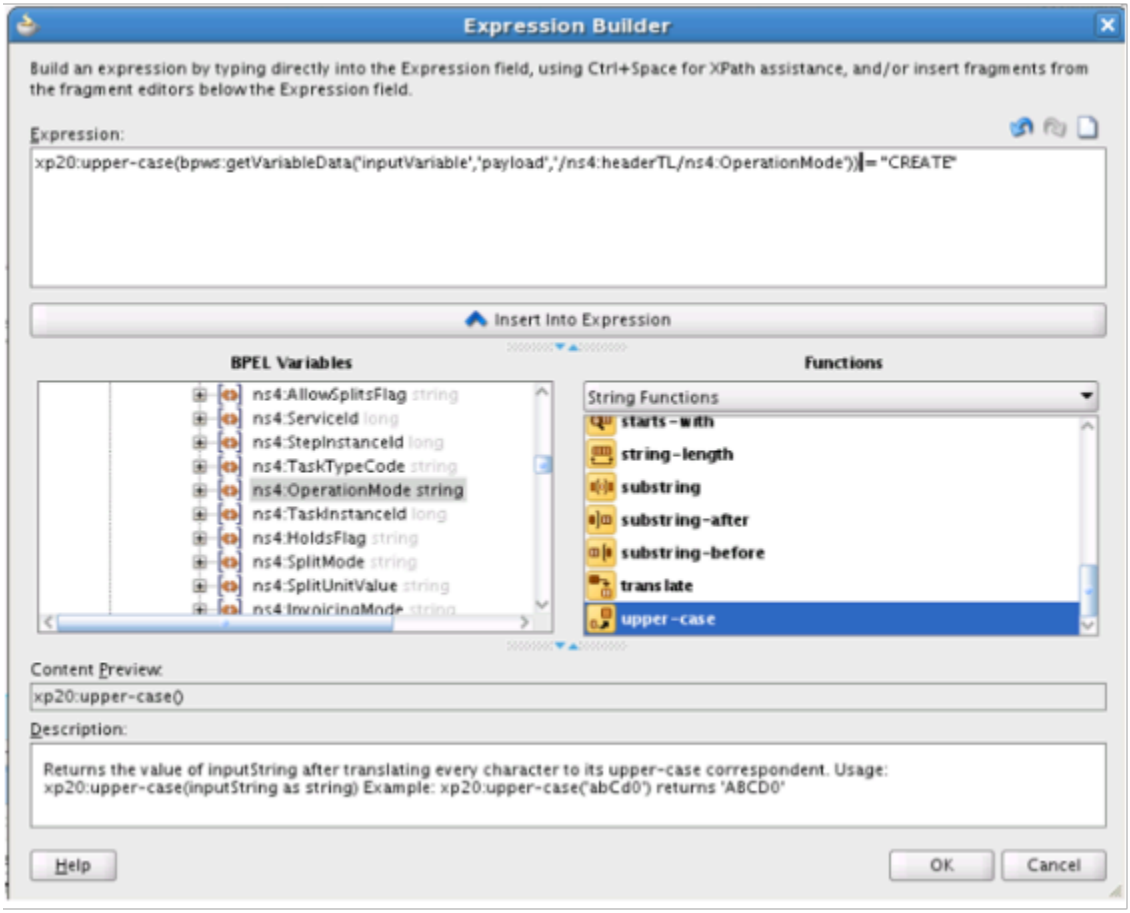
Attribute	Value

You will create a condition.



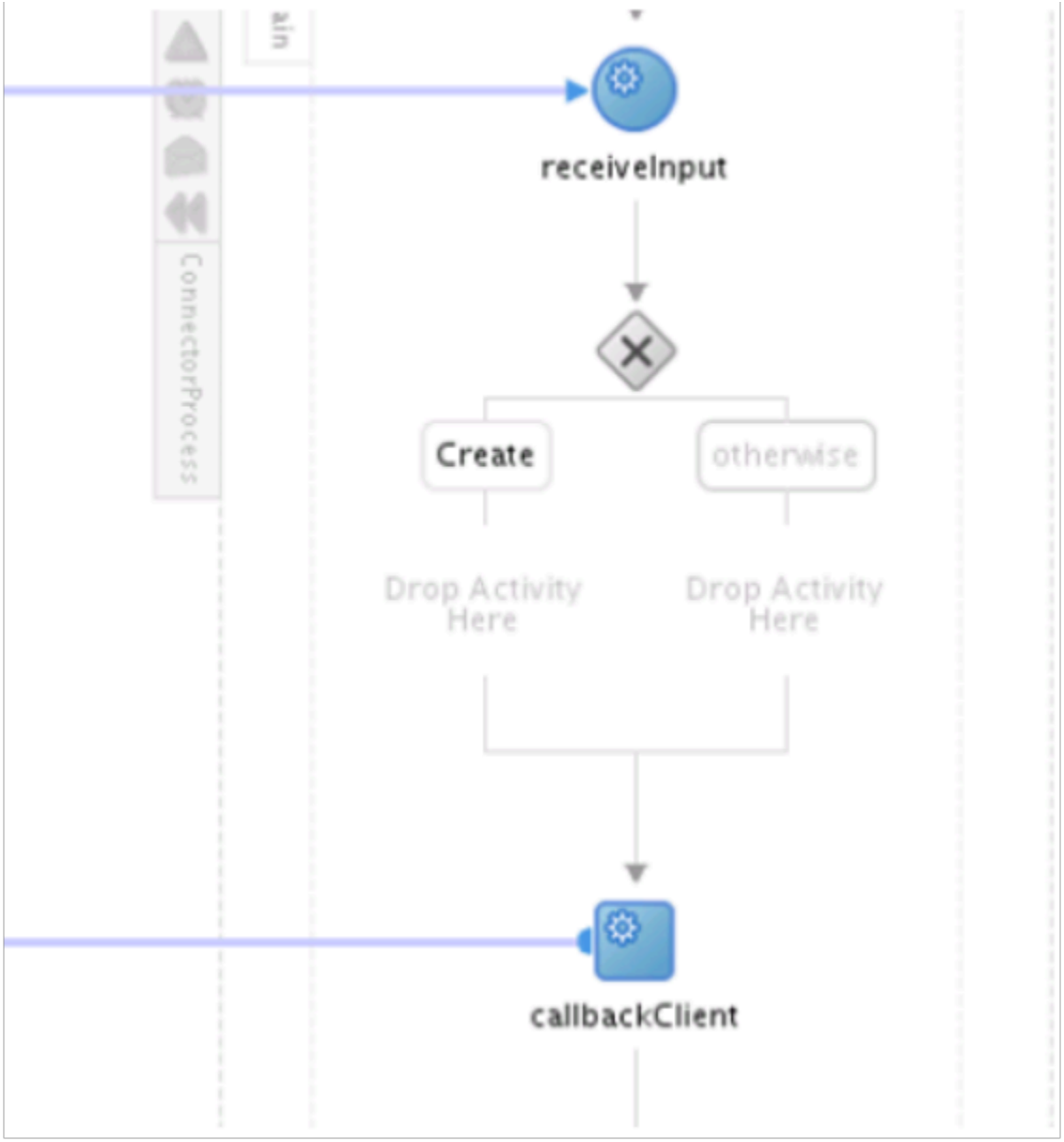
- Click **Expression Builder**.
- In the Expression Builder dialog, in the BPEL Variables window, click `ns4:OperationMode string`.
- In windowFunctions, select **String Functions**, then click **upper-case**.
- Verify that the Expression window contains this value.

```
xp20:upper-case(bpws:getVariableData('inputVariable','payload','/ns4:headerTL/ns4:OperationMode'))
 = "CREATE"
```



- Click **OK**.
- In the Condition dialog, click **OK**.

  The Create step now displays in bold font.



Use Alternative WSDL Files

You typically use the FulfillOrderService service and FulfillmentResponseService service to communicate with your fulfillment system. If you can't access or use these services for some reason, then do a different set up, depending on whether you use extensible flexfields in your implementation.

Extensible Flexfields	Description
You use extensible flexfields.	Click *here* to download the FulfillmentRequest.zip file, then use the WSDL and XSD in the zip.
You don't use extensible flexfields.	Do these steps.

Extensible Flexfields	Description
	1. Go to the Setup and Maintenance work area, then go to the task. o Offering: Order Management o Functional Area: Orders o Task: Manage External Interface Web Service Details 2. On the Manage Connector Details page, click **Download WSDL for External Integration**.

Set the WSDL When Your Implementation Uses Business Events

If you set up Order Management to use a business event, then do these steps.

1. Use this value when you create the FulfillmentApplication connector.

Attribute	Value
WSDL URL	`http://host:port/soa-infra/services/default/DooTaskExternalInterfaceVirtualPartnersComposite/businesseventsconnetor_client_ep?WSDL`

   The connector uses the pushPayload operation and the `body` input value of FulfillOrderService.
2. Make sure the web service operation can do these steps.
   - Accept the user name and password that you enter on the Manage External Interface Web Service Details page. You use this page later during this integration setup after you finish setting up the connector.
   - Receive a payload that uses the signature in the business_events_connnector_payload.xsd file. Get a copy of this file *here*.
3. Use the Associated Connectors tab to specify the connector that references the WSDL that you set up in step 1.

   You can access this tab when you use the Manage Business Events Trigger Points page to set up the business event.

Learn about business events and the Associated Connectors tab. For details, see *Send Notifications from Order Management to Other Systems*.

You typically use FulfillOrderService to handle business events. If you can't access or use this service for some reason, then use a different set up.

Extensible Flexfields	Description
You don't use extensible flexfields.	Click *here* to download the zip that you need, then use the WSDL and XSD in the business_events_wsdl.zip file.
You use extensible flexfields.	1. Go to the Setup and Maintenance work area, then go to the task. o Offering: Order Management o Functional Area: Orders o Task: Manage External Interface Web Service Details

Extensible Flexfields	Description
	2. On the Manage Connector Details page, click **Download WSDL for External Integration**.

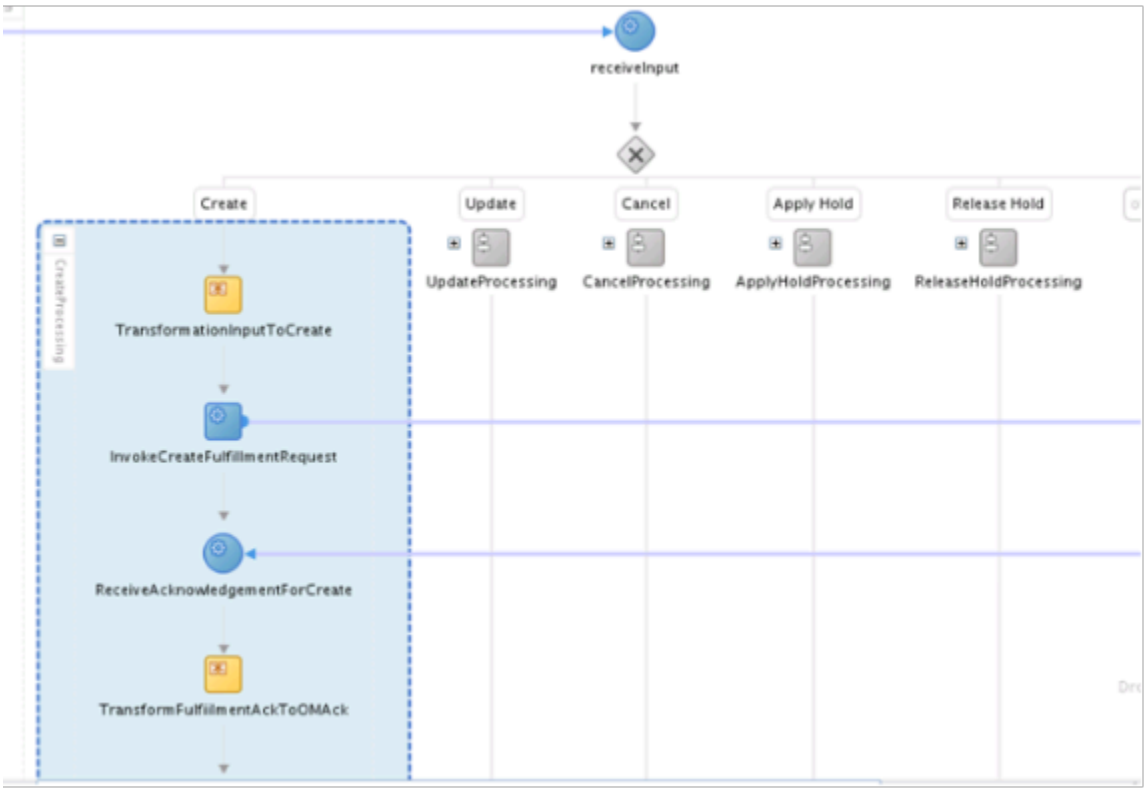
Add Branches for Operations

You will add one branch for each operation that you need for the fulfillment service. For example, this example uses these fulfillment tasks.

- Create
- Update
- Apply Hold
- Release Hold
- Cancel

So you add five switch case branches, one for each task.

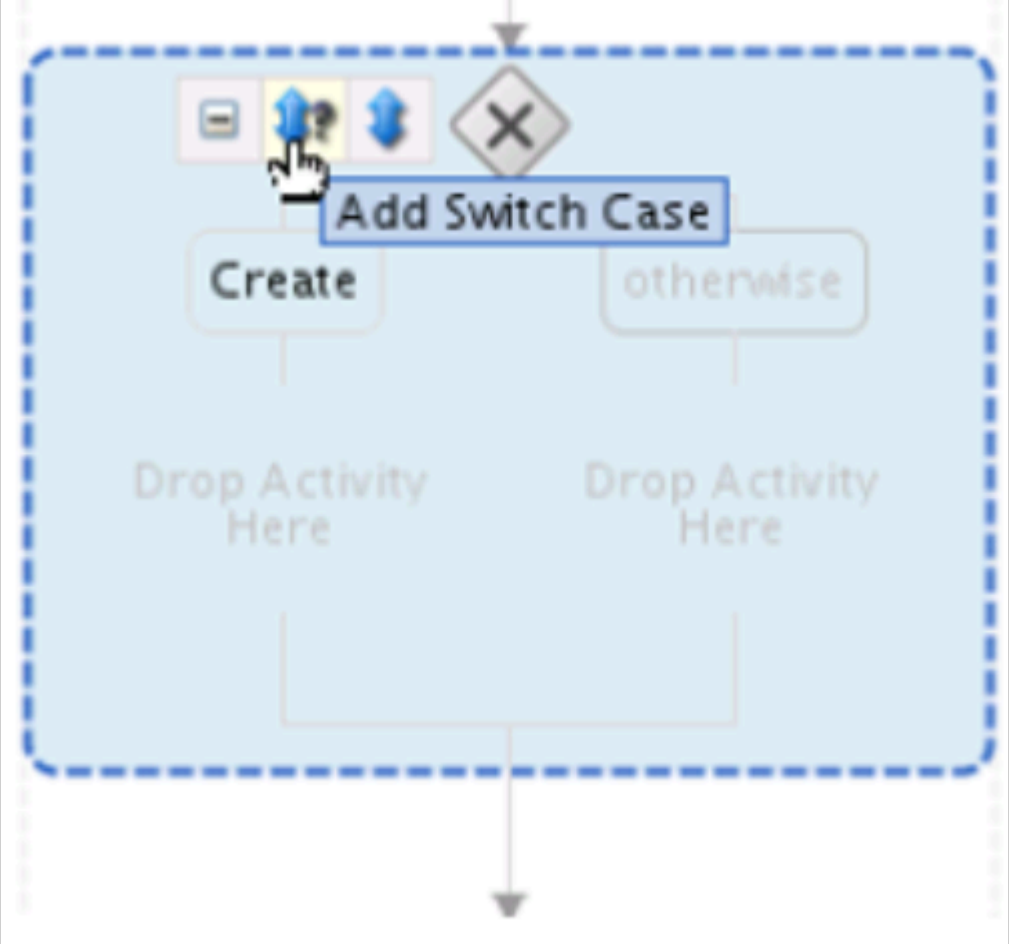
You will add these branches.



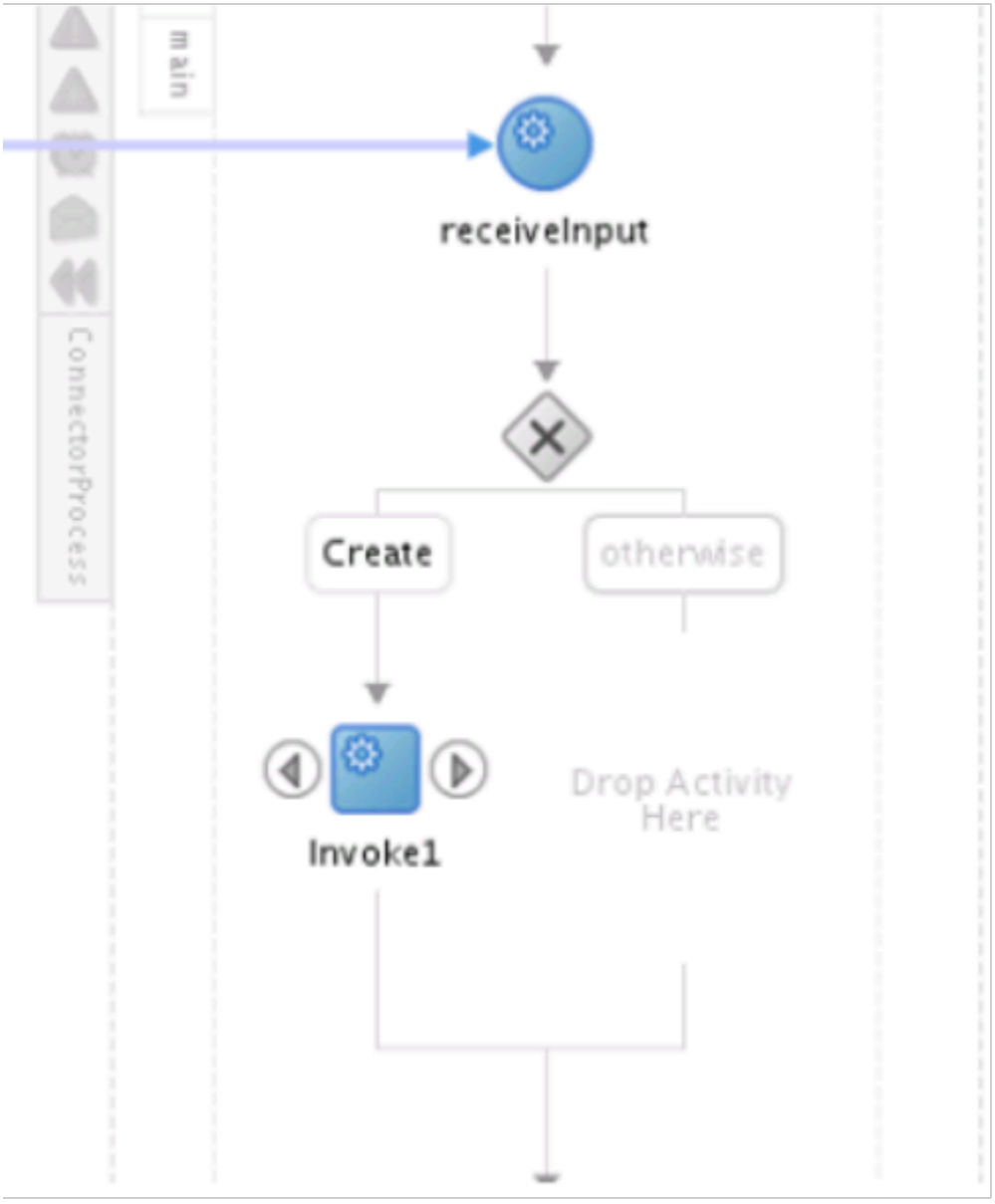
Learn about these tasks and other tasks that you can use. For details, see *Create Your Own Task Type*.

Add branches for operations.

1. Click:



2. Add an activity that calls the branch.
    - Drag and drop from Component Palette onto branch Create.



    - Right-click **Invoke1**, then, in the `Edit Invoke` dialog, set the value.

Attribute	Value
Name	InvokeCreateFulfillmentRequest

Attribute	Value

- Click **Partner Link**.
- In the Partner Link Selector dialog, click **FulfillmentApplication > OK**.
- In the `Edit Invoke` dialog, set the value.

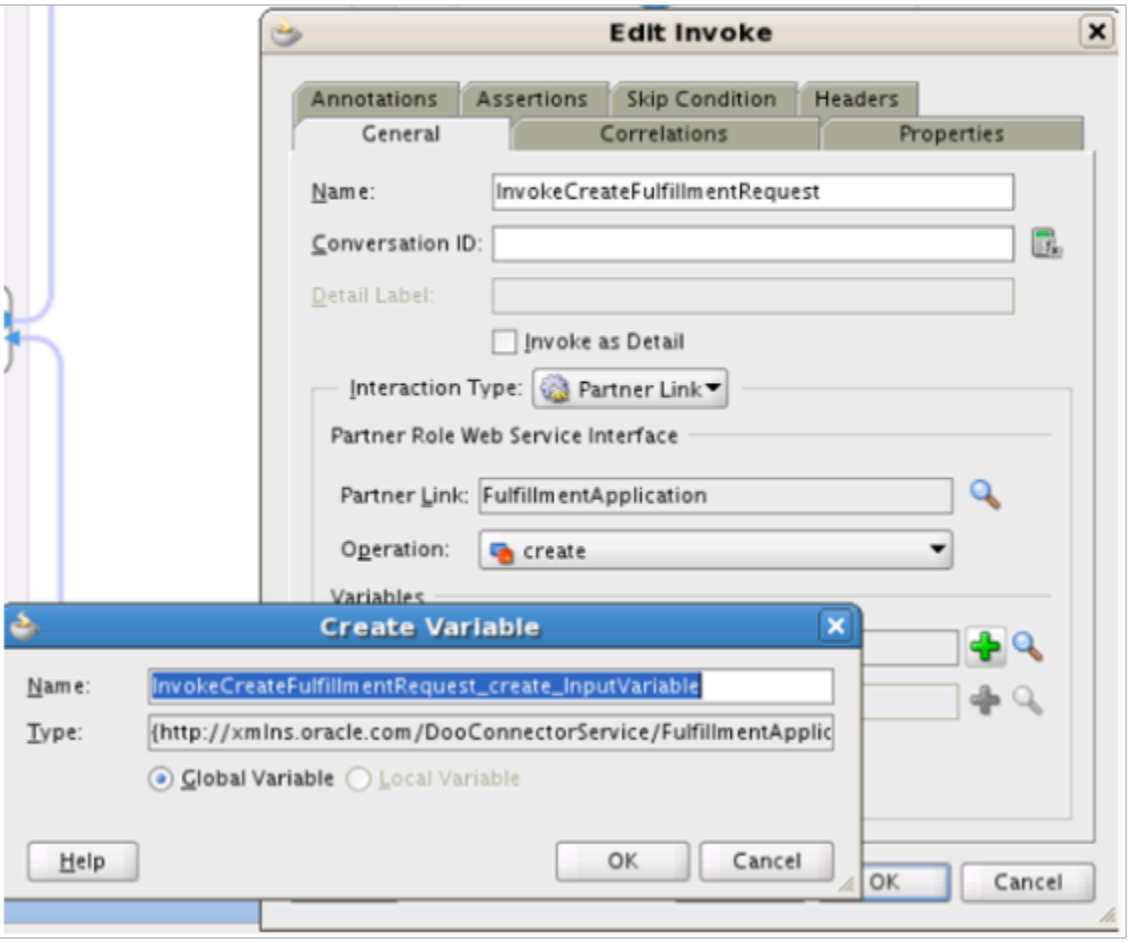
Attribute	Value
Operation	create

- In the `Edit Invoke` dialog, in the Variables section, next to the `Input` window, click **Add**.
- In the Create Variable dialog, set the value, then click **OK**.

Attribute	Value
Name	InvokeCreateFulfillmentRequest_create_InputVariable

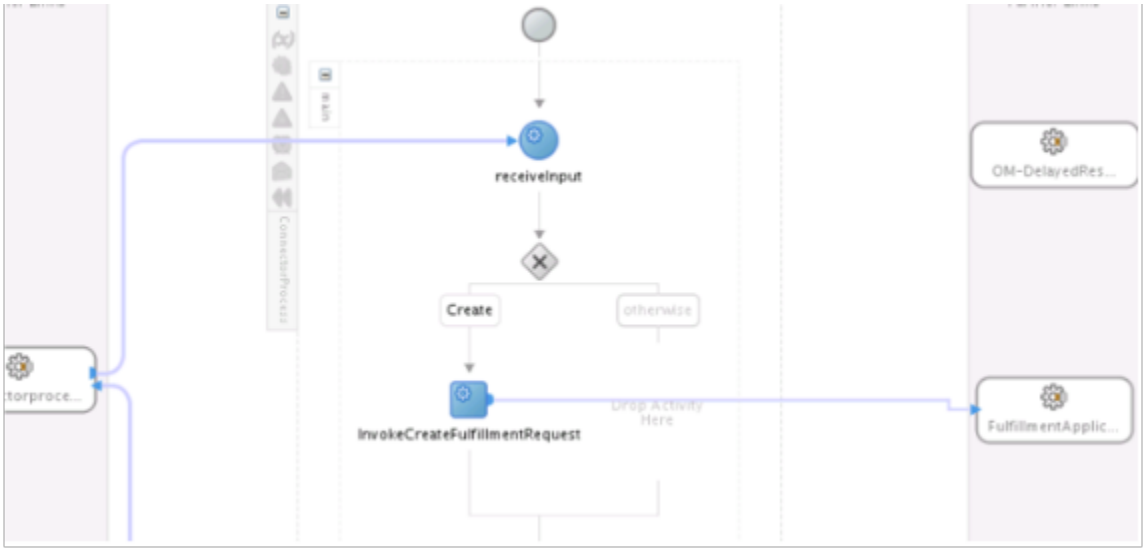
Attribute	Value

For example:

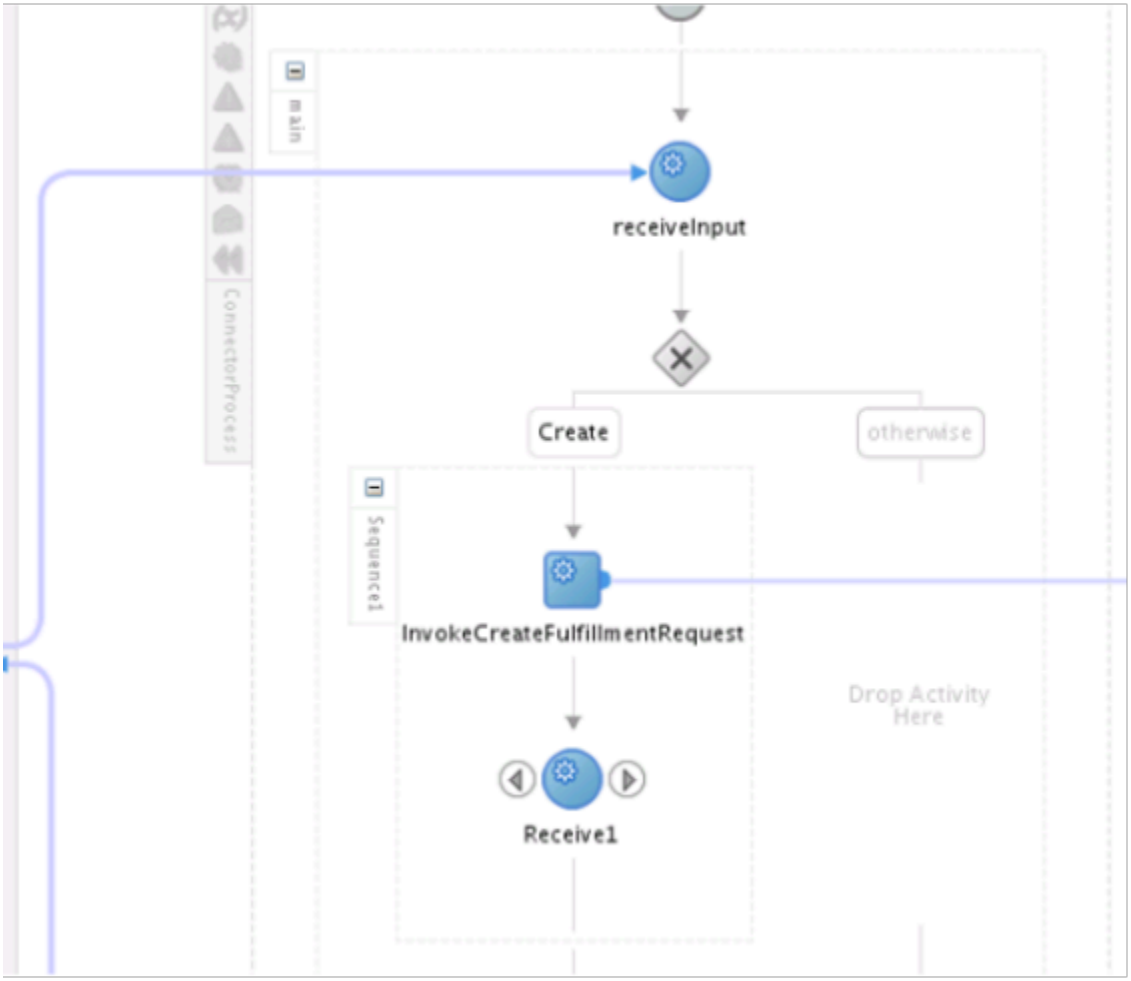


- In the `Edit Invoke` dialog, click **OK**.

  The editor adds the InvokeCreateFulfillmentRequest node to the flow.


receiveInput
Create
otherwise
InvokeCreateFulfillmentRequest
Drop Activity Here
OM-DelayedRes...
FulfillmentApplic...

3. Add an activity that receives a reply from the fulfillment system.
    - Drag a receive activity from Component Palette, then drop it immediately downstream of the InvokeCreateFulfillmentRequest node.

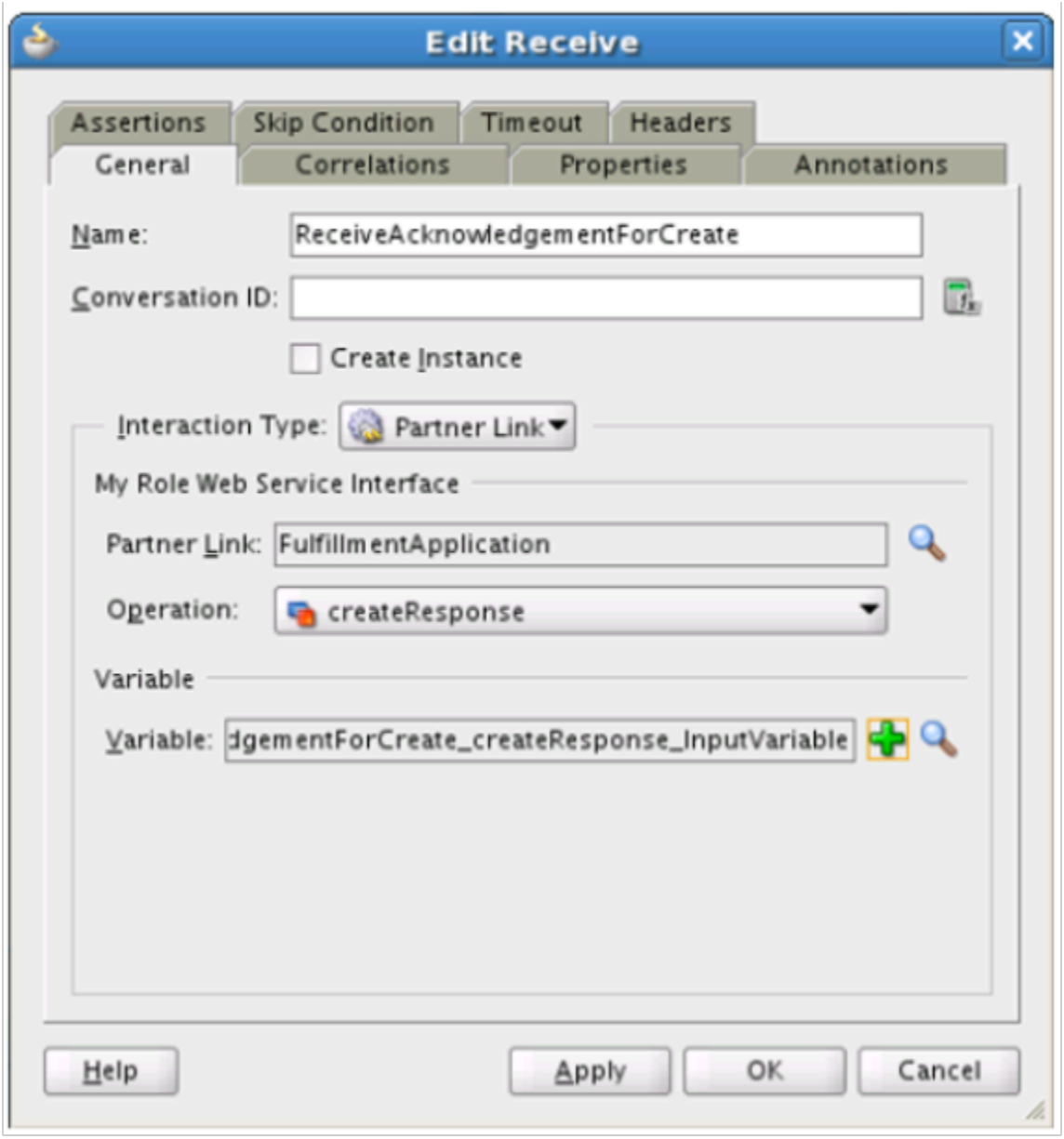


    - Right-click the **Receive1** node that you just added.
    - In the Edit Receive dialog, set values, then click **OK**.

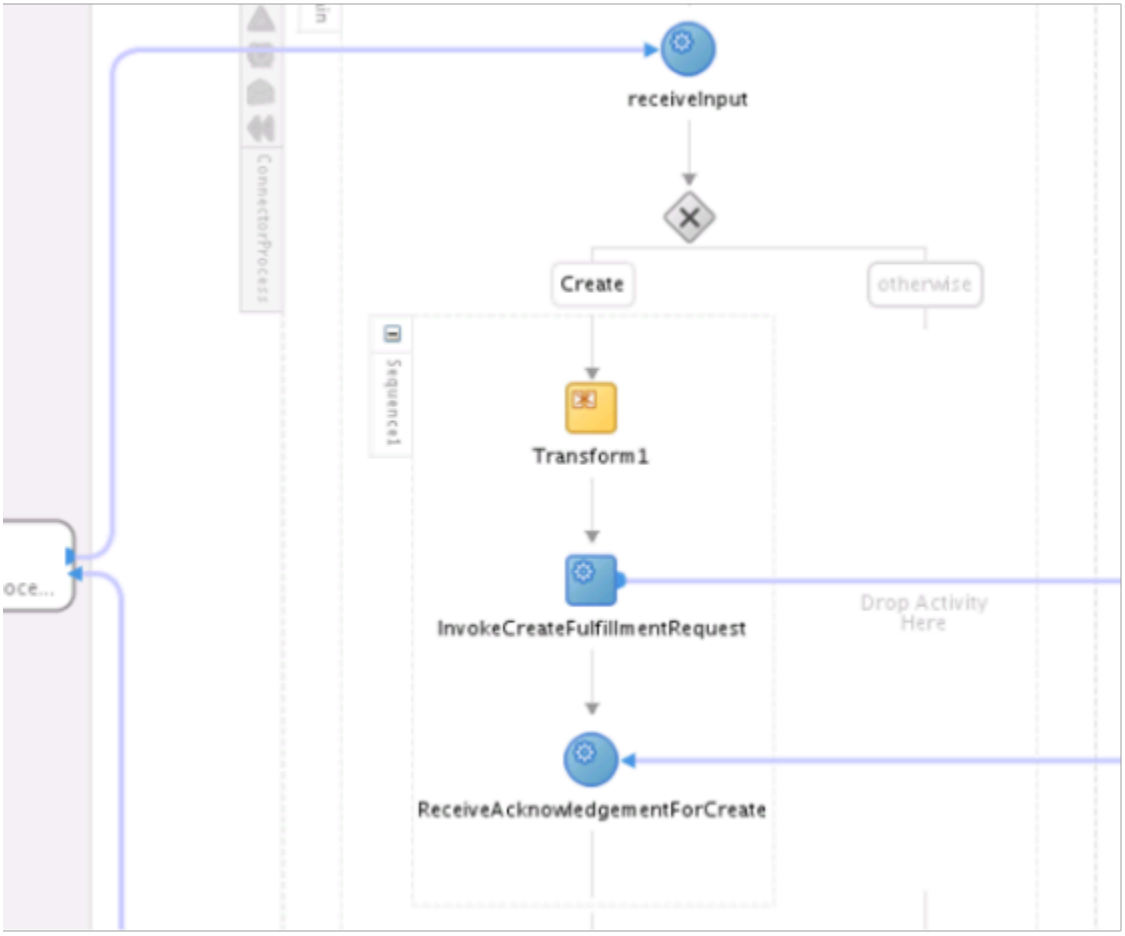
Attribute	Value
Name	ReceiveAcknowledgementForCreate
Conversation Id	Leave empty.
Create Instance	Leave empty.
Interact Type	Partner Link.

Attribute	Value
Partner Link	FulfillmentApplication
Operation	createResponse
Variable	ReceiveAcknowledgementForCreate_createResponse_InputVariable

For example:



4. Add an activity that transforms the input value into a message that your fulfillment system can understand.
    - Drag a transform activity from Component Palette, then drop it immediately upstream of the InvokeCreateFulfillmentRequest node.



    - Right-click the **Transform1** activity that you just added.
    - In the Edit Transform dialog, on the General tab, set the value, then click **Transformation**.

Attribute	Value
Name	TransformationInputToCreate

    - On the Transformation tab, in the Source area, click **Add**, then set the values.

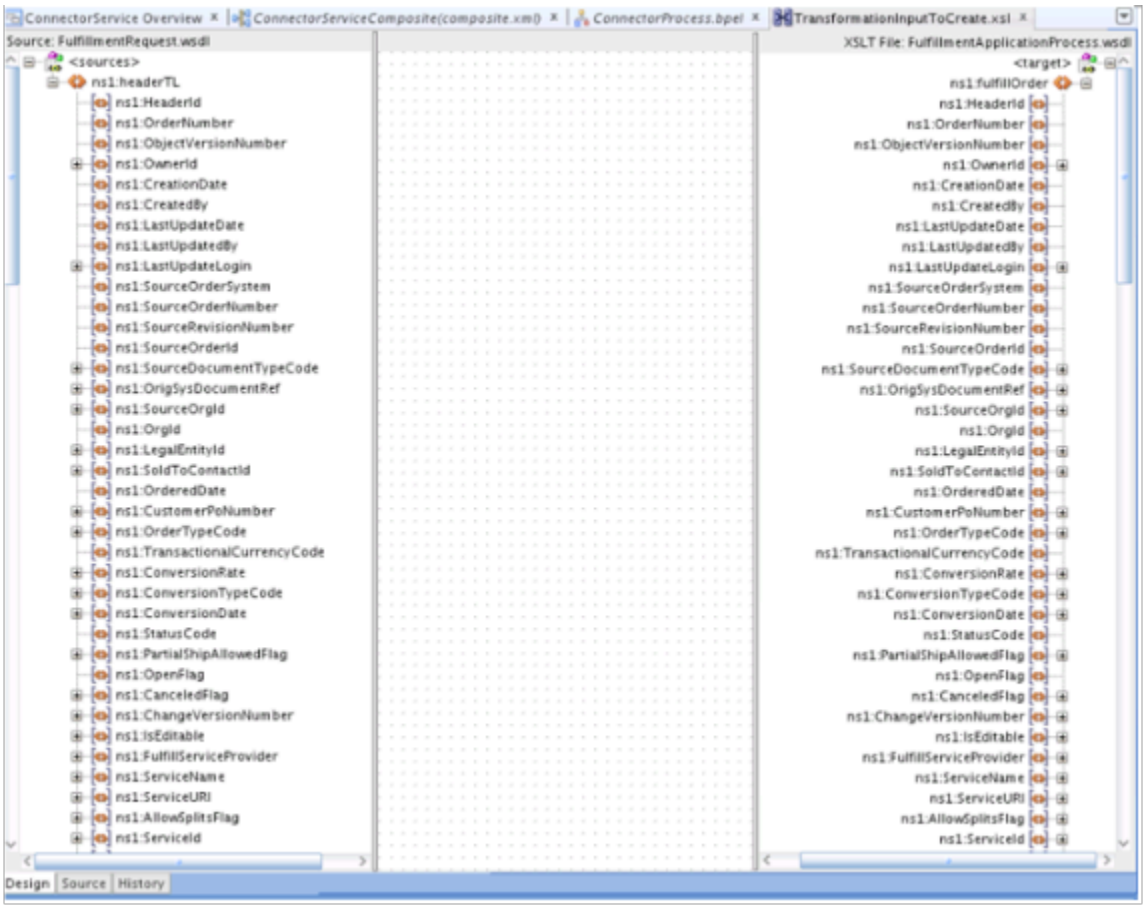
Attribute	Value
Variable	inputVariable
Part	payload

Attribute	Value

- Set the values.

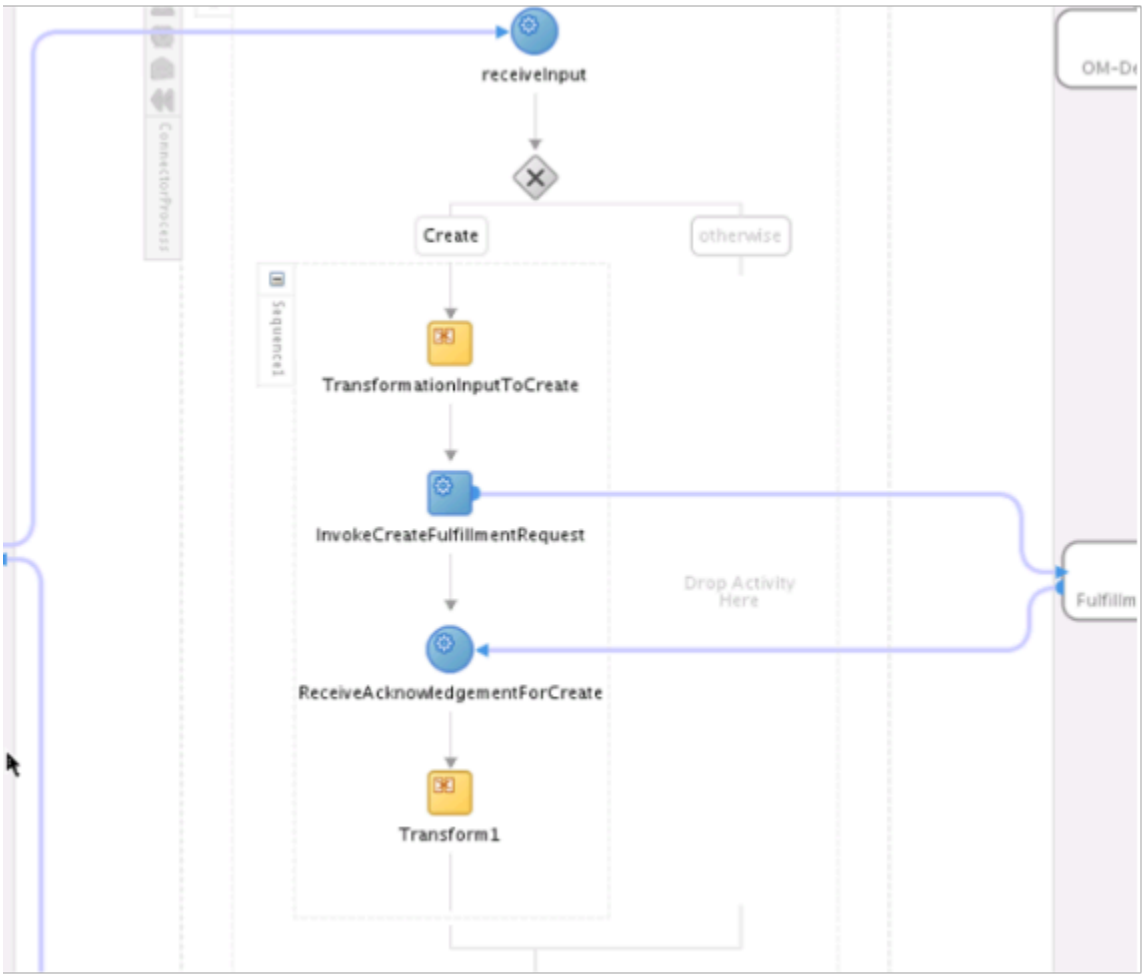
Attribute	Value
Target Variable	InvokeCreateFulfillmentRequest_create_InputVariable
Target Part	payload
Mapper File	xsl/TransformationInputToCreate

- In the Mapper File area, click **Add**.
- In the page that displays, connect each attribute that you must send from Order Management to your fulfillment system.



- In the Edit Transform dialog, click **Apply > OK**.

5. Add an activity that transforms the reply that your fulfillment system sends into a message that Order Management can understand.
    - Drag a transform activity from Component Palette, then drop it immediately downstream of the ReceiveAcknowledgementForCreate node.



    - Right-click the Transform1 activity that you just added.
    - In the Edit Transform dialog, on the General tab, set the value, then click **Transformation**.

Attribute	Value
Name	TransformFulfillmentAckToOMAck

    - On the Transformation tab, in the Source area, click **Add**, then set the values.

Attribute	Value
Variable	ReceiveAcknowledgementForCreate_createResponse_InputVariable

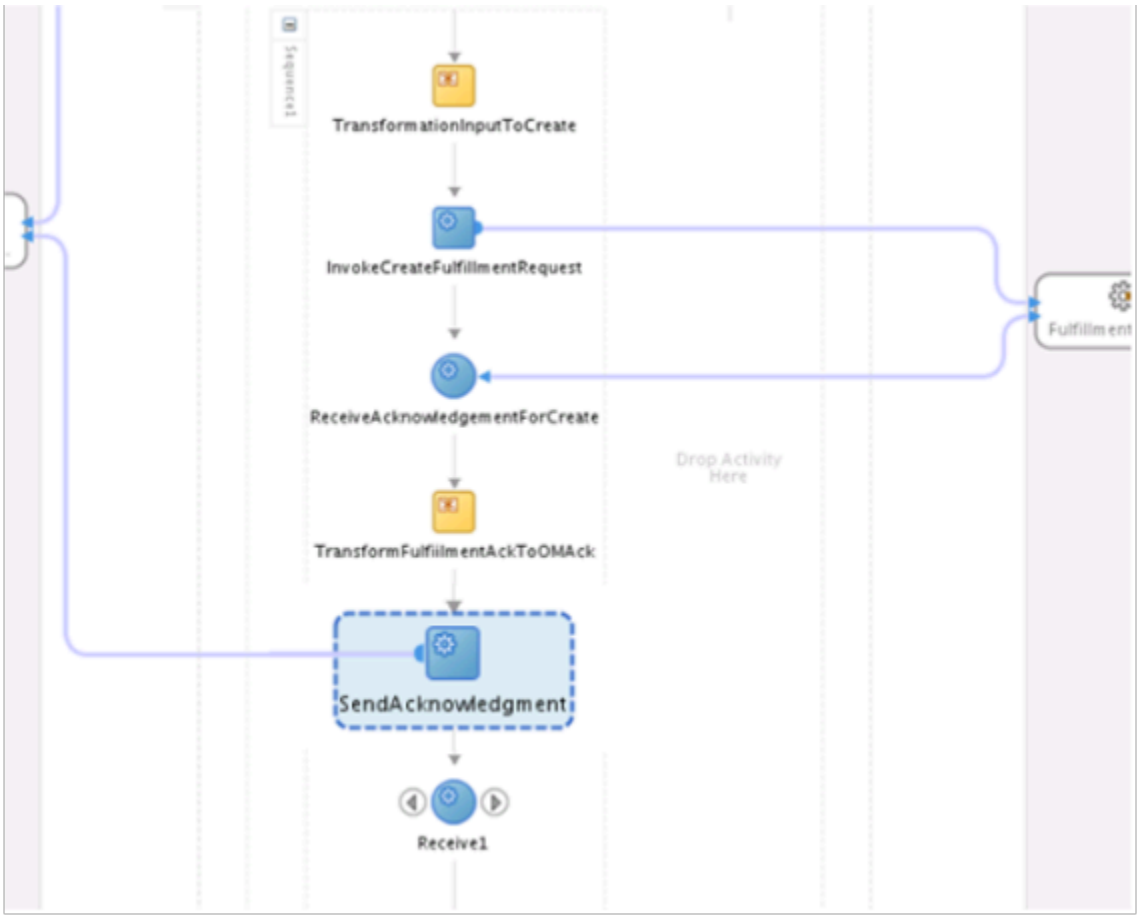
Attribute	Value
Part	payload

- Set the values.

Attribute	Value
Target Variable	outputVariable
Target Part	payload
Mapper File	xsl/TransformFulfillmentAckToOMAck

- In the Mapper File area, click **Add**.
- In the page that displays, connect each attribute that you must send from the fulfillment system to Order Management.
- In the Edit Transform dialog, click **Apply > OK**.

6. Specify how to send the acknowledgment to Order Management.
    - On the ConnectorProcess process that you created earlier, drag an activity from the Component Palette, then drop it immediately downstream of the TransformFulfillmentAckToOMAck activity.
    - Drag a Receive activity from Component Palette, then drop it immediately downstream of the SendAcknowledgement activity that you just added.



This activity receives the delayed response from the fulfillment system.

- Right-click **Receive1**.
- In the Edit Receive dialog, on the General tab, set the values, then click **Apply > OK**.

Attribute	Value
Name	ReceiveDelayedResponseFromFulfillment
Conversation Id	Leave empty
Create Instance	Leave empty

Attribute	Value
Interaction Type	Partner Link
Partner Link	FulfillmentApplication
Operation	CreateResponse
Variable	ReceiveDelayedResponseFromFulfillment_CreateResponse_InputVariable

7. Repeat steps 1 through 6 for the Update operation.

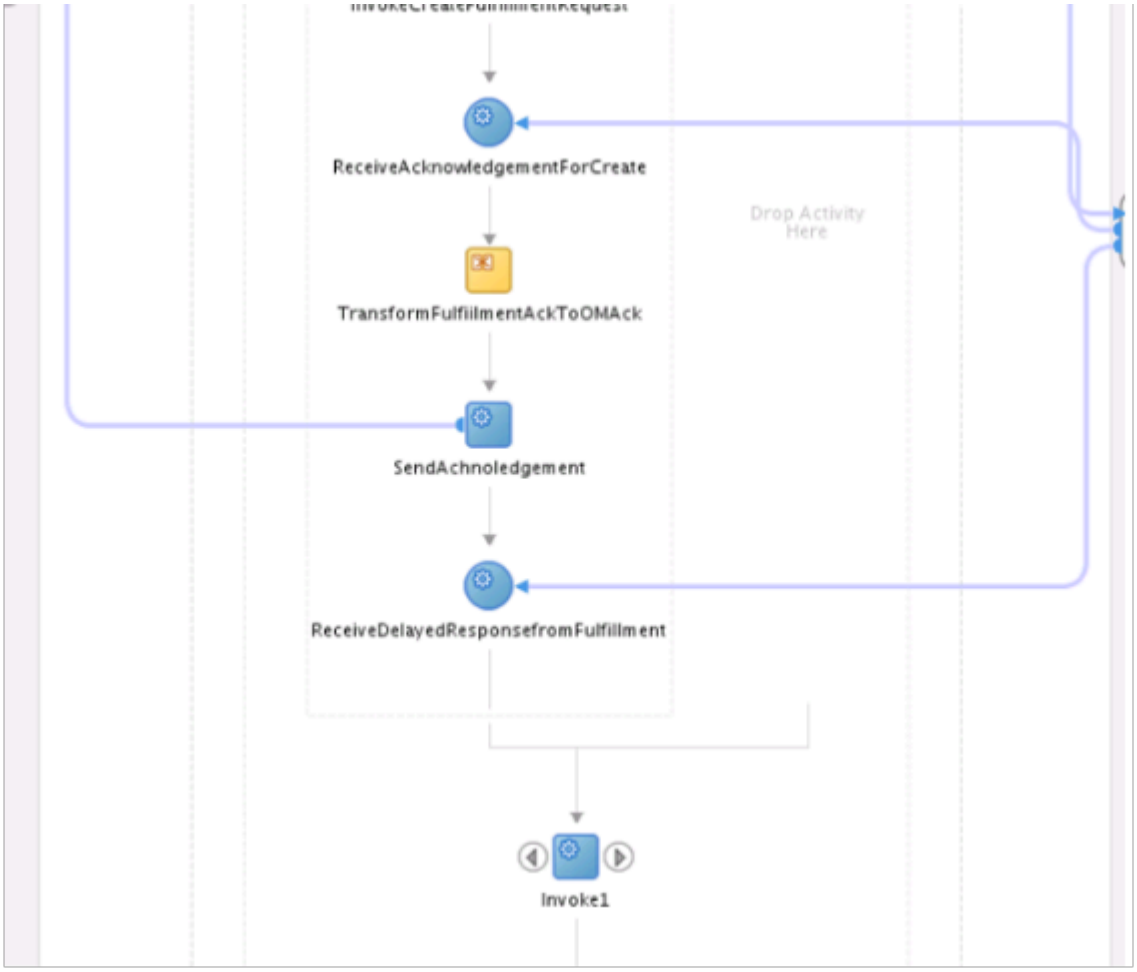
   For each operation, modify the values slightly to reflect the operation. For example, add the Update operation to the Update branch, and use ReceiveAcknowledgementForUpdate instead of ReceiveAcknowledgementForCreate.
8. Repeat steps 1 through 6 for the Apply Hold operation.
9. Repeat steps 1 through 6 for the Release Hold operation.
10. Repeat steps 1 through 6 for the Cancel operation.

Set Up Delayed Response

You will set up the flow that sends a delayed response from your fulfillment system to Order Management. You use the same payload for each operation so you can use a single flow for all operations.

Set up the delayed response.

1. Drag an activity from Component Palette, then drop it immediately downstream of the switch node.

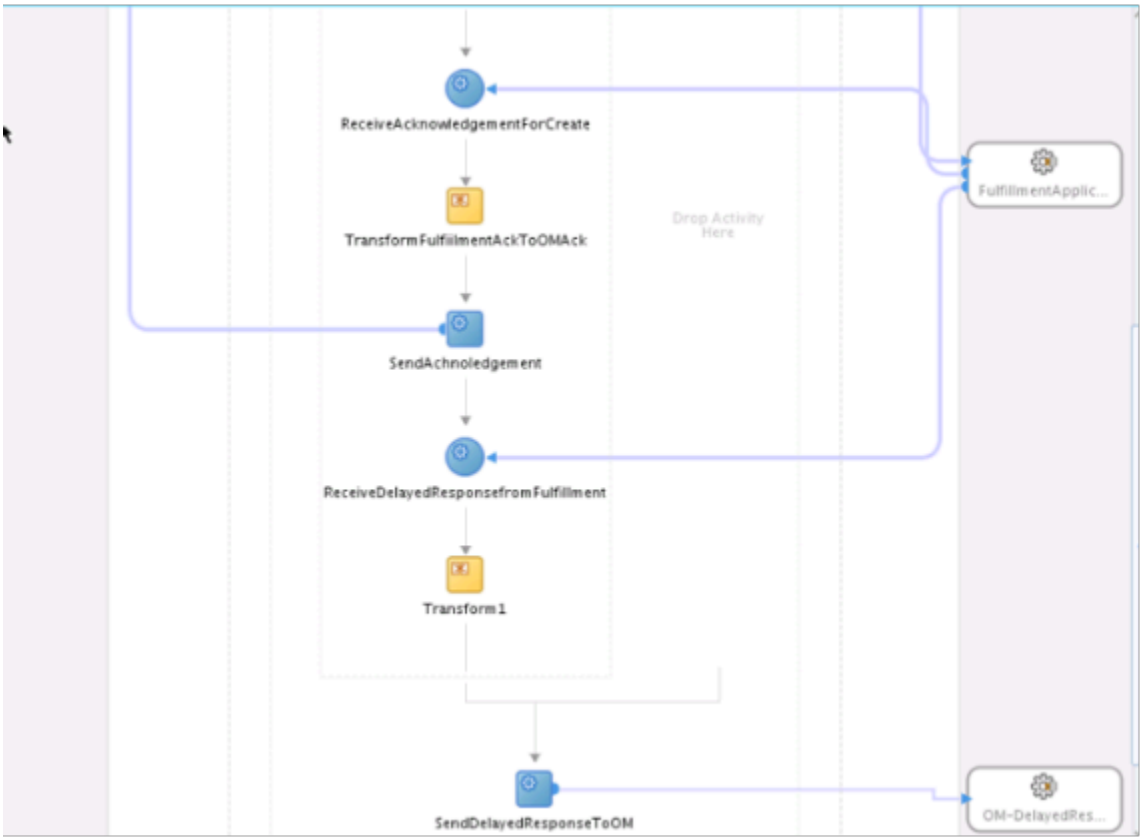


2. Right-click **Invoke1**.
3. In the `Edit Invoke` dialog, on the General tab, set the values, then click **Apply > OK**.

Attribute	Value
Name	SendDelayedResponseToOM
Conversation Id	Leave empty
`Invoke as Detail`	Leave empty
Interaction Type	Partner Link
Partner Link	OM-DelayedResponse

Attribute	Value
Operation	process
Variable	SendDelayedResponseToOM_process_InputVariable

4. Transform the delayed response message that your fulfillment system sends to a message that Order Management can understand.
    - Drag the Transform activity from Component Palette, then, in the create branch, drop it immediately downstream of the ReceiveDelayedResponseFromFulfillment activity.



    - Right-click the **Transform1** activity that you just added.
    - In the Edit Transform dialog, on the General tab, set the value, then click **Transformation**.

Attribute	Value
Name	TransformDelayedResponseToOM

Attribute	Value

- On the Transformation tab, in the Source area, click **Add**, then set the values.

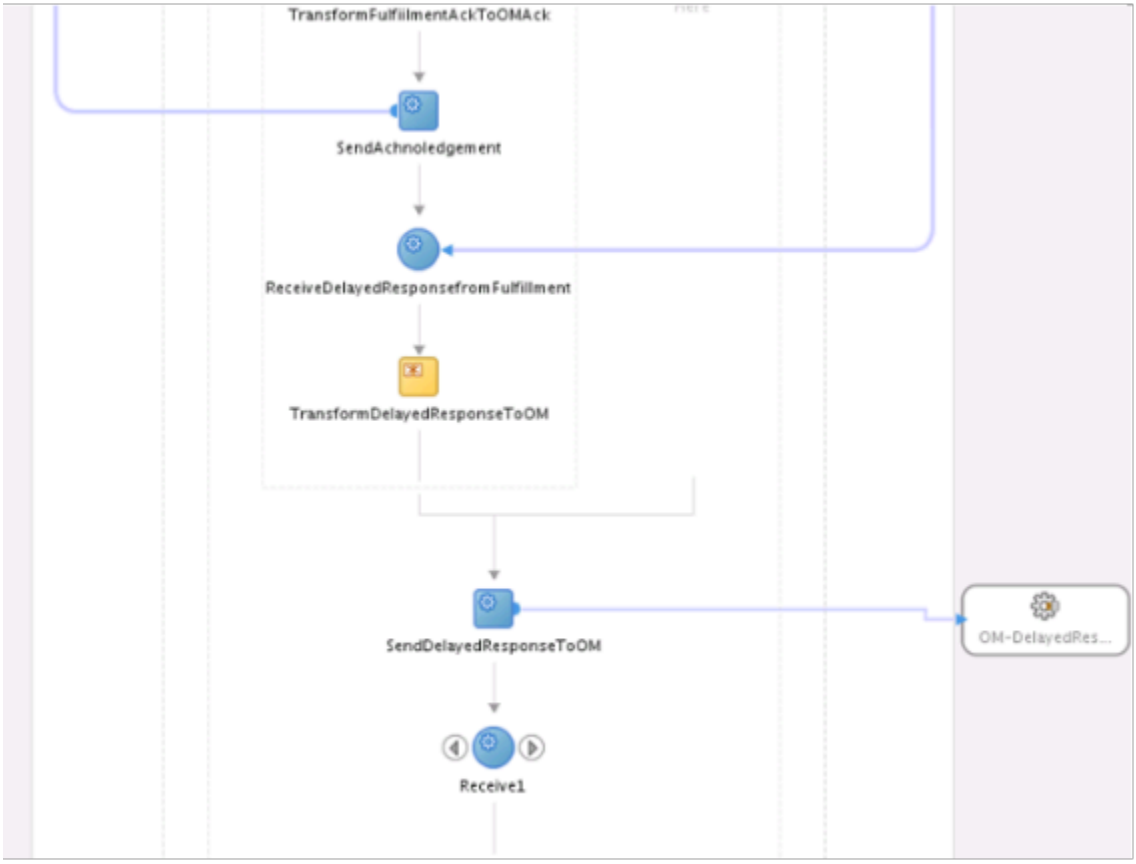
Attribute	Value
Variable	ReceiveDelayedResponseFromFulfillment_createResponse_InputVariable
Part	payload

- Set the values.

Attribute	Value
Target Variable	SendDelayedResponseToOM_process_InputVariable
Target Part	payload
Mapper File	xsl/TransformationDelayedResponseToOM

- In the Mapper File area, click **Add**.
- On the page that displays, connect each attribute that you must send as part of the delayed response from your fulfillment system to Order Management.
- In the Edit Transform dialog, click **Apply > OK**.

5. Add an activity that receives a reply from Order Management.
    - Drag a receive activity from Component Palette, then drop it immediately downstream of the SendDelayedResponseToOM activity.

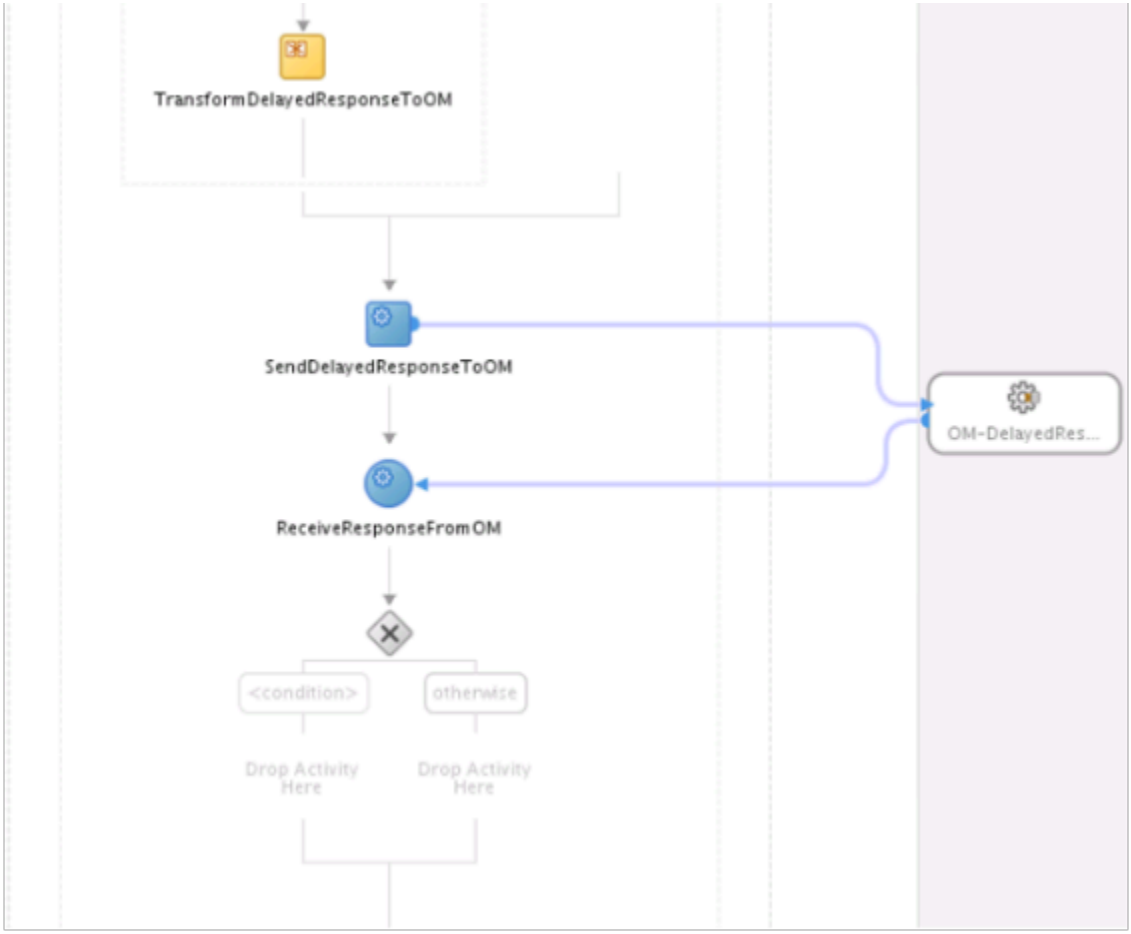


    - Right-click the Receive1 node that you just added.
    - In the Edit Receive dialog, set values, then click **Apply > OK**.

Attribute	Value
Name	ReceiveResponseFromOM
Conversation Id	Leave empty.
Create Instance	Leave empty.
Interaction Type	Partner Link.
Partner Link	OM-DelayedResponse

Attribute	Value
Operation	processResponse
Variable	ReceiveResponseFromOM_processResponse_InputVariable

6. Set up a condition that specifies how to handle errors that Order Management might send.
    - Drag and drop the Switch activity from Component Palette to immediately downstream of the ReceiveResponseFromOM activity.

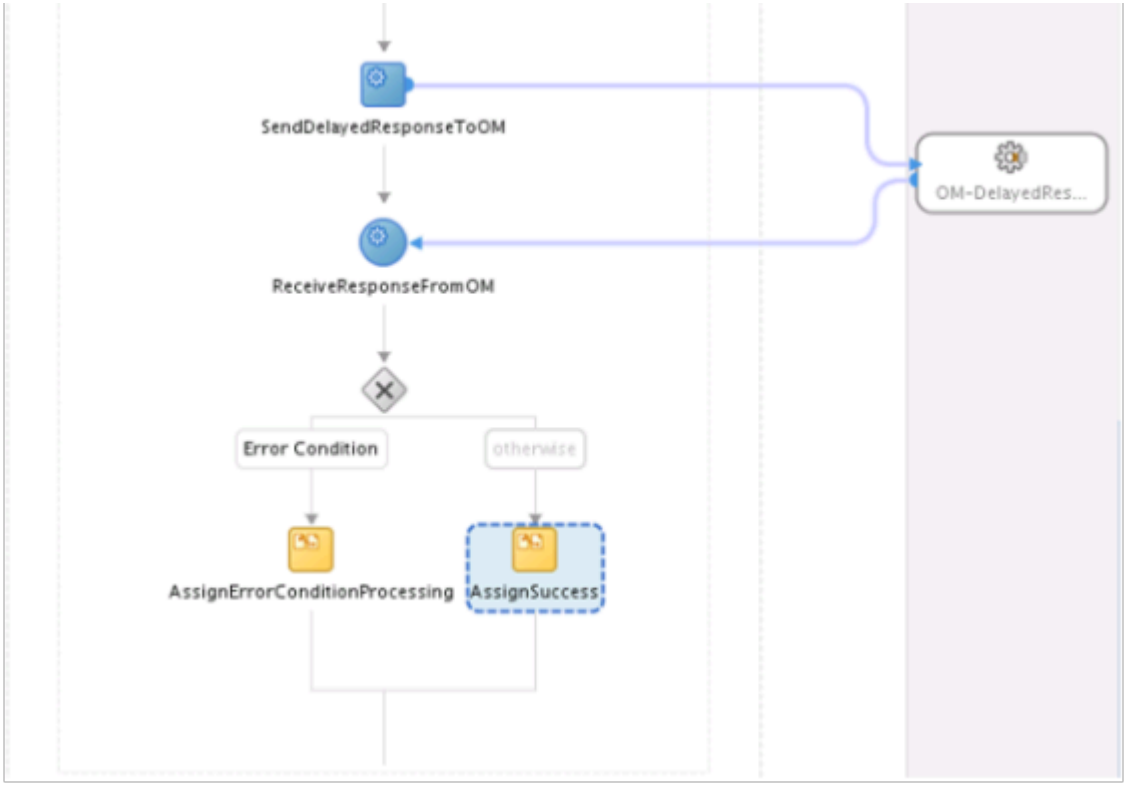


    - Immediately below the switch activity that you just added, click **Condition**, set the values, then click **OK**.

Attribute	Value
Label	ErrorCondition
Description	Handle errors when processing a delayed response

Attribute	Value
Condition	`xp20:upper-case(bpws:getVariableData('ReceiveResponseFromOM_processResponse_InputVariabl ns3:FulfillmentResponse/ns3:Status') !="SUCCESS"`

7. Add nodes.
    - **AssignErrorConditionProcess**. Specify actions to take in the Error Condition branch when an error happens.
    - **AssignSucess**. Specify actions to take in the Otherwise branch when an error doesn't happen.



Including Charges in Delayed Response

If the delayed response from the web service that calls Order Management includes a charge, then Order Management deletes all previous charges and replaces them with the values that the response sends. Therefore, you must make sure the response includes all charges that Order Management sent in the outgoing request.

For example, if Order Management sends the ORA_SHIPPING_FREIGHT value and the QP_NET_PRICE value for the ChargeTypeCode charge, then the response must include ORA_SHIPPING_FREIGHT and QP_NET_PRICE for ChargeTypeCode.

Secure Connector and Deploy Project

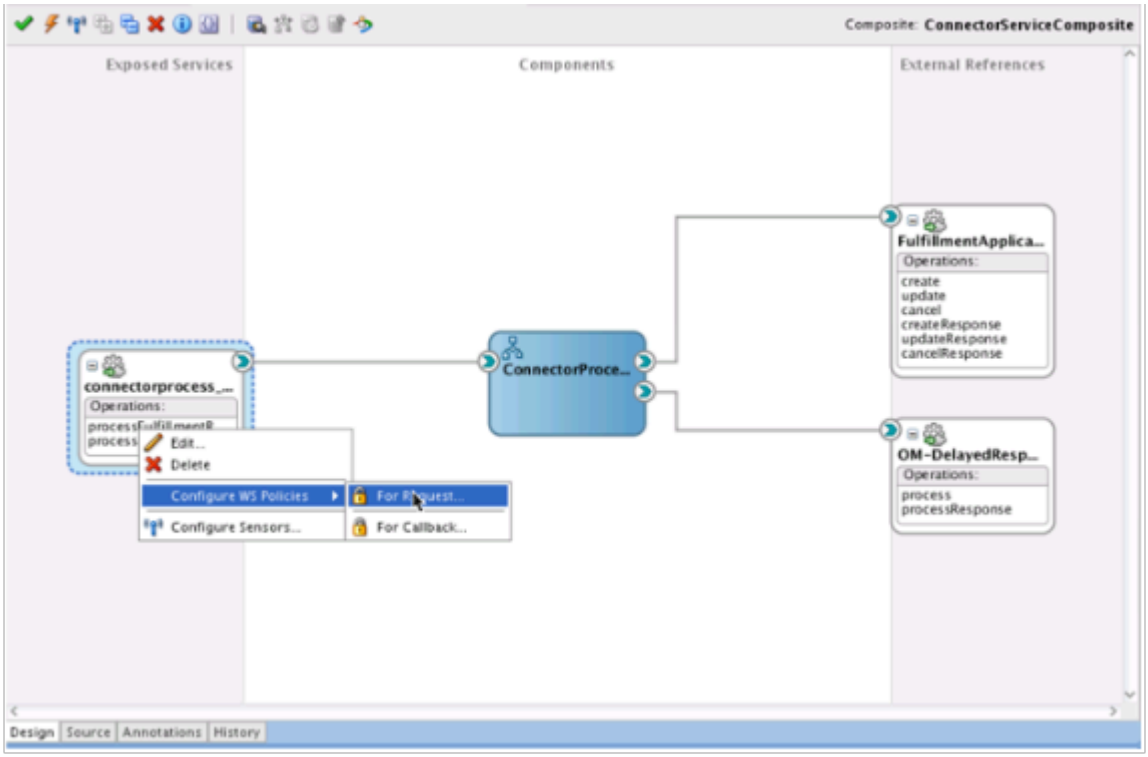
Oracle Applications use policies to secure your connector.

Policy	Description
oracle/wss11_saml_or_username_token_with_message_protection_service_policy	Encrypts and decrypts incoming and outgoing messages.
oracle/wss_username_token_over_ssl_client_policy	Uses SSL (Secure Sockets Layer) to secure communication.

Use Oracle JDeveloper or Oracle Enterprise Manager to secure your connector. This example uses Oracle JDeveloper. For details about Oracle Enterprise Manager, see https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.

Secure your connector and deploy your project.

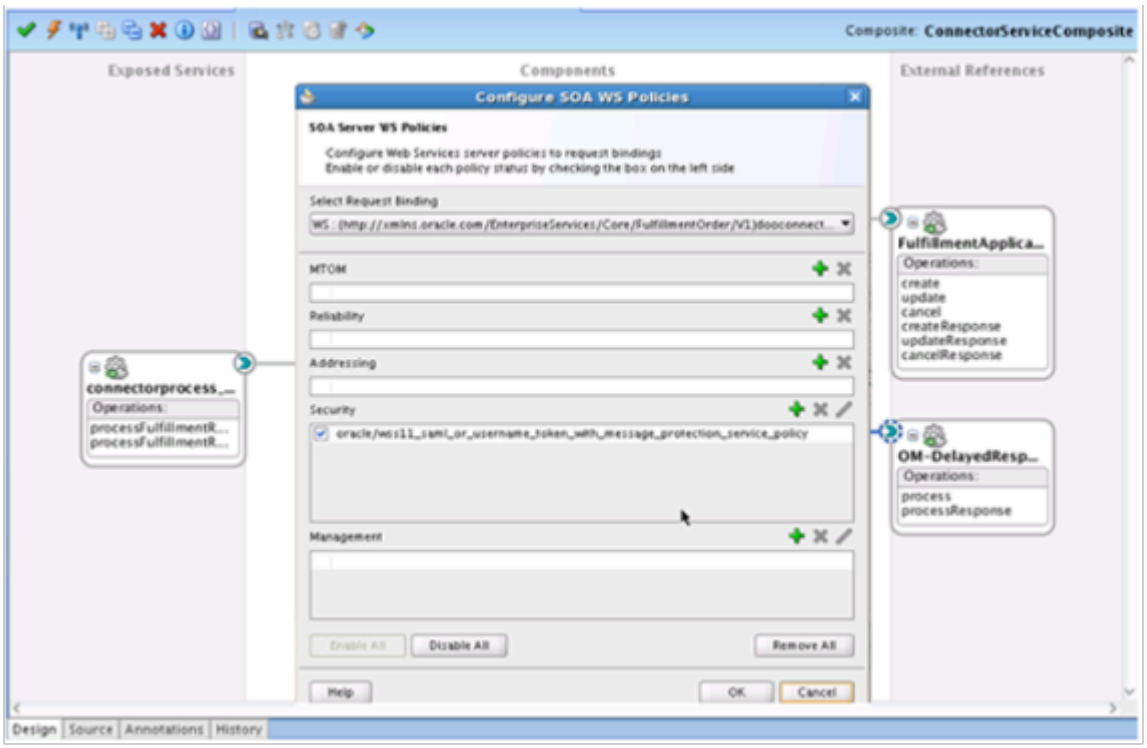
1. Specify the security policy for the Request service. In the Exposed Services pane, right-click the **end point** of ConnectorProcess, then click **Configure WS Policies > For Request**.



2. In the Configure SOA WS Policies dialog, in the Security area, click **Add**.

3. In the Select Server Security Policies dialog, select the value.

   `oracle/ws11_saml_or_username_token_with_message_protection_service_policy`



4. Click **OK**.
5. Specify the security policy for the callback service. In the Exposed Services pane, right-click the **end point** of ConnectorProcess, then click **Configure WS Policies > For Callback**.
6. In the Configure SOA WS Policies dialog, in the Security area, click **Add**.
7. In the Select Server Security Policies dialog, select the value.

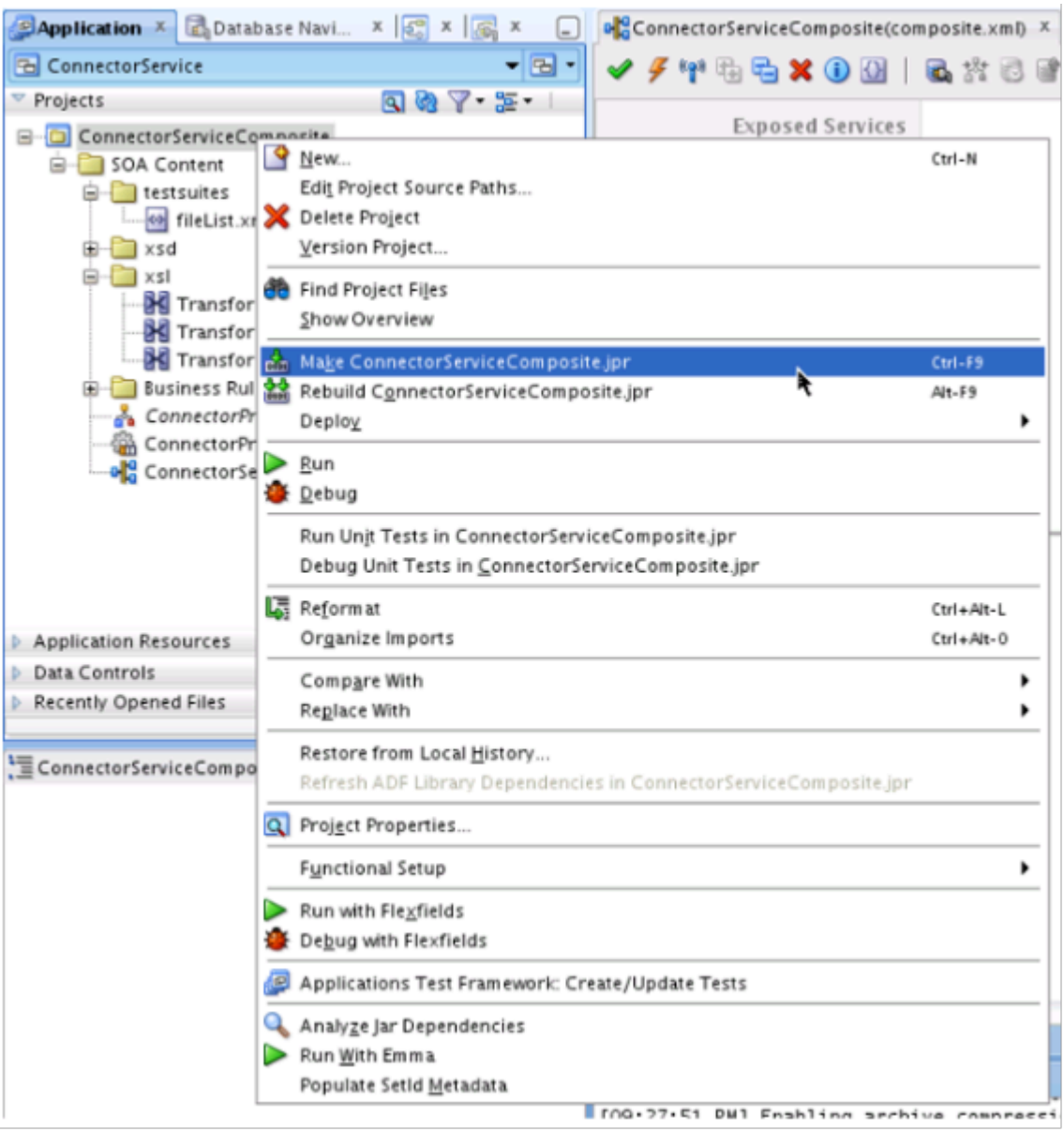
   `oracle/wss_username_token_over_ssl_client_policy`

8. Click **OK**.
9. In the Security area, click the **row** that you just added, then click **Edit**.
10. Set the values.

Attribute	Value
Override Value	FUSION_SCM_SOA_APPID-KEY Note o This value specifies the key for Credential Store Framework (CSF). o You must create this key in the application where you deploy the connector. o This key must use the same user name and password that you use to sign into Order Management.

11. Edit and assign the security policy for the connector that handles the delayed response.

    Repeat the above steps. Use the same security policies and specify the same Credential Store Framework key.
12. Create and deploy your project.



13. Create an object, then add it to the identity store on the server that will call the web service. For details, see *Integrate Order Management with Source Systems*.
14. Get the user credentials that the service provider needs when they call their web service. You can typically get these details from the service provider.
15. Send a request to your IT administrator to add the user credentials that you identified in step 4. Request to add them to the server that will call the web service. Use the CSF-KEY (Credential Store Framework) reference.

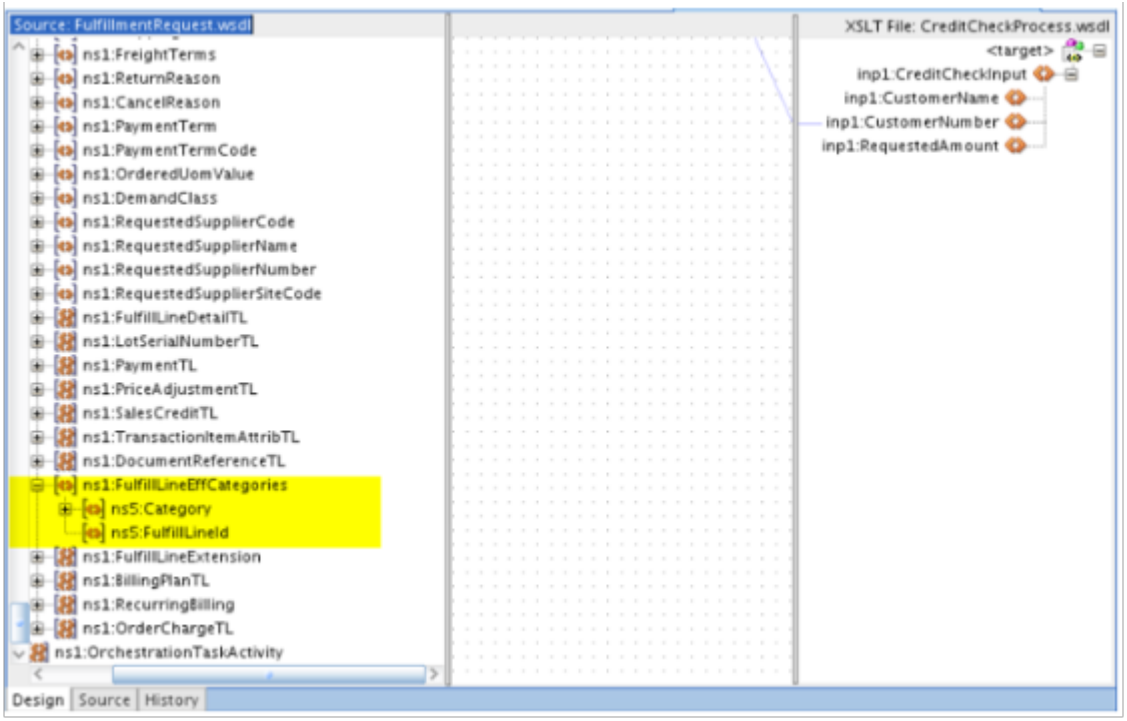
Communicate Extensible Flexfields to Your Fulfillment System

The payload includes extensible flexfields that you have set up, by default. If you have set up extensible flexfields, then you must set up the WSDL so it can communicate them to your fulfillment system.

Communicate extensible flexfields to your fulfillment system.

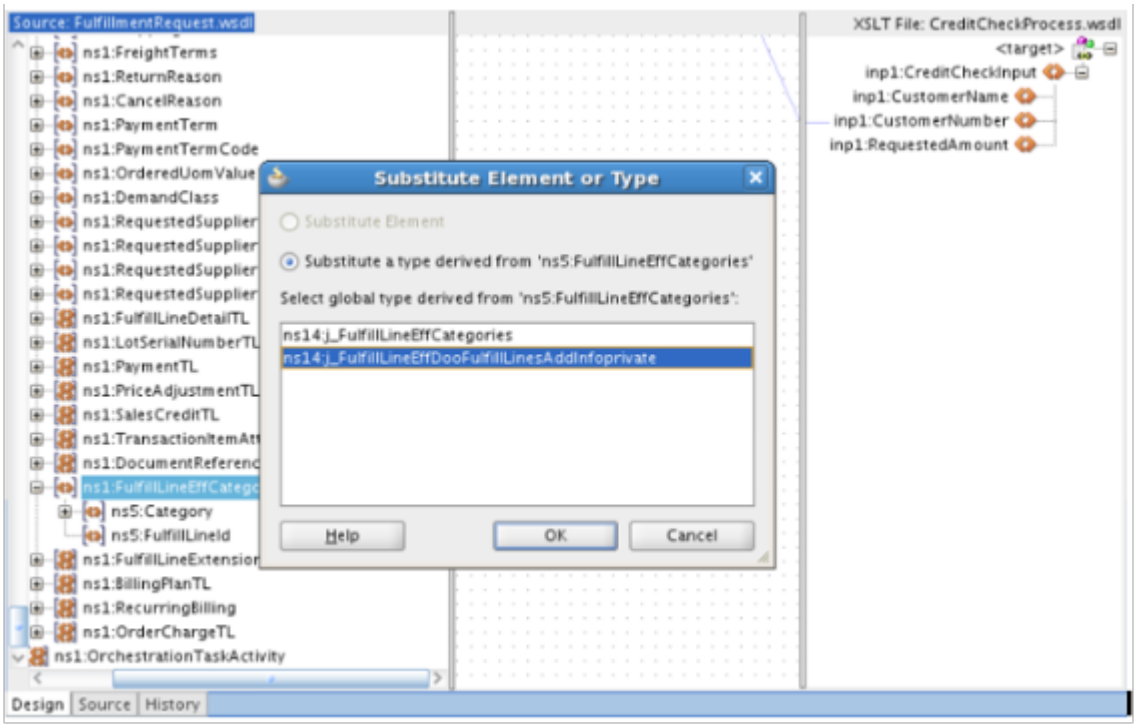
1. In the transformation activity, navigate to the appropriate EffCategories node, such as FulfillLineEffCategories.
2. Expand the node.

   Notice that only two attributes are available in the node.



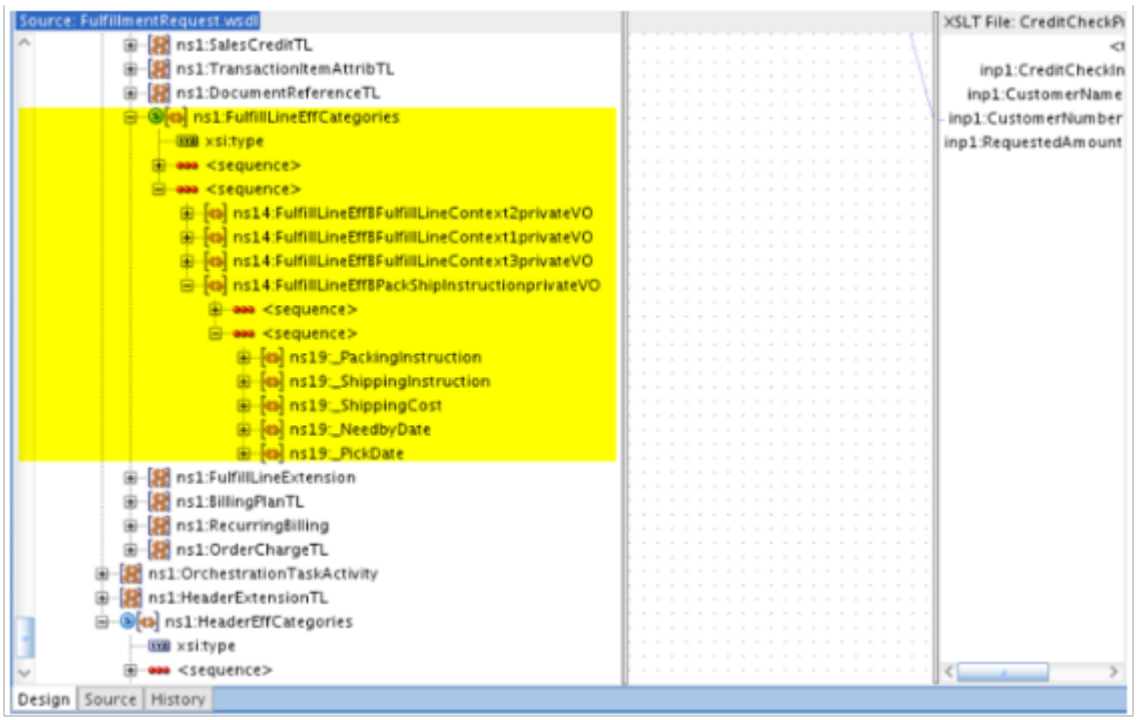
3. In the `Source: FulfillmentRequest.wsdl` pane, in the navigation tree, right-click `ns1:FulfillLineEFFCategories`, then click **Substitute Element or Type**.

4. In the Substitute Element or Type dialog, click **j_FulfillLineEFFDooFulfillLineAddInfoPrivate > OK**.



5. Expand **`ns1:FulfillLineEFFCategories`**.

   The navigation tree displays the hierarchy that includes the contexts and extensible flexfields that you can map to the target XSD.



6. Make sure each attribute that you must map exists.

   For example:

```
<xsl:if test="/ns1:headerTL/ns1:FulfillLineTLVO1/ns1:FulfillLineEffCategories/
ns10:FulfillLineEffBPackShipInstructionprivateVO/ns21:shipDate">
 <ns21:shipDate>
 <xsl:if test="/ns1:headerTL/ns1:FulfillLineTLVO1/ns1:FulfillLineEffCategories/
ns10:FulfillLineEffBPackShipInstructionprivateVO/ns21:shipDate/@xsi:nil">
 <xsl:attribute name="xsi:nil">
 <xsl:value-of select="/ns1:headerTL/ns1:FulfillLineTLVO1/ns1:FulfillLineEffCategories/
ns10:FulfillLineEffBPackShipInstructionprivateVO/ns21:shipDate/@xsi:nil"/>
 </xsl:attribute>
 </xsl:if>
 <xsl:value-of select="/ns1:headerTL/ns1:FulfillLineTLVO1/ns1:FulfillLineEffCategories/
ns10:FulfillLineEffBPackShipInstructionprivateVO/ns21:shipDate"/>
 </ns21:shipDate>
</xsl:if>
```

7. Map each attribute.

Map Document References

Order Management uses a typical entity to describe the relationship that exists between a transaction in a system that resides outside of Order Management and a fulfillment line. Here are some document types that you can use to map the data elements between this entity and your fulfillment system.

Attribute	Description	Example
DocRefType	Abbreviation that identifies the type of business document. Here are the these types that you can use. • **EXT_FULFILLMENT_SALES_ORDER**. References the sales order or document in your fulfillment system. • **ORIGINAL_SALES_ORDER**. References the source order from a source system or sales order from Order Management that contains the item that a return is returning. • **EBS_ORDER**. References a sales order created in Oracle E-Business Suite `(EBS)` where Oracle E-Business Suite is the fulfillment system. • **DROPSHIP_REQ_REFERENCE**. References a requisition for an order line in a drop shipment in a purchasing application. • **DROPSHIP_PO_REFERENCE**. References a purchase order for an order line in a drop shipment in a purchasing application.	DROPSHIP_PO_REFERENCE
DocId	Value that uniquely identifies the document. Order Management creates this value.	Purchase Order Identifier in the purchasing system
DocUserKey	Business document number that an end-user can understand and recognize.	Purchase Order Number

Attribute	Description	Example
DocAltUserKey	Details that accompany DocUserKey, such as a document revision number.	Change Sequence Number
DocLineId	Value that uniquely identifies the document line. Order Management creates this value.	Purchase Order Line Identifier in the purchasing system
DocLineUserKey	Line number that an end-user can understand and recognize.	Purchase Order Line Number
DocSublineId	Value that uniquely identifies the document subline. Order Management creates this value.	Purchase Order Schedule Identifier in the purchasing system
DocSubLineUserKey	Number for the subline that an end-user can understand and recognize.	Purchase Order Schedule Number

Related Topics

- How Order-to-Cash Works with Order Capture Systems
- How Data Flows Through Order Management
- Integrate Order Management with Source Systems
- Route Requests from Order Management to Fulfillment Systems

Manage Connector Details Between Order Management and Your Fulfillment System

Use a web service to allow Order Management to communicate with your fulfillment system.

The connector in this topic sends each message from Order Management to your fulfillment system, then sends the response message from your fulfillment system to Order Management.

1. Sign into *Oracle Enterprise Manager*.
2. In the navigation tree, expand **Farm_fusion_domain > SOA > soa-infra (soa_server1) > default**, then click **ConnectorServiceComposite**.
3. In the ConnectorServiceComposite area, click **Service Endpoint**.
4. In the Service Endpoint and WSDL dialog, copy the value of the Endpoint URI and the WSDL to your clipboard.

   Here are some examples.

Attribute	Example Value
Endpoint URI	`http:// server:port /soa-infra/services/default/ ConnectorServiceComposite/connectorprocess_client_ep`
WSDL	`https:// server:port soa-infra/services/default/ DooTaskExternalInterfaceVirtualPartnersComposite/ fulfillmentrequest_client_ep?WSDL`

5. Go to the Setup and Maintenance work area, then go to the task.

- Offering: Order Management
- Functional Area: Orders
- Task: Manage External Interface Web Service Details

6. On the Manage Connector Details page, click **Actions > Add Row**, then set the values.

Attribute	Description
Target System	Choose a target system.
Connector Name	Enter text that describes the connection you're making.
Connector URL	Enter the URL that locates the web service. For example: `http://server:port/soa-infra/services/default/ConnectorServiceComposite/connectorprocess_client_ep` where - Replace ***`server:port`*** with the name of the server that hosts your web service and the port number that the web service uses to communicate. Note - Your fulfillment system must host this service. - The service must be a synchronous, SOAP web service. - You must not reference an Oracle internal web service. - To get details about the services that you can use, see *SOAP Web Services for Common Features* and *SOAP Web Services for Oracle SCM Cloud*.
User Name and Password	Enter the user name and password that the service requires. As an option, use CSF-KEY. This key references the user credential that Order Management uses to interact with the external web service. There's no requirement to match the actual name of the connector, so you can provide a short name. You can then use this short name in a routing rule. The value CSF-KEY applies to all services that your fulfillment system provides.
Keystore Recipient Alias	Note - Set up your server so it advertises the security certificate in the WSDL. - Set up each server that uses a web service to call Order Management. - The Oracle WebLogic Server advertises the security certificates, by default. If your servers support this advertisement, then enable it on your servers. If you set up your server to advertise the security certificate, then use the keystore recipient alias. Do these steps. a. Ask the service provider for the security certificate.

Attribute	Description
	b. Make sure an IT administrator imports the security certificate for the target server into the calling server and provides the keystore recipient alias. c. Add the alias to the service entry that you created when you specified the user credential. d. Add the alias to the Keystore Recipient Alias attribute on the Manage Connector Details page. This key applies to all services that your target system provides. If you find that these options don't work, then set up the servers to use the Oracle security certificate, and then import the certificate into your servers. The calling server doesn't require you to set up this security certificate.
Response Processing Option	Specify how to proceed when an error occurs. o **Reject All Lines on First Error**. Reject all fulfillment lines as soon as the connector encounters the first error. This setting stops processing immediately so you can fix the first error. If subsequent fulfillment lines contain errors, then you must run fulfillment again, correct the error, and repeat until you correct all errors. o **Reject All Lines When Error on at Least One Line Occurs**. Process all fulfillment lines. Add an error status to any fulfillment line that contains an error. When processing finishes, if any fulfillment line contains an error, then reject all fulfillment lines. Use this setting to examine all lines that contain errors and correct them without having to run fulfillment for each error. o **Reject Groups With Lines That Contain Errors**. Reject the entire group of fulfillment lines even if only one line in the group contains an error. Your fulfillment system can send fulfillment lines in a group. For example, it can send all lines in a shipment as a group or all lines in a sales order as a group. Use this setting to manage these groups.
Invocation Mode	Specify how Order Management should call the connector when the orchestration process requires an interface to your fulfillment system. o **Business Event.** Use a business event so Order Management can interact with the fulfillment system. If you use Integration Cloud Service to integrate Order Management with the fulfillment system, then you must choose Business Event. For details, see *Use Integration Cloud Service with Order Management*. o **Synchronous Service.** Make a synchronous call to the web service. Requires Order Management to wait for the response from the web service before it continues processing. Use synchronous when Order Management depends on the response. For example, use synchronous when calling credit check because Order Management must wait for credit check to finish, then reply with the Credit Check Succeeded status before it can send the sales order to order fulfillment. o **Asynchronous Service.** Make an asynchronous call to the web service. Allows Order Management to continue other processing while it waits for the response from the web service. Use asynchronous when Order Management doesn't depend on the response. An asynchronous call is useful in an environment where a service, such as a loan processor, can take a long time to process a request. For example, scheduling an appointment to install an item that includes a computer network might cause a delay but it doesn't affect order processing. Note

Attribute	Description
	○ You can use Business Event only with Oracle Global Trade Management and Oracle Transportation Management ○ If you screen trade compliance during order submit, then you must use a business event to integrate with Oracle Global Trade Management. ○ Use Business Event to integrate with trade compliance only for compliance screening that occurs when the Order Entry Specialist submits the sales order, and not during order fulfillment.
Send Attachments	Choose Yes to enable Order Management to send attachments on sales orders.

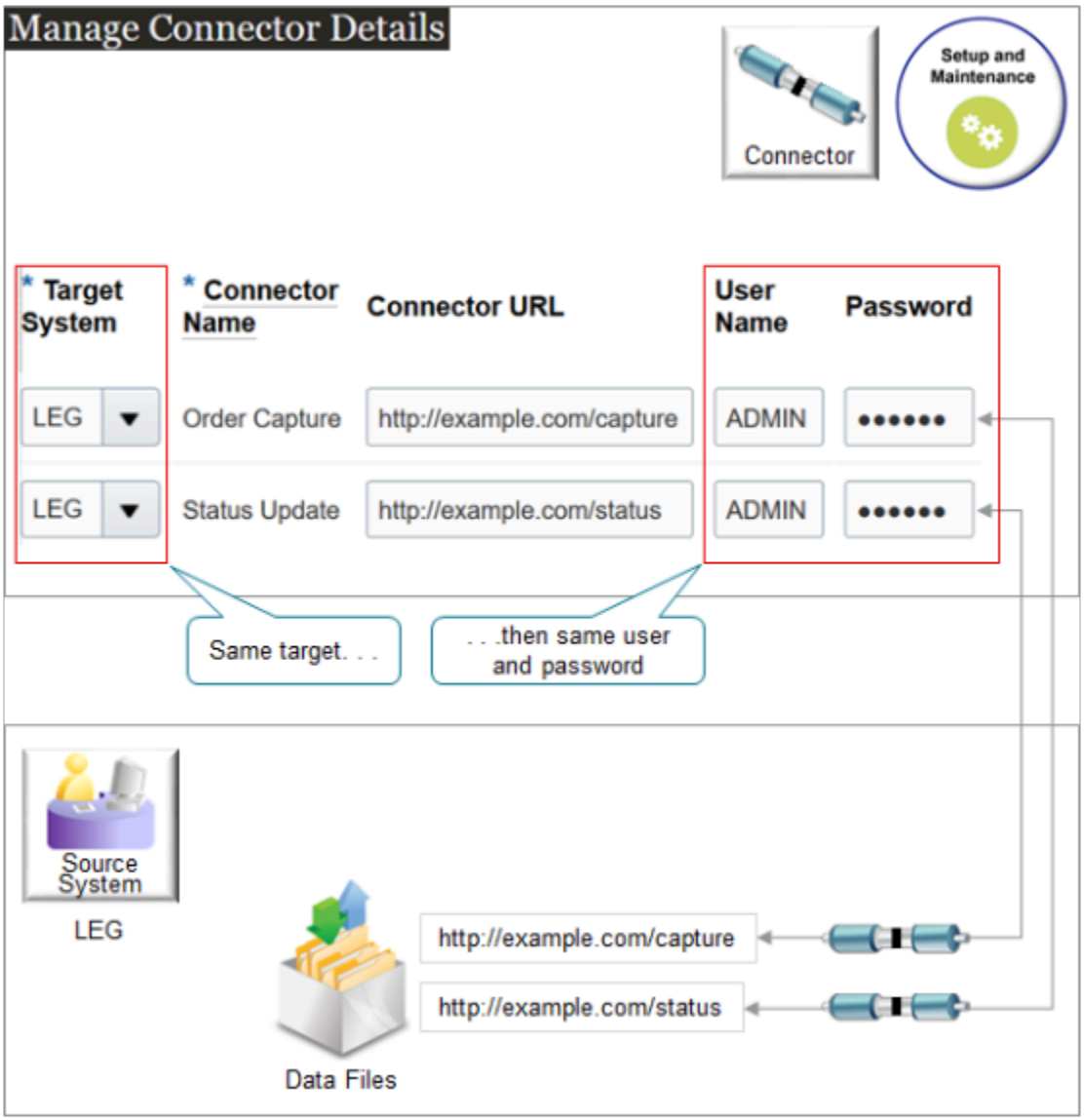
7. Repeat step 7 for each web service that you use with Order Management.
8. Create a routing rule that selects the web services.

   For details, see *Route Requests from Order Management to Fulfillment Systems*.

Create More Than One Connector for Each Target

You can create more than one connector for each target system.

Assume you use a target system named LEG, which is an abbreviation for the word legacy. You need to create one connector to LEG to communicate details about orders that you capture in your legacy system, and another connector to LEG to communicate status updates for these orders. Here's what your set up would look like.



If you create more than one connector for the same target system, then you can use the same user name and password on each of those connectors, or you can use a different user name and password. In this example, assume you want to use the same user name and password for the Order Capture connector and for the Status Update connector.

Use a Web Service

You can specify a connector in any web service that supports an asynchronous operation and that supports the oracle/wss_username_token_over_ssl_client_policy policy. For details, see:

- *Web Services That You Can Use to Integrate Order Management*
- *Guidelines for Using Web Services to Integrate Order Management*

Use this attribute in the DooTaskFulfillOrderResponseInterfaceComposite section of your Fulfillment Response Service payload or in your Order Fulfillment Response Service payload:

Attribute	Value
CallbackConnectorName	Use the same value that you would use in the Connector Name attribute on the Manage External Web Service Details page.

You can use CallbackConnectorName in the DooDecompReceiveOrderExternalComposite section of your payload with these services:

- Process Order Request
- Request Hold
- Release Hold
- Get Availability Check
- Check Availability
- Release Paused Tasks

Related Topics

- How Order-to-Cash Works with Order Capture Systems
- How Data Flows Through Order Management
- Route Requests from Order Management to Fulfillment Systems
- Integrate Order Management with Source Systems
- Guidelines for Using Web Services to Integrate Order Management

Route Requests from Order Management to Fulfillment Systems

Use a routing rule to route a fulfillment request to your fulfillment system.

Specify each rule so it selects the fulfillment system connector according to sales order, fulfillment line, or orchestration process attribute. At runtime, the rule calls the connector that translates the payload into a structure that your fulfillment system can understand.

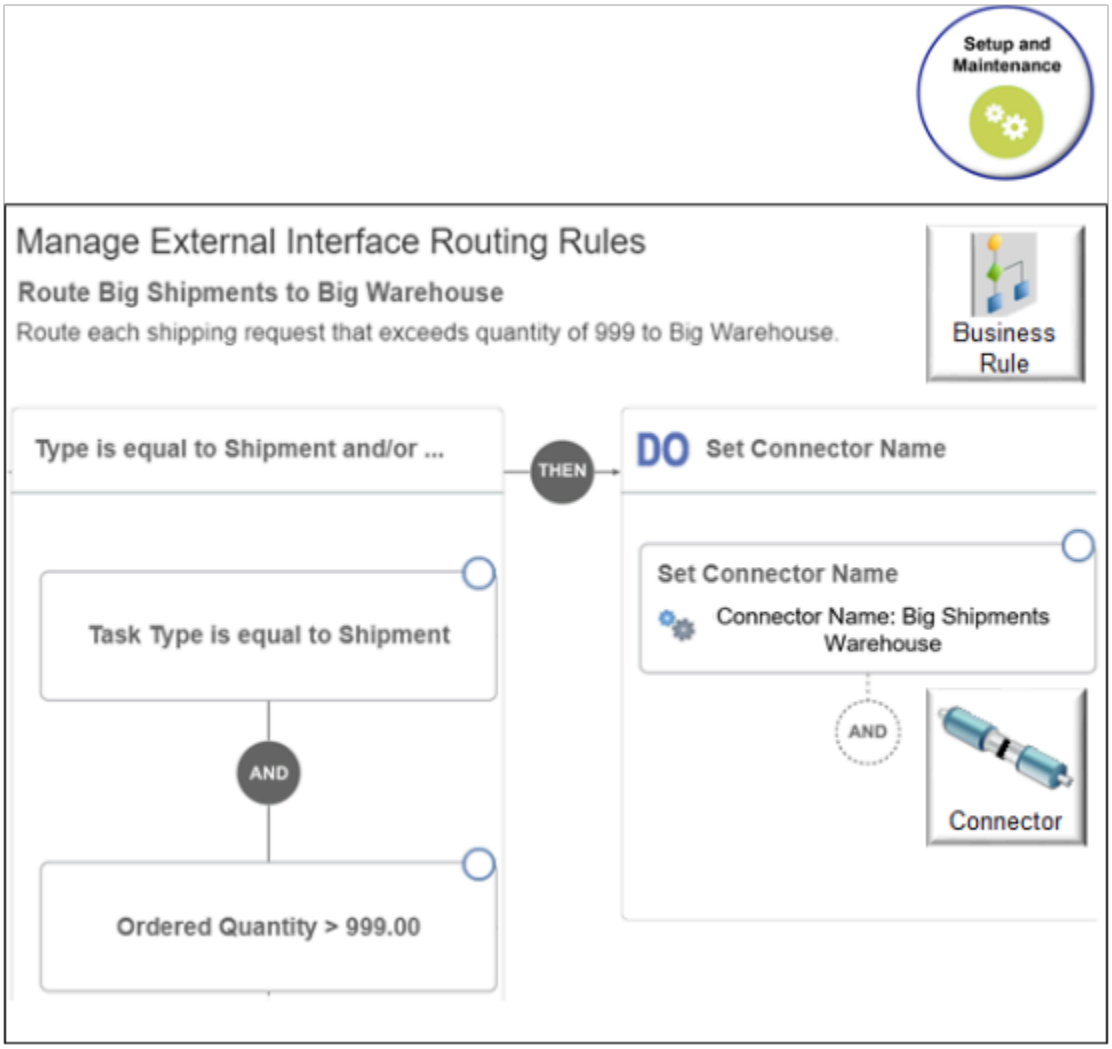
Here are some examples that describe ways you might use a routing rule.

Example	Description
Route sales order according to the value of an orchestration process attribute.	For example, route each sales order that's ready to ship to a fulfillment system according to task type. • If type code is Shipment, then route sales order to connector ABCShippingSystem.
Route sales order according to the value of a customer attribute.	For example, assume your company uses two invoicing systems, and that system ABC sends invoices to customer Computer Service and Rentals. • If product type is Goods, and if task type is Invoice, then route request to connector ABCInvoicingSystem.

Assume you must create a routing rule that implements a condition.

- If task type is Shipment, and if quantity is 1000 or more, then route shipment request to Big Warehouse.

Here's the rule you will create.



Summary of the Set Up

1. Set up connector.
2. Create the If statement.
3. Create the Then statement.
4. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Set Up Connector

Set the values.

Attribute	Value
Name	Big Shipments Warehouse
Description	Route each shipping request that exceeds a quantity of 999 to the Big Shipments warehouse.

For details, see *Connect Order Management to Your Fulfillment System*.

Create the If Statement

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Integration Routing Rules for Sales Orders
2. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Route Big Shipments to Big Warehouse
Description	Route each shipping request that exceeds quantity of 999 to Big Warehouse.

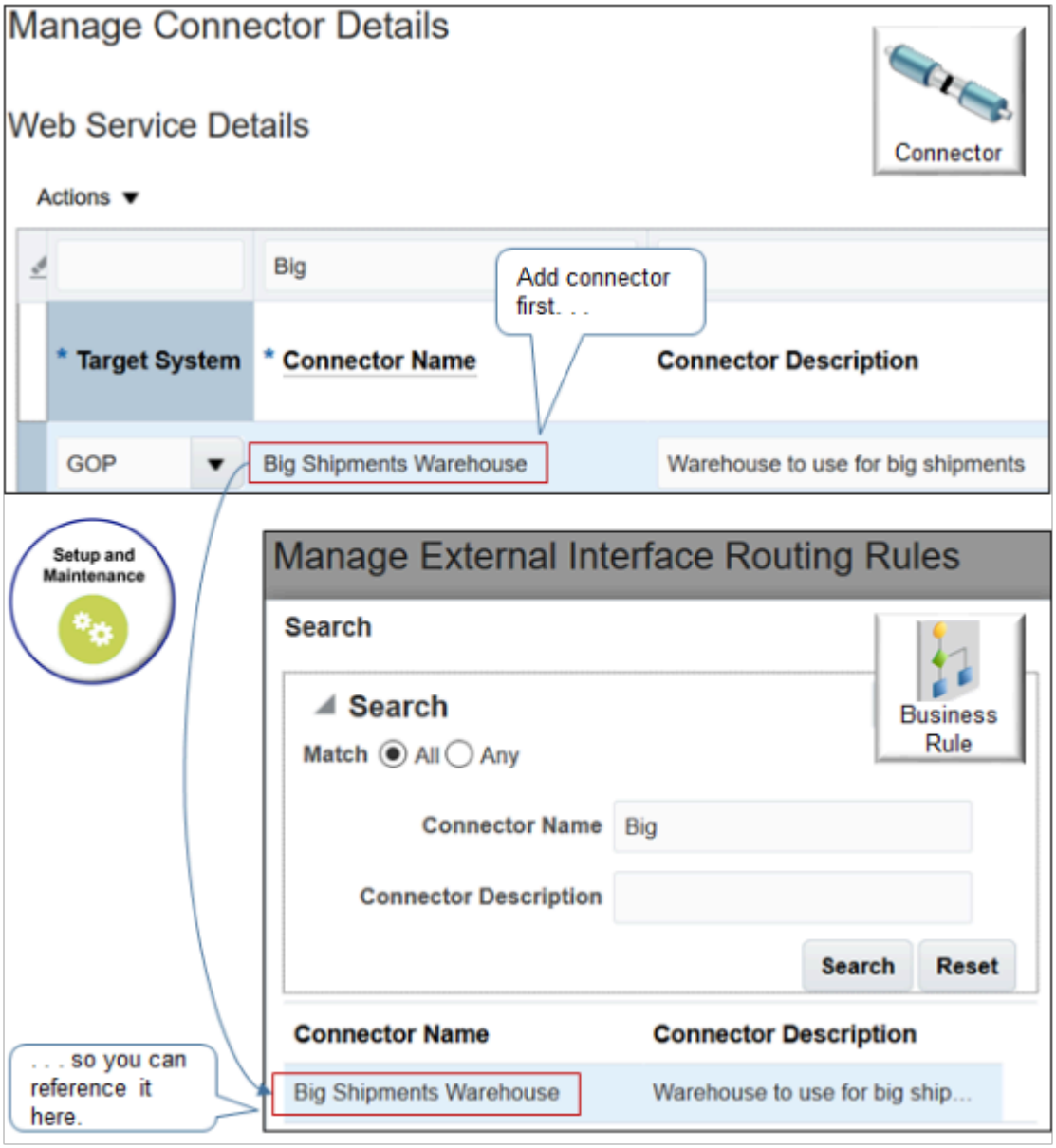
3. Add the If condition.
    - Click **New Condition**.
    - In the Create Condition dialog, enter `Task`, wait a moment, then click **Task Type (Order Header)**.
    - Click **Search**.
    - In the Search dialog, enter `Shipment`, then click **Search > OK > OK**.
4. Add the And condition.
    - Click **And**.
    - In the Create Condition dialog, enter `Quantity`, wait a moment, then click `Ordered Quantity (Order Fulfill Line)`.
    - Change = to >.
    - Enter `999`, then click **OK**.

Create the Then Statement

1. On the flowchart, click **Then > Do > New Action > Perform an Action**.
2. In the Create Action dialog, choose **Set Connector Name**, then click **Search**.
3. In the Search dialog, set the value, then click **Search > OK > OK**.

Attribute	Value
Connector Name	Big Shipments Warehouse

Note that the Manage Connector Details page defines the connectors that the Search dialog displays on the Manage External Interface Routing Rules page.



4. Click **Save and Close**.
5. On the Manage External Interface Routing Rules page, click your **rule**.
6. In the dialog that displays, add a check mark to **Activate Rule**, then click **Save and Close > Publish**.

Test Your Set Up

1. Navigate to the Order Management work area, create a sales order, add an order line with a Quantity of 1000, then click **Submit**.
2. Sign into Oracle Enterprise Manager, navigate to Flow Trace, Instance of EILMainProcess.
   For details about Oracle Enterprise Manager, see https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.
3. In the Audit Trail area, in the InvokeFulfillmentService area, examine the payload and verify that it includes the Create Fulfill Order service and the ServiceURI that you specified earlier.

Related Topics

- Use Visual Information Builder
- Manage Routing Rules

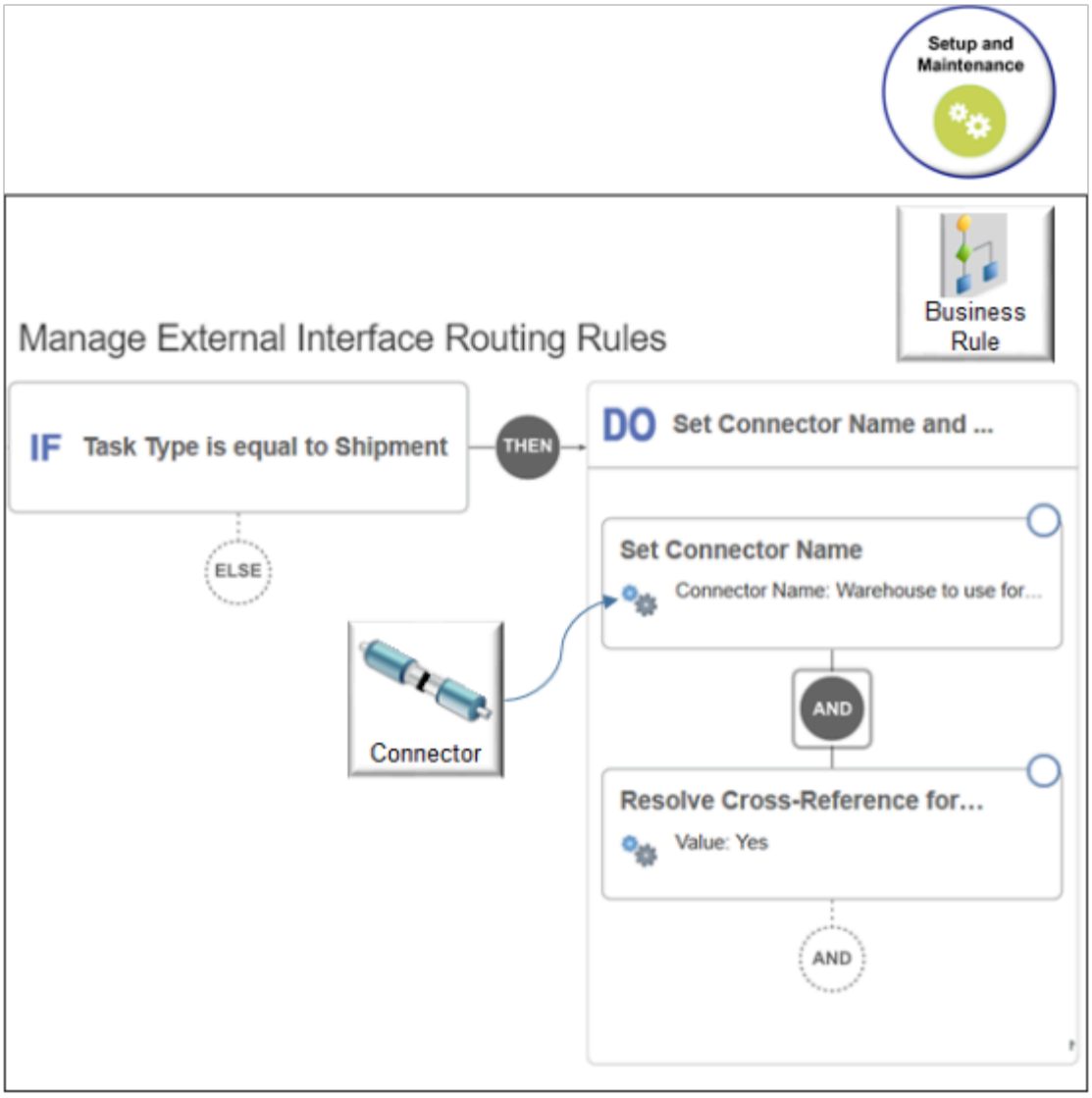
Route Requests from Order Management to Fulfillment Systems Without Cross-References

Create a routing rule that routes a request to a fulfillment system when Oracle Order Management can't find a cross-reference that identifies the fulfillment system.

Assume you must implement this logic.

- If the task type on the fulfillment line is Shipment, then route the shipment request to Big Warehouse.
- If Order Management can't find a cross-reference for Big Warehouse in an Oracle Application, then get the cross-reference from your fulfillment system.

You will create this rule.



Try it.

1. Create a routing rule that implements this logic:
    - If the task type is Shipment, and if the quantity is 1000 or more, then route the shipment request to Big Warehouse.

      For details, see *Route Requests from Order Management to Fulfillment Systems*.
2. On the Manage External Interface Routing Rules page, open your rule for editing, then, in the flowchart, click **And > Perform an Action**.
3. In the Create Action dialog, select **Resolve Cross-Reference for Customer**.
4. Select a value, then click **OK**. For this example, select **No**.

Value	Description
Yes	Use a cross-reference from an Oracle Application.

Value	Description
No	Use a cross-reference from the fulfillment system.

5. Click **Save and Close**, then publish your rule.

Actions That You Can Set When Routing Requests to Fulfillment Systems

Use actions in your routing rules to specify how to route each request to your fulfillment system.

For details about:

- Where you set actions in the routing rule, see *Manage Routing Rules*.
- How to specify actions in the payload that the Fulfillment Order service uses, see *Task Services*.

Here are the actions that you can use.

Action	Description	Value	Attribute to Use in Payload	Data Type to Use in Payload
Override Compensation Pattern	Override the compensation pattern that the orchestration process specifies. Use this action only for your own custom fulfillment task. Don't use it for a predefined fulfillment task because predefined tasks already have a compensation pattern.	• CANCEL_CREATE • UPDATE	COMPENSATION_PATTERN	String
Override Operation	Override an operation. Order Management uses operations, such as Create, Update, Process, and so on. Your fulfillment system might not recognize these operations. Use this action to specify an operation that your fulfillment system recognizes. Use this action only with task types that you create. Don't use it with predefined task types.	Text or alphanumeric value.	MODIFIED_OPERATION	String
Prepare Result	Don't use this action or the ResultKey and ResultValue parameters that it uses. They are for Oracle internal only.	Not applicable	Not applicable	Not applicable
Resolve Cross-Reference for Customer	Specify whether to use a cross-reference. If your	• Yes	RESOLVE_XREF_FOR_CUSTOMERINFO	String

Action	Description	Value	Attribute to Use in Payload	Data Type to Use in Payload
	fulfillment system uses the same attribute values that Order Management uses, then set this action to No because a cross-reference is useful only when these values are different. For details, see *Create Cross-References in Order Management*.	• No		
Set Acknowledgement Timeout	Amount of time to wait before exiting out of an implicit wait during an interaction with your fulfillment system or with an Oracle Application. Don't use this action with tasks that involve inventory management, shipping, or accounts receivable.	Use a numeric value for days, hours, minutes, or seconds. For example, 10 minutes.	ACK_TIMEOUT_PERIOD	Number
Set Connector Name	Connector name that the web service uses to communicate details to your fulfillment system. This name helps Order Management route the message to the web service URL that this connector references. For details about how to set up a connector so you can specify it in this action, see *Manage Connector Details Between Order Management and Your Fulfillment System*.	You can use these predefined connectors: • Connector to Oracle Receivables. • Connector to Oracle Inventory Management. • Connector to Oracle Receiving. • Connector to Oracle Shipping.	SERVICE_NAME	String
Set Interaction Interface Type	Format to use when Order Management interacts with your fulfillment system.	• Service Data Object • Enterprise Business Message	InterfaceType	String
Set Maximum Lines to Aggregate and Send	For details, see *Aggregate Requests That Order Management Sends to Your Fulfillment System*.	Not applicable	AGGREGATOR_MAX_FLINES	Number
Set Maximum Time to Wait Before Enabling Abort	Order Management sends a request to your fulfillment system or to some other Oracle Application, and then waits to receive a reply. The reply typically arrives within a few seconds.	Use a numeric value to express time of day, such as days, hours, minutes, or seconds.	ABORT_ENABLE_PERIOD	Number

Action	Description	Value	Attribute to Use in Payload	Data Type to Use in Payload
	Starting with Update 21C, use the Set Maximum Time to Wait Before Enabling Abort parameter instead of Set Maximum Time to Wait Before Allowing Cancel. Use this action to specify how long to wait for the reply and avoid a situation where Order Management goes into a perpetual wait state because of a problem that happens during the transaction, such as the network going down. Here are the wait times that Set Acknowledgement Timeout sets, by default. • **30 minutes**. If the reply doesn't arrive within 30 minutes, then Order Management enables the Cancel Current Task action in the orchestration process that processes the fulfillment line. If the Order Entry Specialist uses the Cancel Current Task action, then Order Management places the fulfillment line in error recovery, and the Order Entry Specialist can then end the task. • **90 minutes**. If the reply doesn't arrive within 90 minutes, then Order Management places the fulfillment line in error recovery. Don't use this action with tasks that involve inventory management, shipping, or accounts receivable.			
Set Maximum Time to Wait Before Aggregation	Starting with update 21C, use Set Maximum Time to Wait Before Aggregation instead of Set Maximum Time To Wait Before Sending. For details, see *Aggregate Requests That*	Not applicable	AGGREGATOR_MAX_TIMEOUT	Number

Action	Description	Value	Attribute to Use in Payload	Data Type to Use in Payload
	Order Management Sends to Your Fulfillment System.			

Note

- Don't use the Set a Value operation in the DO clause.
- You must use the Connector Name action and the Interface Type action. All other actions are optional.
- You can use any action when you integrate with an application that isn't an Oracle application. You can also use the Set Acknowledgement Timeout action or the Set Maximum Time to Wait Before Allowing Cancel action when you integrate with some other Oracle application.

Related Topics

- Use Visual Information Builder
- Manage Routing Rules
- Task Services
- Actions That You Can Set When Routing Requests to Fulfillment Systems
- Manage Connector Details Between Order Management and Your Fulfillment System

Aggregate Requests That Order Management Sends to Your Fulfillment System

Aggregate requests to your fulfillment system to help minimize problems that might happen with the timing of requests.

Use this topic only if you implemented Order Management on Update 23C or earlier. If you implement starting with Update 23D or later, then see the Aggregate Fulfillment Lines subtopic in *Set Up Features, Manage Change, and More for Drop Ship*.

Use these actions on your routing rule.

Action	Description
Set Maximum Lines to Aggregate	Number of pending requests that must aggregate before calling the fulfillment system.
Set Maximum Time to Wait Before Aggregation	Amount of time to wait before calling the fulfillment system. This time starts when the aggregator receives the first request. Starting with update 21C, use Set Maximum Time to Wait Before Aggregation instead of Set Maximum Time To Wait Before Sending.

An aggregator collects requests, and then sends them in a single request when a time limit expires or when the aggregator has aggregated a number of fulfillment lines for the sales order. If the sales order includes more than one fulfillment line, and if these lines finish the task before the timeout happens, then it sends all requests when the task finishes.

- The aggregator can aggregate only one time for each fulfillment system for each sales order. If Order Management receives more fulfillment lines for the sales order after the aggregator sends a request, then it sends each of these lines individually.

- The default timeout is five minutes.
- You can use the aggregator only with the Fulfill Order task type or with a task type that you create.
- If you specify Set Maximum Lines to Aggregate and also specify Set Maximum Time to Wait Before Aggregation, then Order Management uses the first action that meets the conditions.
  For example, assume you set these values.

Action	Value
Set Maximum Lines to Aggregate	50
Set Maximum Time to Wait Before Aggregation	10

  At run time, assume 50 lines aggregate before 10 minutes elapse. Order Management will send the lines as soon as 50 lines aggregate. If only 40 lines aggregate after 10 minutes, then Order Management won't send any lines until 10 minutes elapse.

Note

For This Behavior	Do This Set Up
Consider only wait time. Ignore maximum fulfillment lines.	Set the Set Maximum Lines to Aggregate action to 0. This setup is equivalent to the default behavior when you don't specify Set Maximum Lines to Aggregate.
Ignore wait time. Send only one line for each request.	Set the Set Maximum Lines to Aggregate action to 1. This setup is equivalent to the default behavior when you don't specify either action.

For example, assume:

- You set the Set Maximum Lines to Aggregate attribute to 10.
- You set the Set Maximum Time to Wait Before Aggregation attribute to 5.
- An Order Entry Specialist creates a sales order and adds 16 order lines.
- Three minutes elapse between the time the aggregator receives the first order line on the template task and the tenth order line.

Set Maximum Lines to Aggregate equals 10 and Order Management will send 10 lines to the fulfillment system in the first request as soon as the aggregator receives the tenth order line.

Next, assume five minutes elapse by the time the aggregator receives the eleventh order line. During this time, the aggregator receives another three order lines. The value of Set Maximum Time to Wait Before Aggregation equals 5, so Order Management will send these four order lines in the second request to the fulfillment system.

For another example, assume:

- You set the Set Maximum Lines to Aggregate attribute to 0.

- You set the Set Maximum Time to Wait Before Aggregation attribute to 5.
- An Order Entry Specialist creates a sales order and adds 16 order lines.
- Five minutes elapse by the time the aggregator receives the first order line. During this time, the aggregator receives another eleven order lines.

The value of Set Maximum Time to Wait Before Aggregation is 5, so Order Management will send these 12 lines in the first request to the fulfillment system.

Related Topics

- Use Visual Information Builder
- Manage Routing Rules
- Task Services
- Actions That You Can Set When Routing Requests to Fulfillment Systems

Business Events

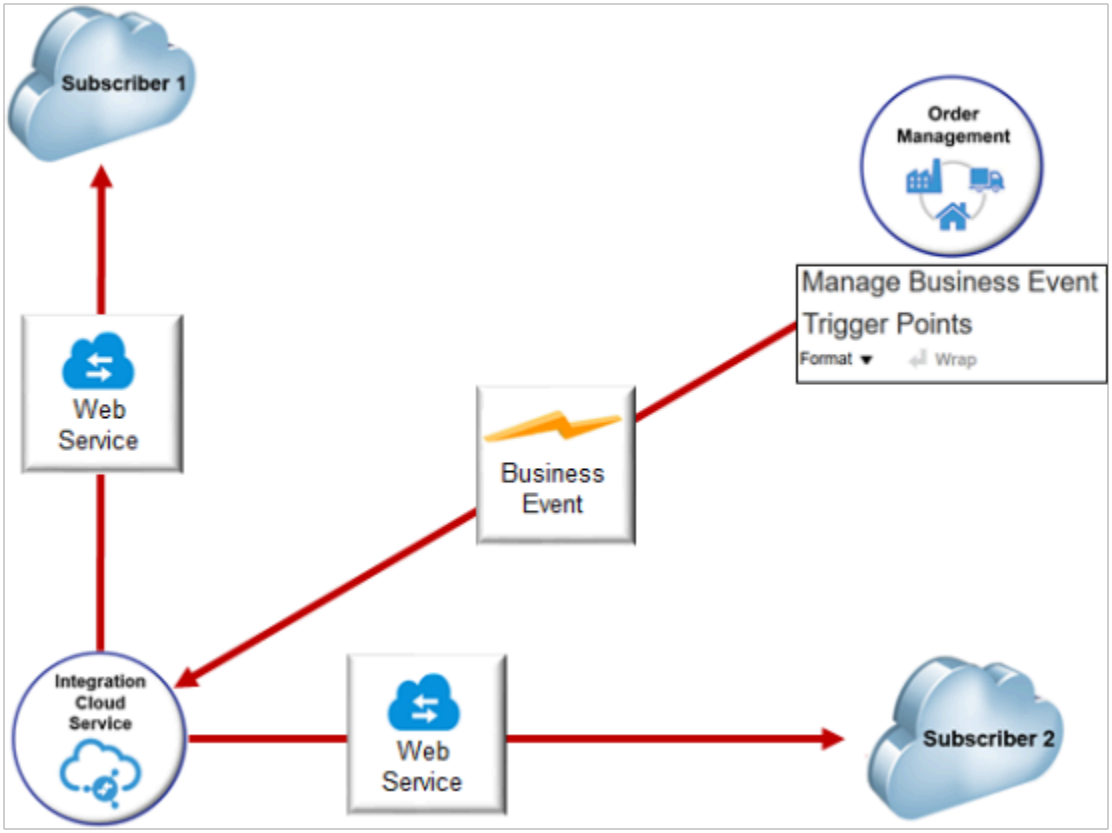
Overview of Using Business Events with Order Management

Use business events to integrate Oracle Order Management with some other Oracle Application or with an application that resides outside of Oracle Applications.

Use the Oracle ERP Cloud adapter to achieve results:

- Send a notification to each subscriber when a condition happens.
- Use the Manage Business Event Trigger Points page to set up the criteria that uses the business event.
- Use a different integration for each subscriber system. The subscriber can use a single web service that listens for the conditions through the business event that you specify.
- Allow each subscriber to enrich data in the integration.

You can use this flow:



Note

- Use this flow to deploy a single web service on each of your subscriber systems.

  The subscriber can receive the notification for each condition that triggers the business event.

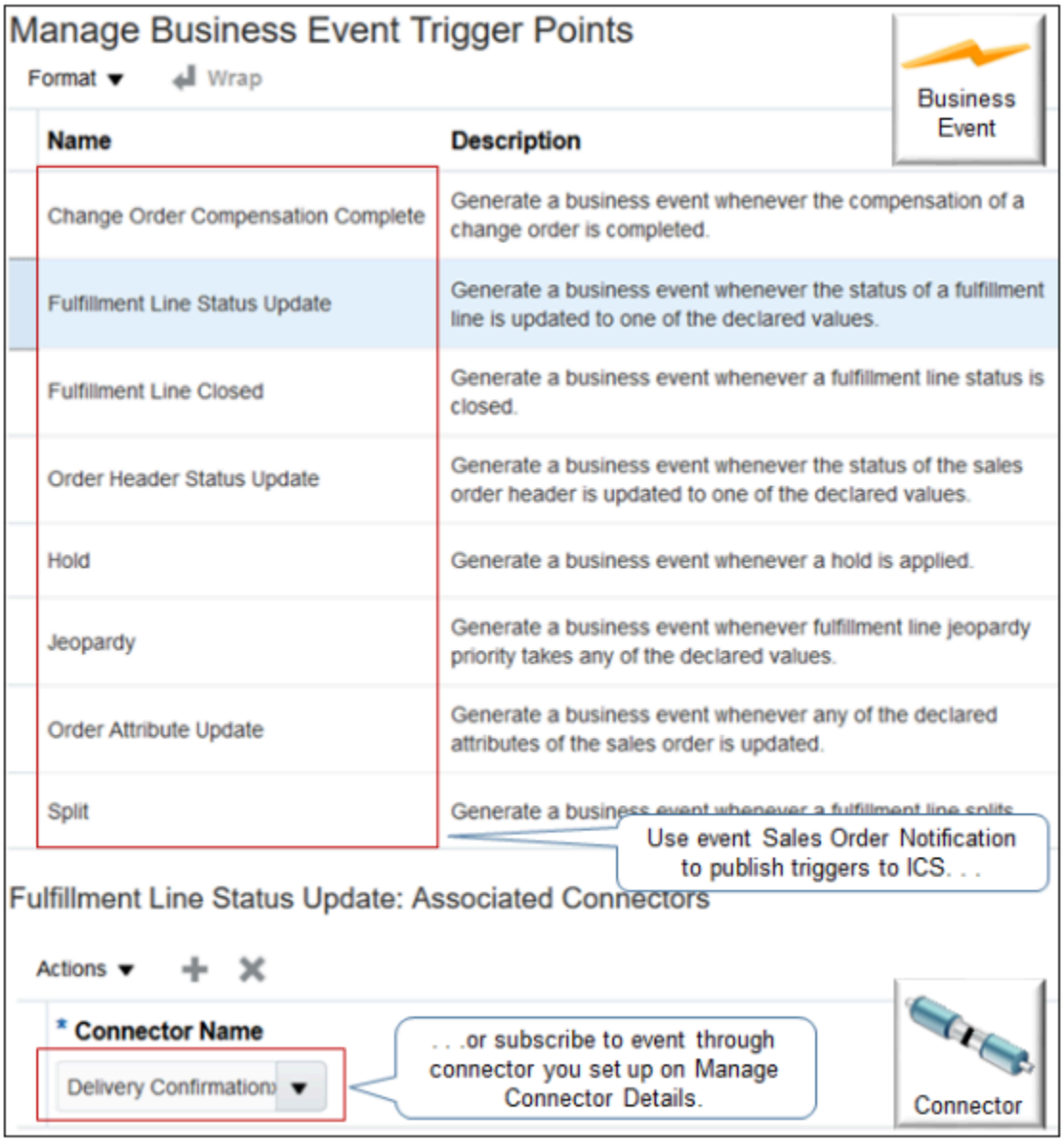
  You use the Subscribe to ICS integration to subscribe each channel system to the business event.
- Order Management sends one event notification to Integration Cloud Service, and then Integration Cloud Service broadcasts it to each of the subscribers that you integrate.
- If you set up the web service that you deploy on these systems to subscribe to the business event, and if they share the same underlying schema, then you use a similar mapping between the business event's output and each web service.

If you use the Sales Order Notification business event, then Order Management can send a notification each time one of these conditions happens:

- Update order header status.
- Update fulfillment line status.
- Split fulfillment line.
- Close fulfillment line.
- Apply hold.
- Change the jeopardy priority.
- Update a predefined attribute on a sales order.
- Finish changing a fulfillment plan.

Manage Business Event Trigger Points

To get started, go to the Setup and Maintenance work area, then use the Manage Business Event Trigger Points page to set up the trigger points that determine the notifications that you want to send to subscribers.



Note

- Use the Sales Order Notification event when you set up the end point in Integration Cloud Service. You use this event to publish trigger points to Integration Cloud Service.
- As an alternative to using Integration Cloud Service, you can add a connector that you set up on the Manage Connector Details page in the Setup and Maintenance work area. The trigger publishes the event to the connector.

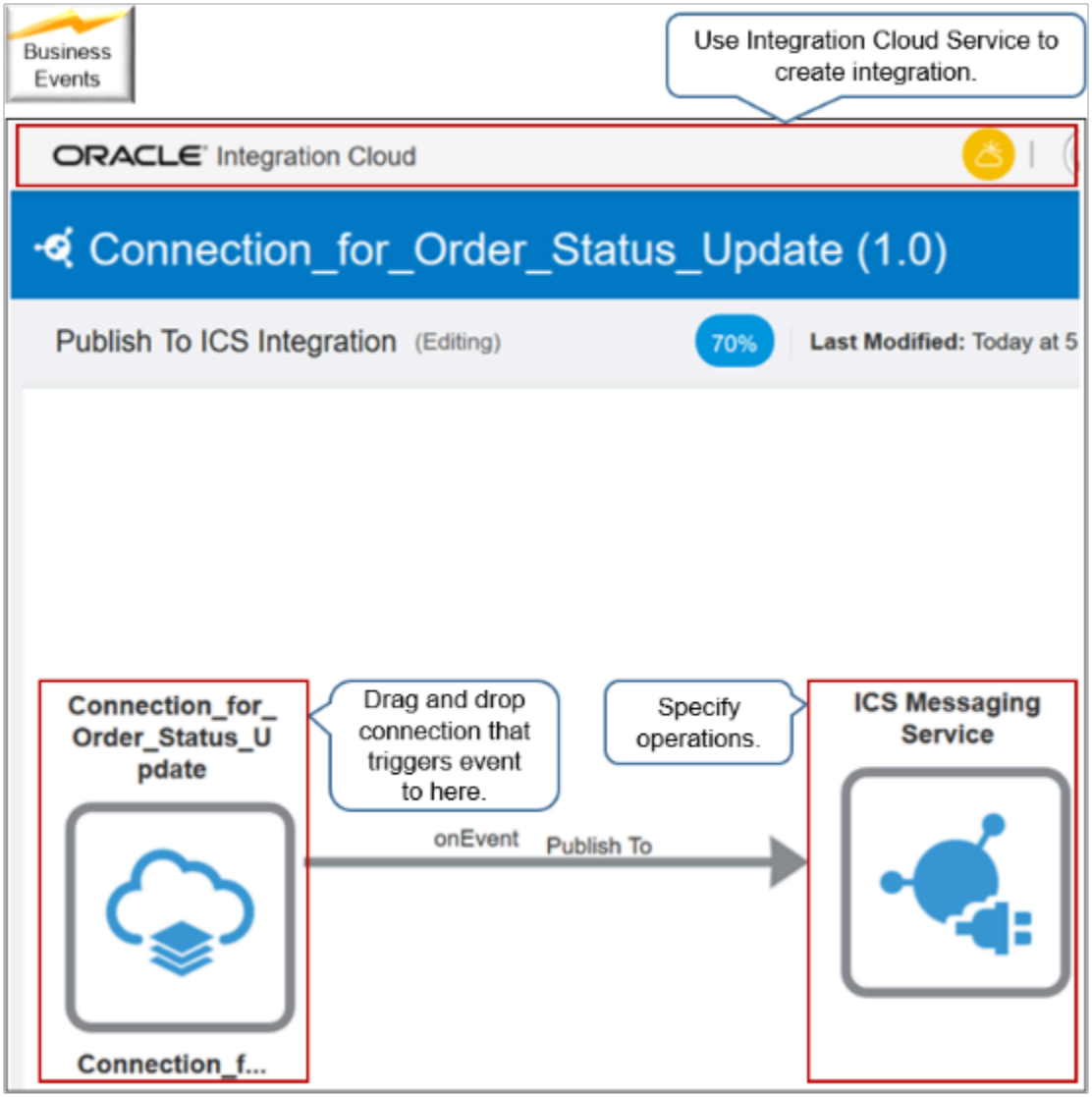
  For example, if you add the Delivery Confirmation connector to the Fulfillment Line Status Update trigger point, then Order Management will send the business event to Delivery Confirmation every time it updates the status on the fulfillment line.

The connector that you create in Integration Cloud Service is entirely separate from the connector that you create on Manage Connector Details. These connectors aren't related to one another in any way.

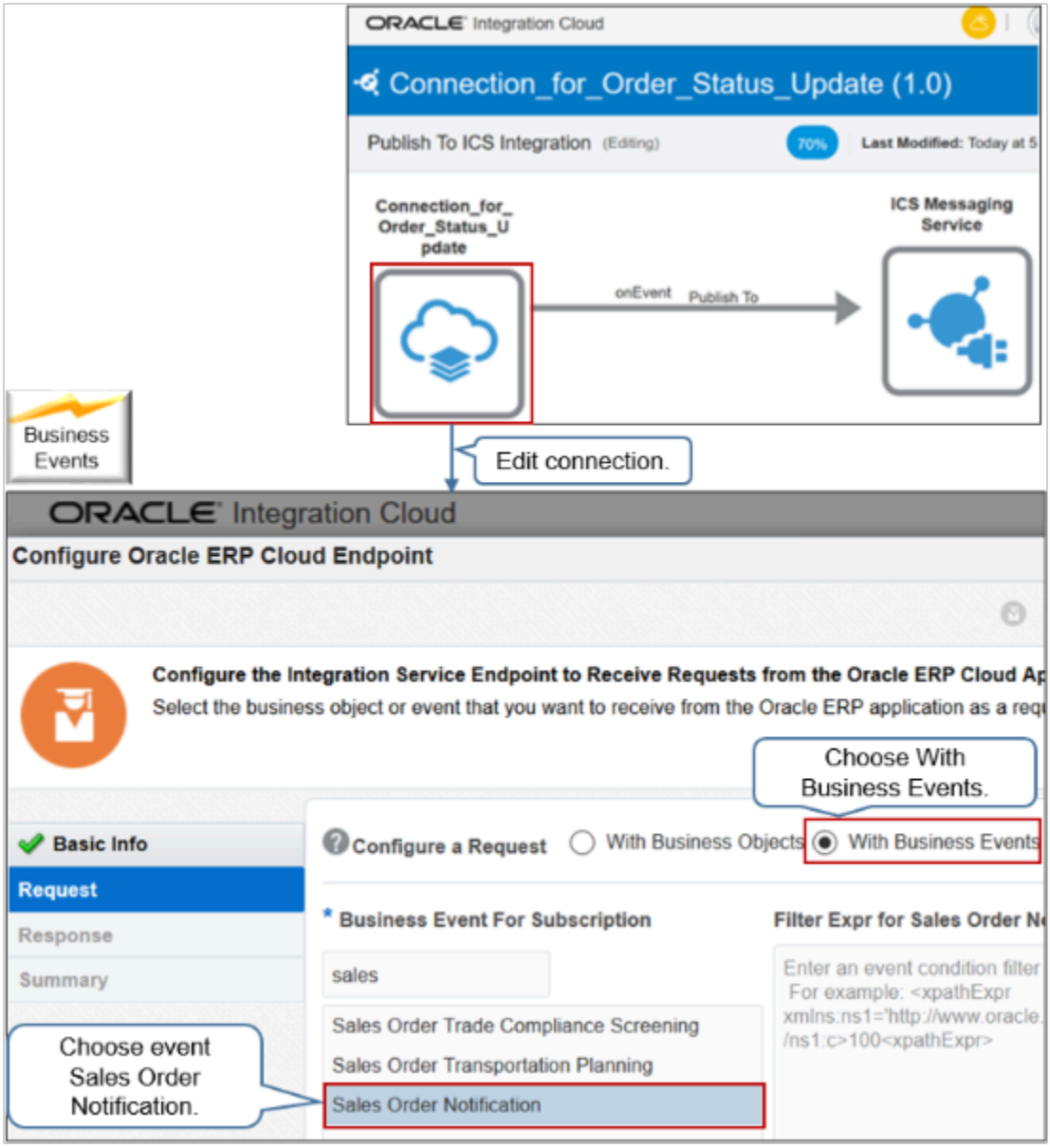
For details, see *Send Notifications from Order Management to Other Systems*.

Create Connector and Integrate

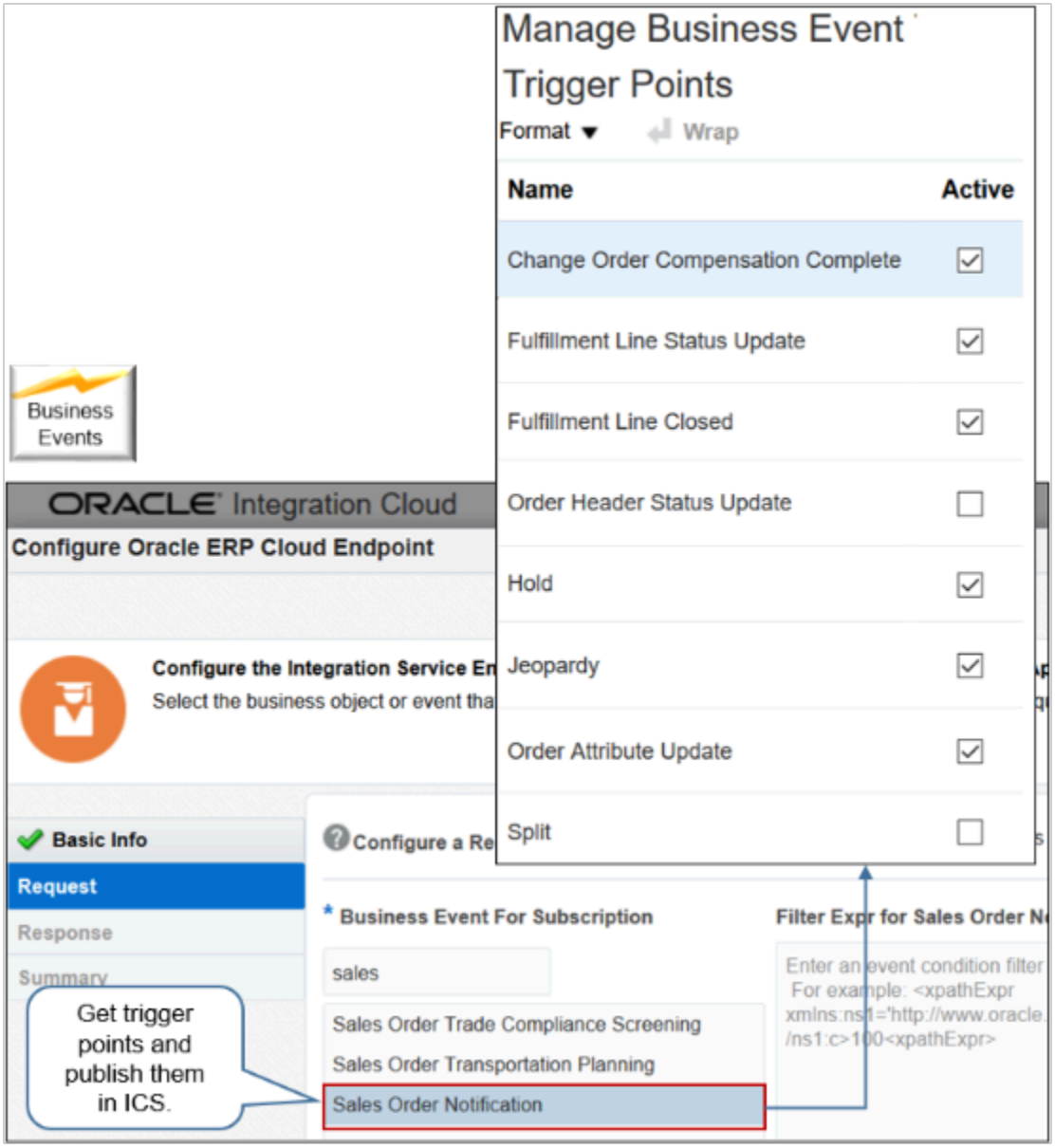
Sign into Integration Cloud Service, create the connector, create the integration, then drag and drop your connector onto the integration.



To configure the end point, click the connection that you dropped onto your integration, then click **Edit**. Use the wizard to select With Business Events and the Sales Order Notification event.



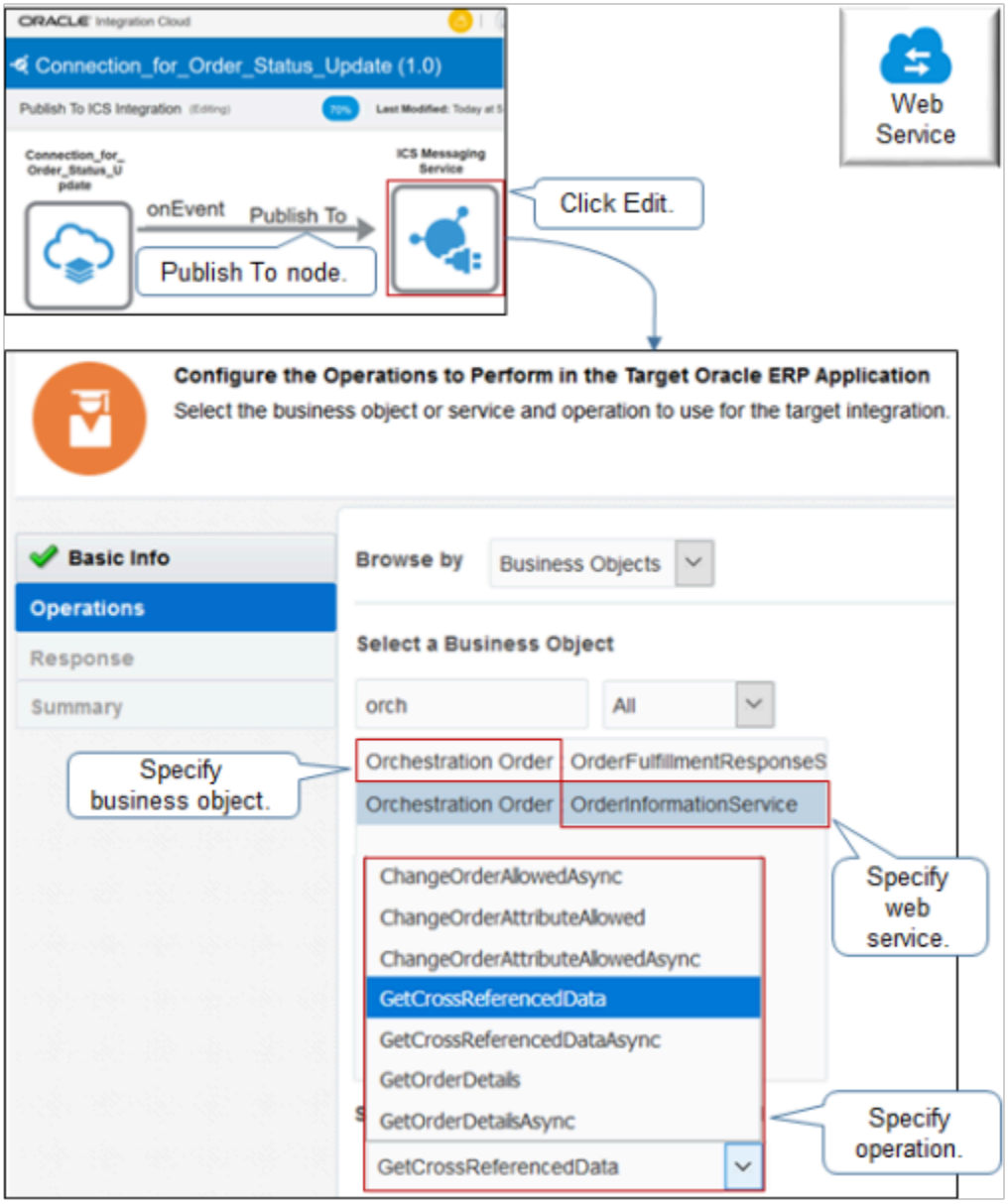
Sales Order Notification gets the active trigger points that you set up on Manage Business Event Trigger Points, then publishes them in Integration Cloud Service.



The trigger points inform each subscriber when something significant happens on the sales order. For example, when Order Management finishes compensation for a change order, updates the status on the fulfillment line, closes the fulfillment line, and so on.

Specify Operations

Specify the operations that each web service must do. You edit the Publish To node in your integration.



Use Order Information Service

Select the Orchestration Order business object, select the OrderInformationService web service, then select an operation.

Operation	Description
ChangeOrderAllowed	Determines whether Order Management allows changes on the sales order or the fulfillment line.
ChangeOrderAttributeAllowed	Determines whether Order Management allows changes on each sales order's attribute.
GetCrossReferencedData	Allow a source system that subscribes to the Sales Order Notification business event to get cross-referenced values that the calling application can use.

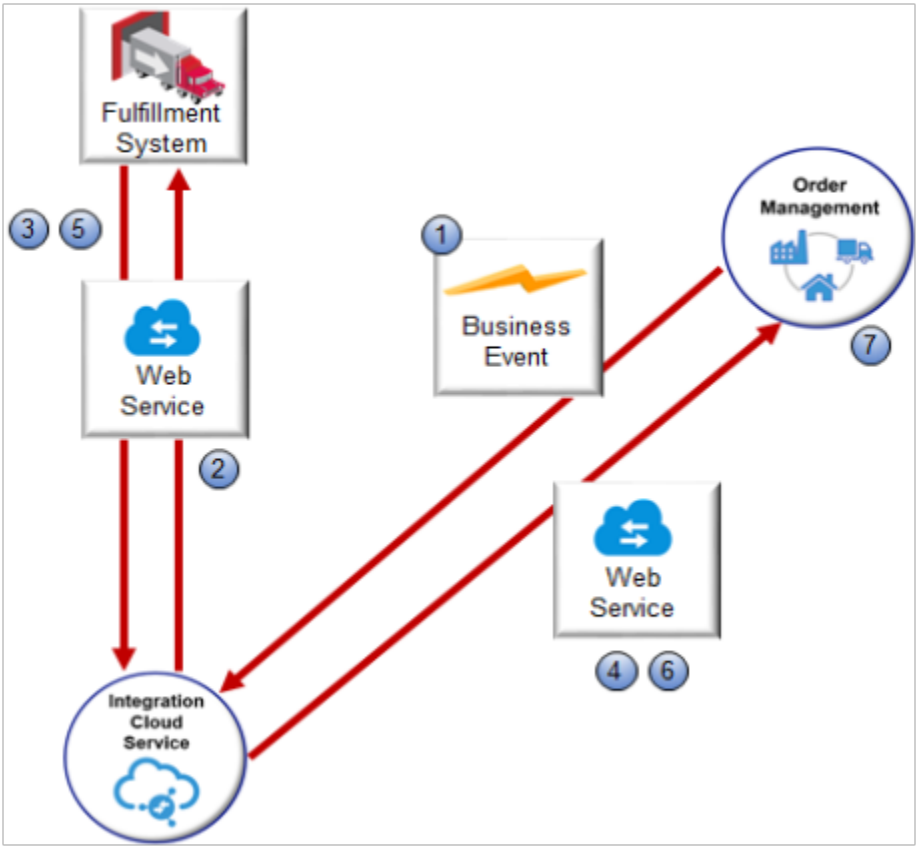
Operation	Description
	Use this operation as an enrichment service when you receive Sales Order Notification. For details, see *Overview of Creating Cross-References in Order Management*.

You can also use the OrderImportService web service to send a new sales order or to revise a sales order.

Learn how to specify the data that you need to communicate. For details, see the Create Connections chapter in *Using Integrations in Oracle Integration*.

Use Order Fulfillment Response Service

Here's how you can use OrderFulfillmentResponseService to process the response that your fulfillment system sends to Order Management.

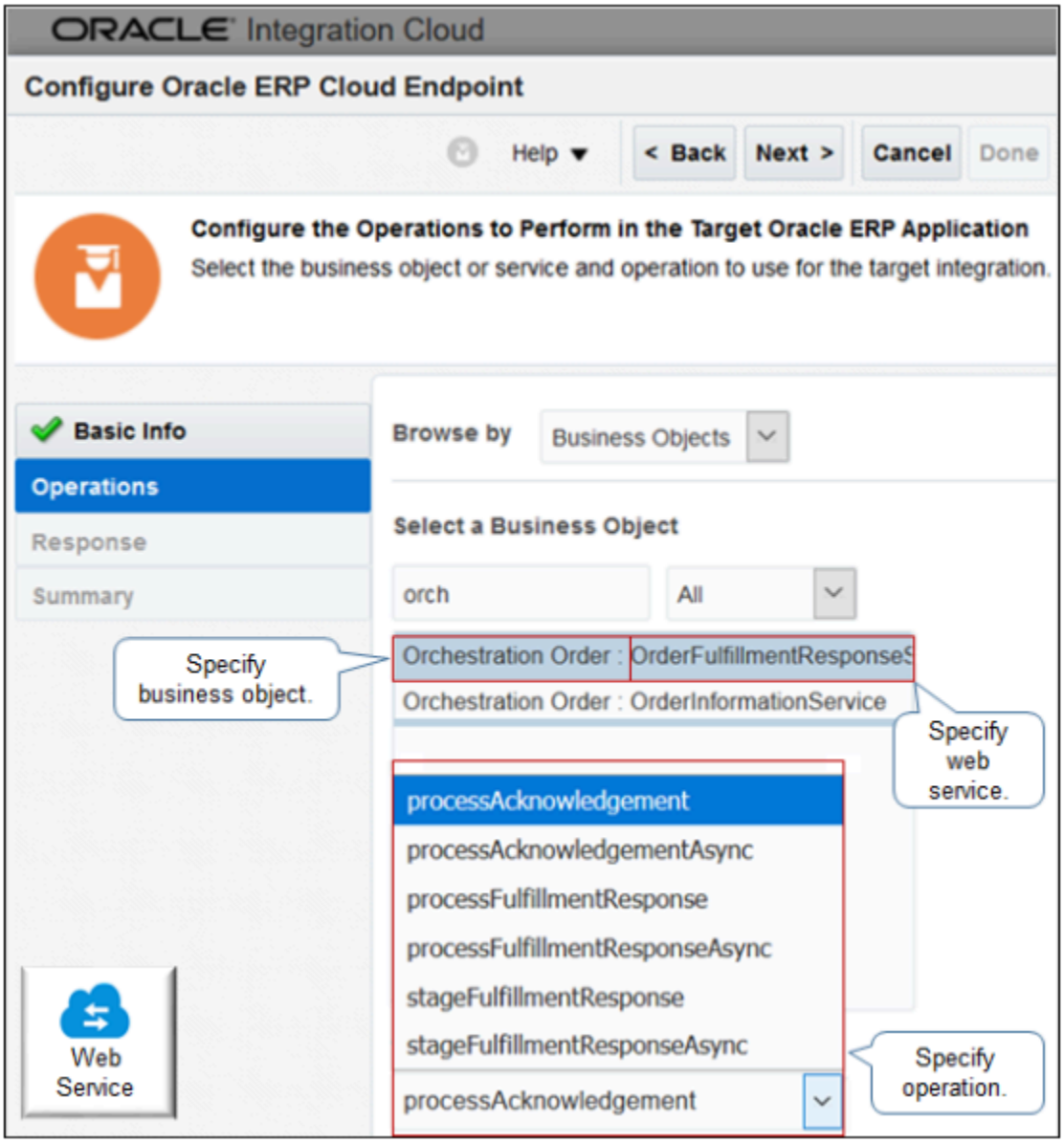


Note

1. Order Management uses a business event so it can send a fulfillment request to your fulfillment system.
2. Integration Cloud Service calls the document service of the business event to send the request to the fulfillment system. The the document service's output payload includes data that the fulfillment system uses to fulfill the request. You can map this output payload to the web service's interface that processes the inbound call on the fulfillment system.
3. The fulfillment system receives the fulfillment request from Order Management, then calls the processAcknowledgement operation of OrderFulfillmentResponseService to acknowledge that it received the fulfillment request.
4. Integration Cloud Service sends the acknowledgment to Order Management.

5. The fulfillment system fulfills the request, then calls the processFulfillmentResponse operation to communicate the result to Order Management.
6. Integration Cloud Service sends the response to Order Management.
7. Order Management processes the response.

Set the web service to OrderFulfillmentResponseService when you configure the endpoint:



Select an operation.

Operation	Description
processAcknowledgement	The fulfillment system will send an acknowledgment that confirms it received the fulfillment request. processAcknowledgement accepts and processes the acknowledgment.

Operation	Description
processFulfillmentResponse	The fulfillment system will call processFulfillmentResponse to communicate the result of the fulfillment request to Order Management. The service adds the **`Response`** suffix to the operation name, which results in a response name of **`processFulfillmentResponseResponse`**. The processFulfillmentResponse name operation indicates that it processes a fulfillment response in reply to an Order Management fulfillment request.
StageFulfillmentResponse	For future use.

Map Source to Target

If you map a data element from the source to a data element on the target, then the interface adds a green check mark to the source and target.

- The target also displays the name of the data element at the source.
- The mapping applies to one target web service, so you must do this mapping for each target system.
- If your deployment must do other operations under Order Information Service, then you must create an integration in Integration Cloud Service, and then map the source payload to the target payload for each channel. However, you only need one connection between ERP Service Catalog and Integration Cloud Service because it can support an integration under each web service that you publish in the catalog with a WSDL (Web Services Description Language).

Learn how to create a mapping. For details see the Map Data chapter in *Using the Oracle Mapper with Oracle Integration*.

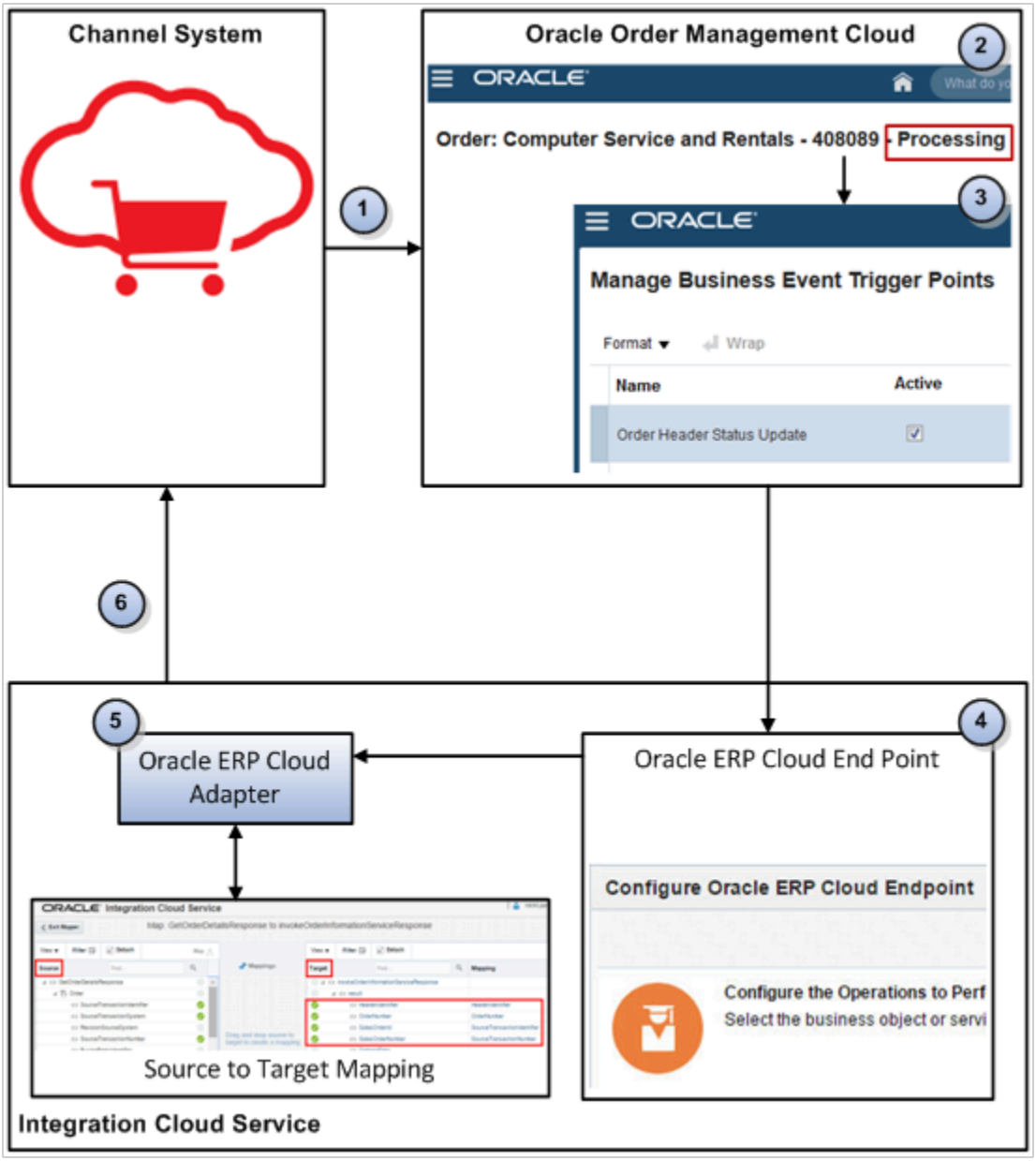
Related Topics

- Use Integration Cloud Service with Order Management
- Filter Details when Using Integration Cloud Service with Order Management
- How Integration Cloud Service Integrates Order Management
- Overview of Creating Cross-References in Order Management

How Integration Cloud Service Integrates Order Management

Order Management uses a business event trigger point to send a notification to each subscriber when the order status changes. The subscriber can also request to get more details about the sales order.

Here's an example that integrates a channel system that's in the cloud.



Assume you set up the integration to broadcast the Sales Order Notification event to subscribers when Order Management updates the order status, such as from Processing to Closed.

1. The channel receives shipment for the sales order, then sends a successful delivery notification to Order Management.
2. Order Management changes the order status from Processing to Closed.
3. Order Management uses a business event. Assume you set the Order Header Status Update event to Active on the Manage Business Event Trigger Points page, and also set it up to use an event when Order Management updates the order header status to Closed.
4. Assume you set up Oracle ERP Cloud Adapter to listen for Sales Order Notification events that are occurring on `https://my_server.com:9999/fndAppCoreServices/ServiceCatalogService?wsdl`.

   Oracle ERP Cloud Adapter recognizes the event, then uses your set up on the Oracle ERP Cloud Endpoint to determine the service and operation to use.

5. To determine how to map source payload to target payload, Oracle ERP Cloud Adapter reads the source to target mapping you set up.
6. Integration Cloud Service sends a notification to the subscriber on the channel.

Integration Cloud Service can integrate Order Management with a channel that's in or out of the cloud.

Related Topics

- Filter Details when Using Integration Cloud Service with Order Management
- Use Integration Cloud Service with Order Management
- Get CSF Keys So You Can Sign Into Integration Cloud Service

Use Integration Cloud Service with Order Management

Use Oracle Integration Cloud Service to integrate Order Management.

Summary of the Set Up

1. Manage trigger points for business events.
2. Send status updates for fulfillment lines.
3. Create a connection.
4. Monitor business events.
5. Track business events.
6. Test your set up.

Learn more about how to do this set up. For details, see *Developing Integration Flows with Integration Cloud Service*.

This topic includes example values. You might need different values, depending on your business requirements.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Manage Trigger Points for Business Events

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Business Event Trigger Points
2. Set up the trigger points so Order Management uses a business event and sends a notification to each subscriber for each trigger point that you need.

    For example, send a notification to each subscriber when Order Management changes the status on the sales order header to Closed.

    - On the Manage Business Event Trigger Points page, click the **Order Header Status Update** row, then make sure the Active option in this row contains a check mark.
    - In the Details area, in the Closed row, add a check mark to the **Raise Event** option.
    - Click **Save**.

Send Status Updates for Fulfillment Lines

Send an update to each subscriber when the status changes on a fulfillment line.

1. On the Manage Business Event Trigger Points page, click the **Fulfillment Line Status Update** row, then make sure the **Active** option in this row contains a check mark.
2. Click **Save and Close**.
3. Go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
4. On the Manage Orchestration Process Definitions page, search for the orchestration process that your deployment uses.

    For example, search for OrderFulfillmentGenericProcess.

    Each orchestration process controls the status value for each fulfillment line, so you must modify the orchestration process that controls the status value. In this example, you modify the orchestration process that controls the shipping status value.
5. In the search results, click the **row** that contains OrderFulfillmentGenericProcess, then click **Actions > Edit**.
6. In the Process Details area, click **Status Conditions > Fulfillment Line Status Values > Edit Status Rule Set**.
7. On the Edit Status Rule Set page, add a check mark to the **Notify External System** option for each Status Value where your deployment must send a notification.

    For example, to send a notification when Order Management changes the fulfillment line status to Shipped, add a check mark to the **Notify External System** option in the Shipped row.
8. Repeat step 7 for other status values, as necessary.

    Order Management will send a notification when each fulfillment line that this orchestration process processes reaches the status you specify in steps 7 and 8.
9. Repeat steps 4 through 8 for each orchestration process in your deployment that updates status values, as necessary.

Create a Connection

Create a connection between Integration Cloud Service and Order Management. For details about how to use Integration Cloud Service, including the URL that you use when you sign in, see *Developing Integration Flows with Oracle Integration Cloud Service*.

1. Create a service request so the Cloud Operations team can register the CSF-KEY (Credential Store Framework). You need the key to sign into Integration Cloud Service.

    Request a separate key for each order administrator who uses Integration Cloud Service. For details, see *Get CSF Keys So You Can Sign Into Integration Cloud Service*.
2. Get the privileges that are in the SOA Infra Operations Duty role. SOA means Service Oriented Architecture. The Order Administrator job role comes predefined with this duty role.
    - If you get privileges from a different role, such as Order Manager, then use the Security Console to get access to the SOA Infra Operations Duty role.
    - If your connection publishes the Sales Order Notification event to Integration Cloud Service, then you must specify the login credentials of an Oracle Cloud Application user so you can access the Event Catalog URL. You must assign the privileges that are in the SOA Operator Role to this user.

- Use privileges from the Order Manager job role so this same user can call the web services that Order Management uses. The integration must use the same login credentials to call web services at run time that you use when you set up the integration at design time. These are Oracle Cloud Application user credentials.

  If you use this connection to publish a business event to Integration Cloud Service, and to call Order Management web services, then the Oracle Cloud Application user must have the privileges that are in the Order Manager job role and the SOA Infra Operations Duty role.

For details, see *Privileges That You Need to Implement Order Management*.

3. Create the connection.
   - Sign into Integration Cloud Service.
   - On the Welcome page, click **Create Connections**.
   - On the Connections page, click **New Connections**.
   - In the Create Connection Select Adapter dialog, under Oracle ERP Cloud, click **Select** .
   - In the Create New Connection dialog, set the values, then click **Create**.

Attribute	Value
Name	Enter any text that describes the connection. For example, `Connection_for_Order_Status_Update`.
Identifier	Enter any text that describe the connection. For example, `CONNECTION_FOR_ORDER_STATUS_UPDATE`.
Role	Select `Trigger and Invoke`.
Description	Enter any text that describe the connection. For example, `Connection for the order status update`.

   - On the page that displays, click **Configure Connectivity**.
   - In the Connection Properties dialog, set the values, then click **OK**.

Attribute	Value

ERP Services Catalog WSDL URL	Enter the URL that locates the WSDL. Use this format.
	
	`https://server:port/fndAppCoreServices/ServiceCatalogService?wsdl`
	
	For example:
	
	`https://my_server.com:9999/fndAppCoreServices/ServiceCatalogService?wsdl`

Attribute	Value
ERP Events Catalog URL (optional)	Enter the URL that locates the events catalog. Use this format. `https://server:port/soa-infra` For example:`https://my_server.com:7818/soa-infra`

These URLs allow Integration Cloud Service to connect to Oracle ERP so Integration Cloud Service can get details about the services and events that are available in Oracle ERP Cloud. Contact your system administrator to determine the URLs you must use.

4. Configure security.
    - Click **Configure Security**.
    - In the Credentials dialog, enter the user name and password you use to access Order Management as an administrator, then click **OK**.
    - At the top of the page, click **Test**, then wait for the indicator that displays immediately to the right of Test to change to a green color, and to display 100%.
    - Click **Save > Close**.

Create an Integration

Here's the integration you create.

Create an integration that monitors business events.

1. On the Connections page, click **Integrations**.
2. On the Integrations page, click **Create**.
3. In the Create Integration Select dialog, under Publish to ICS, click **Select**.

    This procedure provides only part of an example integration. In an actual integration, its more likely you will select Map My Data. To create a full, end-to-end integration, see the Integration Cloud Service documentation.
4. In the Create New Integration dialog, set the values, then click **Create**.

Attribute	Value
What do you want to call your integration?	Describe the connection. For example, enter `Connection_for_Order_Status_Update`.

Attribute	Value
Identifier	Describe the connection. For example, enter `CONNECTION_FOR_ORDER_STATUS_UPDATE`.
Version	Accept the value that displays.
What does your integration do?	Describe the connection. For example, enter `Integration that uses a business event to monitor updates to the order status`.
Which package does your integration belong to?	Leave empty.

5. On the page that displays, you will identify the source of the connection that provides the details. For this example, in the search window, enter `Connection_for_Order_Status_Update`, which is the integration that you created earlier, then click **ENTER**.
   The test you did earlier for the connection you created must finish successfully. If it doesn't, then the search won't return the connection.
6. Drag and drop **Connection_for_Order_Status_Update** from the search results onto the **Drag and Drop a Trigger** area.
7. In the Configure Oracle ERP Cloud Endpoint dialog, specify the events and scenarios for the ERP Cloud connection. Set the values, then click **Next**.

Attribute	Value
What do you want to call your endpoint?	Describe the endpoint. For example, enter `PublishOrderStatusUpdated`.
What does this endpoint do?	Describe the connection. For example, enter `Publish the OrderStatusUpdated event`.

8. Select the **With Business Events** option.
   Integration Cloud Service uses the URLs you set up earlier to get the event catalog, and then display it in the Business Event for Subscription list. This list helps you select from the events that are available in Oracle ERP Cloud.
9. In the Business Event for Subscription list, enter `Order`, then click **Sales Order Notification > Next**.
10. On the Configure the Response to Send to the Oracle ERP Cloud Application page, note that Integration Cloud Service only listens for events in this integration, then sends them to subscribers without replying to Order Management. So, click **None > Next**.
11. On the Summary page, click **Done**.

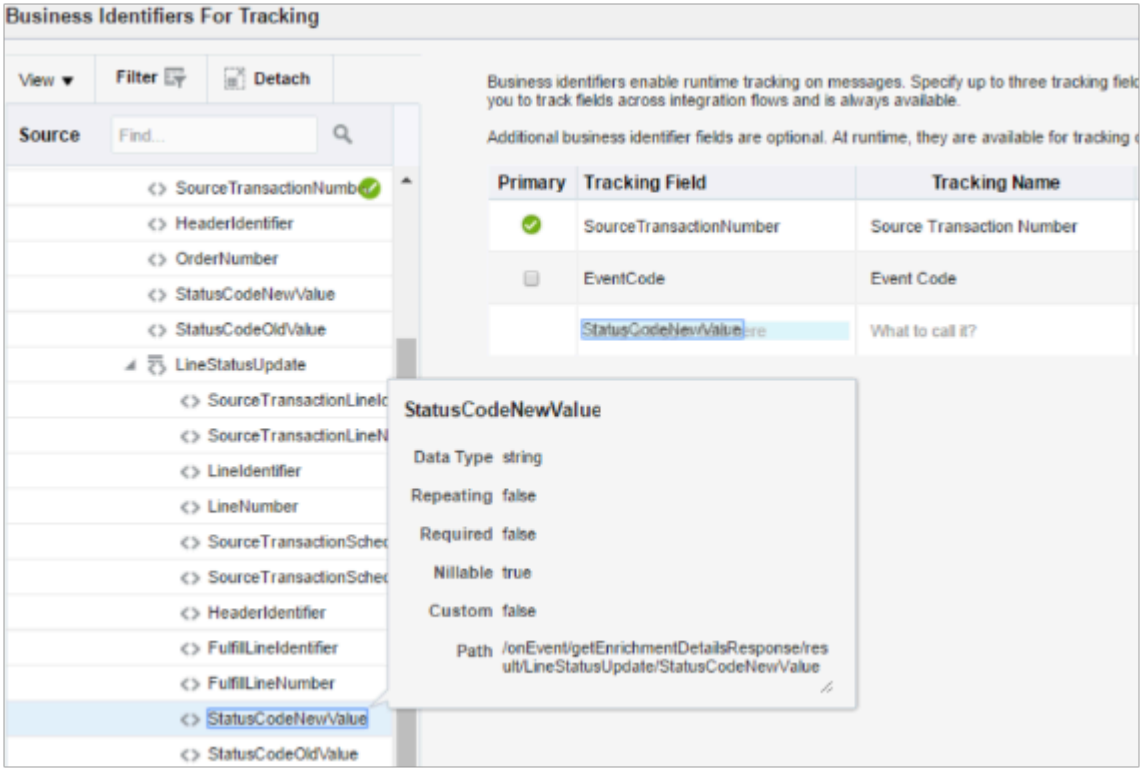
Track Business Events

Track business events so you can test and monitor your integration. Track at least one business event so you can monitor your deployment during normal operations.

1. On the page that displays, click **Tracking**.

Use Tracking to test your set up. You can view the business events that your integration uses in Integration Cloud Service's monitoring system.

2. For this example, on the Business Identifiers for Tracking page, in the Source tree, under the result branch, click **Load More**.
   Here are the business identifiers you will track.



3. Drag and drop **SourceTransactionNumber** from the hierarchy tree to the first row of the **Tracking Field** column.
4. Drag and drop **Event** from the hierarchy tree to the second row of the **Tracking Field** column.
5. In the result branch, expand **LineStatusUpdate**, then click **Load More**.
6. Drag and drop **StatusCodeNewValue** from the hierarchy tree to the third row of the **Tracking Field** column.
7. Click **Done**.
8. On the page that displays, click **Save > Exit Integration**.
9. On the Integrations page, in the Connection_for_Order_Status_Update row, click **Pending Activation**.
10. In the Confirmation dialog, add a check mark to the **Enable Tracing**option, then click **Yes**.

Test Your Set Up

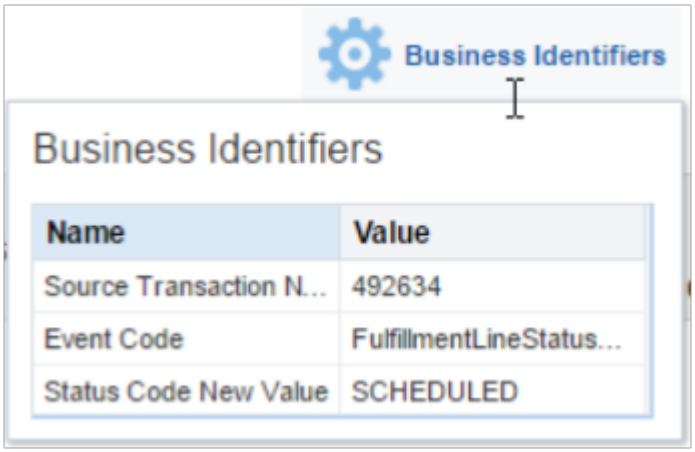
1. Sign into Order Management, create a sales order, add one order line, use the Shipment Details tab to set the Requested Date to a time that happens in the future, then click **Submit**.
2. In the Confirmation dialog, make a note of the order number that displays, such as 492634, then click **OK**.
   The shipment step of the orchestration process runs, sets the Status to scheduled, then displays the status on the fulfillment line in the Order Lines tab. This shipment step uses the business event that you specified earlier. You must have already set up this orchestration process and finished other setup in Order Management.
3. Navigate back to Integration Cloud Service, then click **Monitoring**.
4. On the Integrations Dashboard, click **Tracking**, then click anywhere in the Messages area.

5. In the search window, enter the order number you noted earlier, such as `492634`, then click **ENTER**.
6. In the search results, notice that Order Management used more than one event for the sales order. Click the **event** at the top of the list, which is the most recent event.
7. On the page that displays, click **Business Identifiers**, then verify that the tracking details you set up for tracking earlier displays.

   Here are the details you set up in this example.

   - **SourceTransactionNumber.** Identifies the order number.
   - **Event.** Identifies the business event that gets triggered, which is FulfillmentLineStatusUpdate.
   - **StatusCodeNewValue.** Identifies the new status value of the fulfillment line, which is Scheduled.

   For example:



Other Setups

Get details about other ways you can use Integration Cloud Service with Order Management.

- *Order Management Integration ICS Simulating External Fulfillment / Shipping (Doc ID 2578377.1)*
- *Order Management Integration ICS Testing Integration (Doc ID 2578387.1)*
- *Order Management Integration Submitting a Fulfillment Response Payload From an External Fulfillment Application (Doc ID 2579275.1)*

Related Topics

- Filter Details when Using Integration Cloud Service with Order Management
- Use Integration Cloud Service with Order Management
- Get CSF Keys So You Can Sign Into Integration Cloud Service
- Overview of Integrating Order Management with Other Oracle Applications
- Overview of Security Console

Filter Details when Using Integration Cloud Service with Order Management

Filter the event notifications that Integration Cloud Service sends to your channel or fulfillment system.

1. Use the Manage Business Event Trigger Points page to set up trigger points that enable event Order Status Updated.
2. Use Integration Cloud Service to create an integration.

   For details, see *Use Integration Cloud Service with Order Management* and *Developing Integration Flows with Integration Cloud Service*.
3. Sign into Integration Cloud Service and add filters.

   - Open the mapping that maps the integration to Order Management.
   - To allow events that only originate for sales orders from Order Management, create a filter on the Source Transaction System attribute.
   - To ignore events for a trigger that you set up on the Manage Business Event Trigger Points page, create a filter on the Event Code attribute for each trigger you must filter.

   Use different attributes, such as Country, SourceTransactionSystem, or EventCode, to create an expression.

   Get more details.

   - *Configuring Oracle ERP Cloud Trigger Request Properties*
   - *Configuring Oracle HCM Cloud Trigger Request Properties*
   - Create Basic Routing Integrations in *Using Integrations in Oracle Integration*

Related Topics

- How Integration Cloud Service Integrates Order Management
- Use Integration Cloud Service with Order Management
- Get CSF Keys So You Can Sign Into Integration Cloud Service

Get CSF Keys So You Can Sign Into Integration Cloud Service

Get the CSF (Credential Store Framework) key you need to sign into Oracle Integration Cloud Service.

1. Send a service request to Oracle.

Service Request Section	Value
Subject line	Include text. `Create the CSF-KEY that we can use to integrate Order Management with Oracle Integration Cloud Service`
Body	Provide the identity domain that your integration can use for Order Management and for Oracle Integration Cloud Service subscriptions. For example, `icssvc.identity.domain=idm2152`. Request a separate CSF-KEY for each order administrator who must access Integration Cloud Service.

2. Wait for Oracle to reply to your service request. When Oracle does reply, provide them with the user name and password that you use to access Integration Cloud Service. Oracle will then create the CSF-KEY.

Related Topics

- Filter Details when Using Integration Cloud Service with Order Management
- How Integration Cloud Service Integrates Order Management
- Use Integration Cloud Service with Order Management

Cross-References

Overview of Creating Cross-References in Order Management

Create and maintain cross-references that relate data between your source system, fulfillment system, and Oracle Order Management.

Each cross-reference helps manage data across systems. For example, your source system might use the value **Net 30 Days** for a payment term while your billing system uses **30 Days**. Use a cross-reference in Order Management to standardize how you represent the payment term.

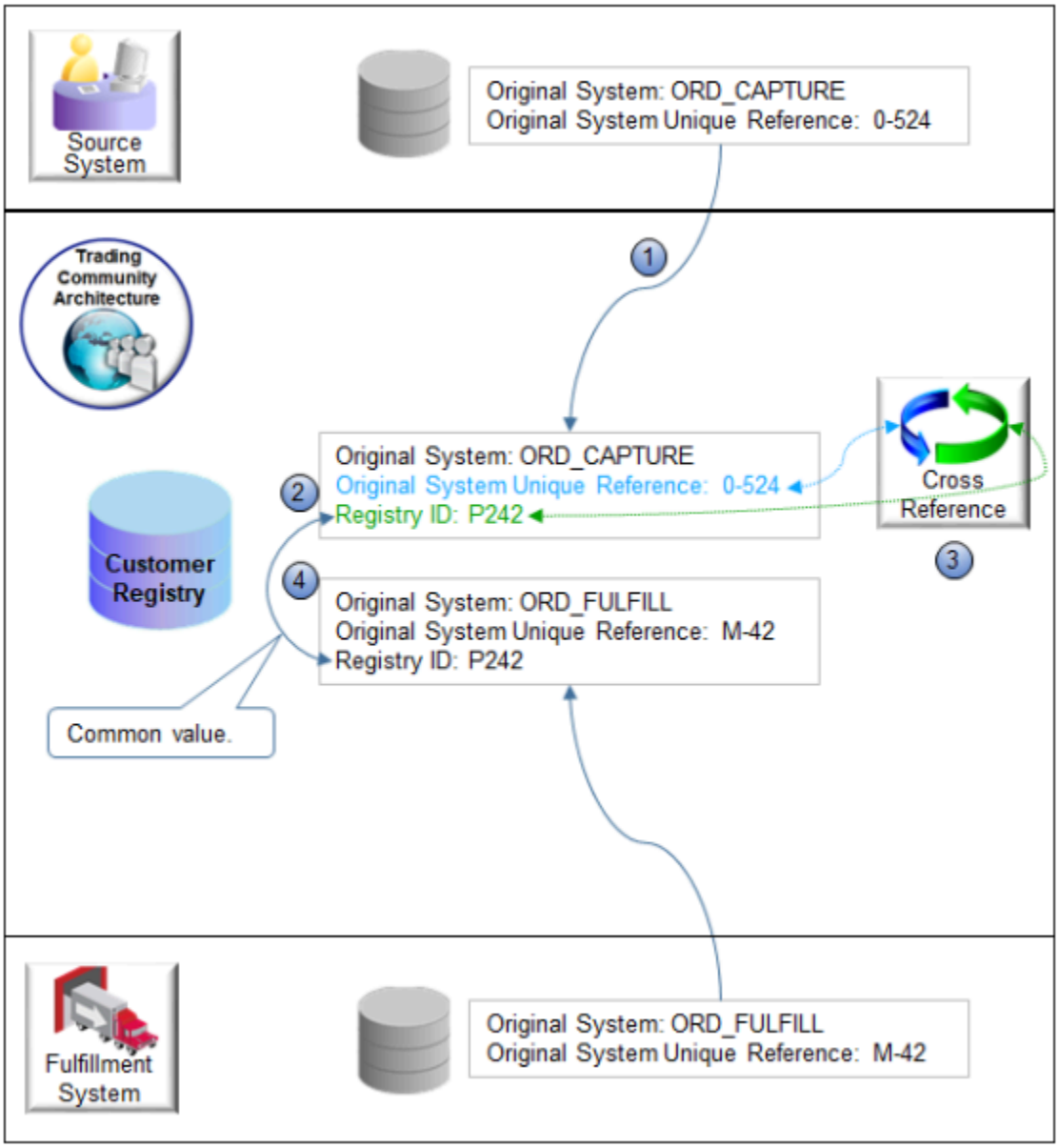
Here's how it works.

Interaction	Description
Receive a source order from your source system.	The web service that receives the source order uses a cross-reference in the Order Orchestration and Planning repository to resolve the imported value to a common value in Oracle Applications. If a cross-reference doesn't exist between your source system and the common value, then the order fails with an error when you submit it to fulfillment.
Send a request to your fulfillment system.	Order Management cross-references values from the common value to the source system value. If it can't find the source system value in the Order Orchestration and Planning repository, in Trading Community Architecture, or Product Information Management, then fulfillment goes into error.

Cross-Reference Your Customer

You can maintain cross-references between your source system and Trading Community Architecture.

For example, assume your source system sends a record that includes an identifier value of O-524.



The web service uses the Original System Unique Reference (OSR) to map customer details from the source system to the registry ID in the customer registry in Trading Community Architecture.

1. Load the record from the source system into the customer registry.
2. Assign a registry ID of P242 to the record in the registry.
3. Create a cross-reference between Original System Unique Reference O-524, and registry ID P242.
4. Use registry ID P242 as the relationship to Original System Unique Reference M-42 from the fulfillment system.

   P242 is the common value that ties the source system to the order fulfillment system.

Here's how it works at runtime.

1. Your source system sends a source order to Order Management that includes customer data.
2. If the customer already exists in Trading Community Architecture, then Order Management uses a customer cross-reference to get the master customer record and the customer ID of the fulfillment system.

3. Order Management sends the sales order, including the customer ID and other attributes it gets from Trading Community Architecture, to the fulfillment system.

Note

- You can use other values to establish the cross-reference, such as customer name, contact name, address, and so on.
- Order Management expects the value in Trading Community Architecture will match the value in your source system. If your source system sends a value, then the web service will use it to identify and resolve the master data instead of using the Original System Unique Reference.
- To help identify each attribute that references a source system, make sure you include the phrase **Original System Reference** in the attribute name when you cross-reference your customer master data. For example:
  - Buying Party Original System Reference
  - Ship To Address Original System Reference
  - Bill To Address Original System Reference
- If you don't use Trading Community Architecture, but instead use some other application outside of Oracle to store customer data, then you must still maintain cross-references in Trading Community Architecture.
- Learn about integrating with Trading Community Architecture. For details, see *Overview of Displaying Customer Details on Sales Orders*.

Cross-Reference Your Item

Use attributes of the item to relate data between different systems.

- **Source system.** A relationship between the item in the source system and the Oracle item when you use the Product Information Management work area.
- **Named item.** A relationship between the item in the source system and the Oracle item when you bring items from different systems into Product Information Management.

Use the Source System Product Reference attribute when you cross-reference item data.

Collect Orchestration Data

Order Management creates a cross-reference between each attribute in your order capture system and each attribute in your fulfillment system so it can use a single representation across different systems. The order capture systems and fulfillment systems use these values to communicate with Order Management. You must use Trading Community Architecture to set up customer data and Product Information Management to set up product data before you collect.

Here are the types of data you must collect.

- Activity types
- Currencies, conversions, and types
- Demand class
- Document category
- FOB points
- Freight term
- Invoicing and accounting rules
- Payment methods

- Payment terms
- Receipt methods
- Return reason
- Resources
- Sales credit types
- Shipment priority
- Shipping carriers
- Shipping class of service
- Shipping mode of transport
- Tax classification code
- Tax exemption reason
- Units of measure
- Warehouse

These attributes aren't related to the customer or the item.

Use data collection tools.

- **Continuous collection.** Collect data incrementally. It provides fast collection for each entity that Order Management must source.
- **Targeted collection.** Refresh data for a single business object. You run a scheduled process that does targeted collection, as needed or on a schedule.

For details, see *Collect Planning Data for Order Management*.

Cross-reference a Whole Bunch of Items

If you must cross-reference a large volume of items, then use Open Interface tables to import them. You can automatically create cross-references for a spoke system when you use item batches through Open Interface tables or web services to import a spoke system item. For details about how to import cross-references, see *Document ID 1311629.1 (Oracle PIM: Sample SQL to Import Item Cross References into Oracle Product Information Management Using Open Interface Tables)* on My Oracle Support.

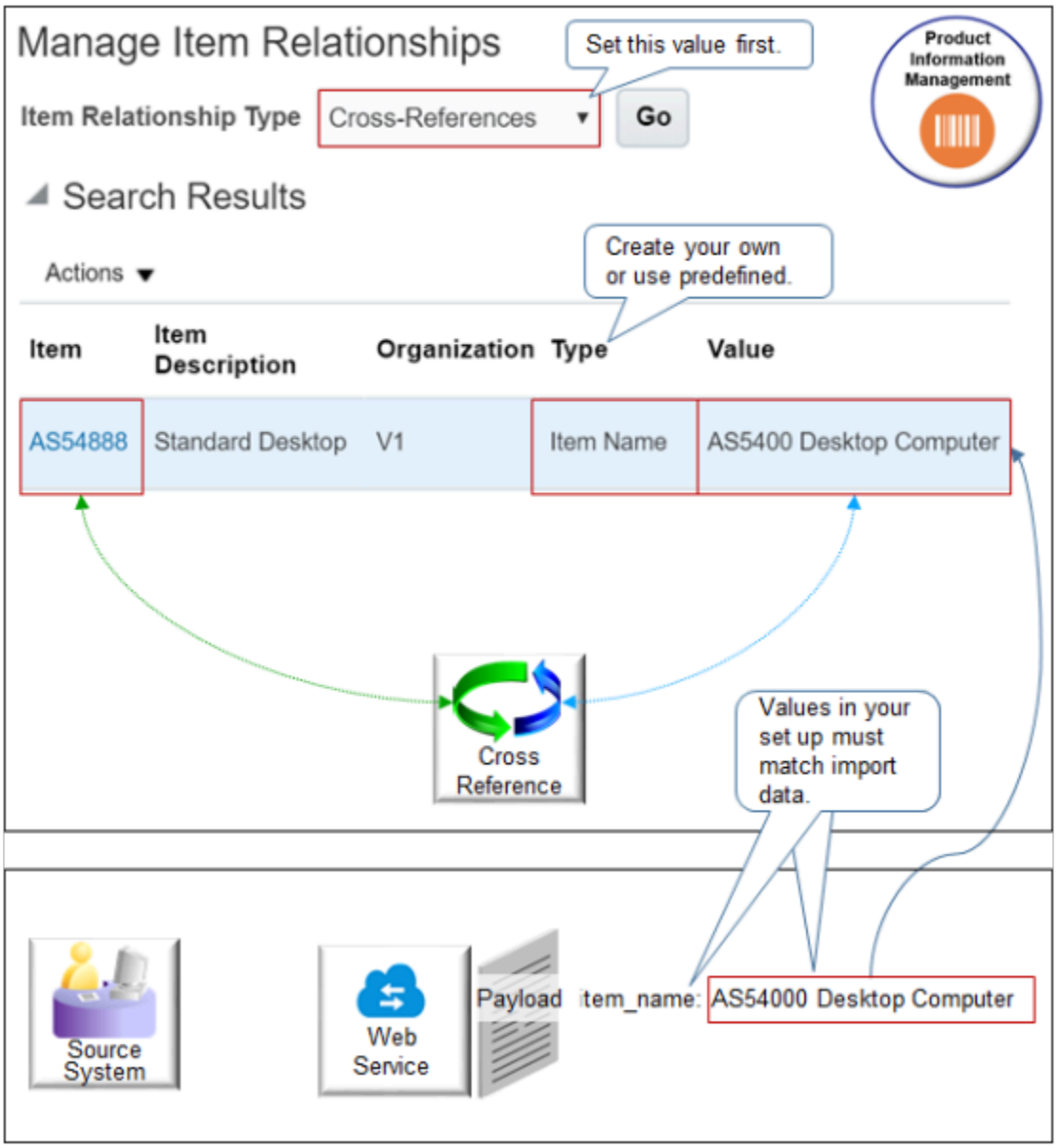
Related Topics

- Overview of Integrating Order Management with Other Oracle Applications
- Import Customer Data for Your Sales Orders
- Overview of Collecting Promising Data for Order Management
- How Order Management Transforms Source Orders Into Sales Orders
- How the Order Orchestration and Order Promising Processes Use the Collected Planning Data

Create Cross-References in Order Management

Use Product Information Management to create a cross-reference for Order Management.

In this example, assume your source system uses AS54000 Desktop Computer for the item name, but you use AS54888 Desktop Computer in Product Information Management and in your fulfillment system. You will create a cross-reference that establishes a relationship between these names.



This topic uses example values. You might need different values, depending on your business requirements.

For details about how to create cross-references, including using the Manage Item Relationships page, see *Document ID 1309859.1 (Oracle Product Information Management, Understanding Item Cross References)*, on My Oracle Support.

1. Make sure you have the privileges that you need to administer Order Management.

2. Create a lookup.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Cross Reference Types

      You will use this lookup later in this procedure.
    - On the Manage Cross Reference Types page, make a note of the rows that the search results display.
        - Charge Master
        - Old Item Number
        - Supplier Part Number
        - Golden Tax Adaptor

      You can use these predefined values, or create a new one. For this example, you create a new one.
    - In the Lookup Codes area, click **Actions > New**, set the values, then click **Save and Close**.

Attribute	Value
Lookup Code	item_name The value you enter must match the value you import from the source system. Product Information Management uses this value when it receives data from your source system, such as in a web service payload, to complete the cross-reference.
Enabled	Contains a check mark.
Meaning	Item Name
Description	Item relationship type to use when cross-referencing to an old item name in a legacy source system.

3. Create the cross-reference.
    - Make sure you have the privileges that you need to administer Product Information Management.
    - Go to the Product Information Management work area, then click **Tasks > Manage Item Relationships**.
    - On the Manage Item Relationships page, set **Item Relationship Type** to Cross-References, then click **Go**.
    - In the search results, click **Actions > Create**
    - In the Create Cross-Reference Relationship dialog, set the values.

Attribute	Value
Organization	Vision Operations

Attribute	Value
Item	AS54888
Type	Item Name This is the value you created earlier in this procedure on step 2.
Value	AS54000 Desktop Computer
Description	Cross-reference to the AS54000 on our legacy source system.

- Click **Save and Close**.

  Product Information Management creates a cross-reference between item AS54888 in Product Information Management and item AS54000 Desktop Computer in your legacy order capture system.

Use the Product Information Management Work Area

If you use the Product Information Management work area, then you set Item Relationship Type to Spoke System Items. This relationship is useful when you must map and identify items that Product Information Management consolidates from more than one source system into a single master item.

Do step 3 described above, but with these differences.

1. On the Manage Item Relationships page, set **Item Relationship Type** to Spoke System Items.
2. In the Create Spoke System Item Relationship dialog, set the values.

Attribute	Value
Organization	Vision Operations
Item	AS54888 Desktop Computer
Spoke System	LEGACY_SOURCE_SYSTEM Assume you already set up this spoke system in the Product Information Management work area.
Spoke System Item	AS54000 Desktop Computer

Product Information Management creates a cross-reference between item AS54888 in Product Information Management and item AS54000 Desktop Computer in the Product Information Management work area.

Related Topics

- Overview of Using Business Rules With Order Management

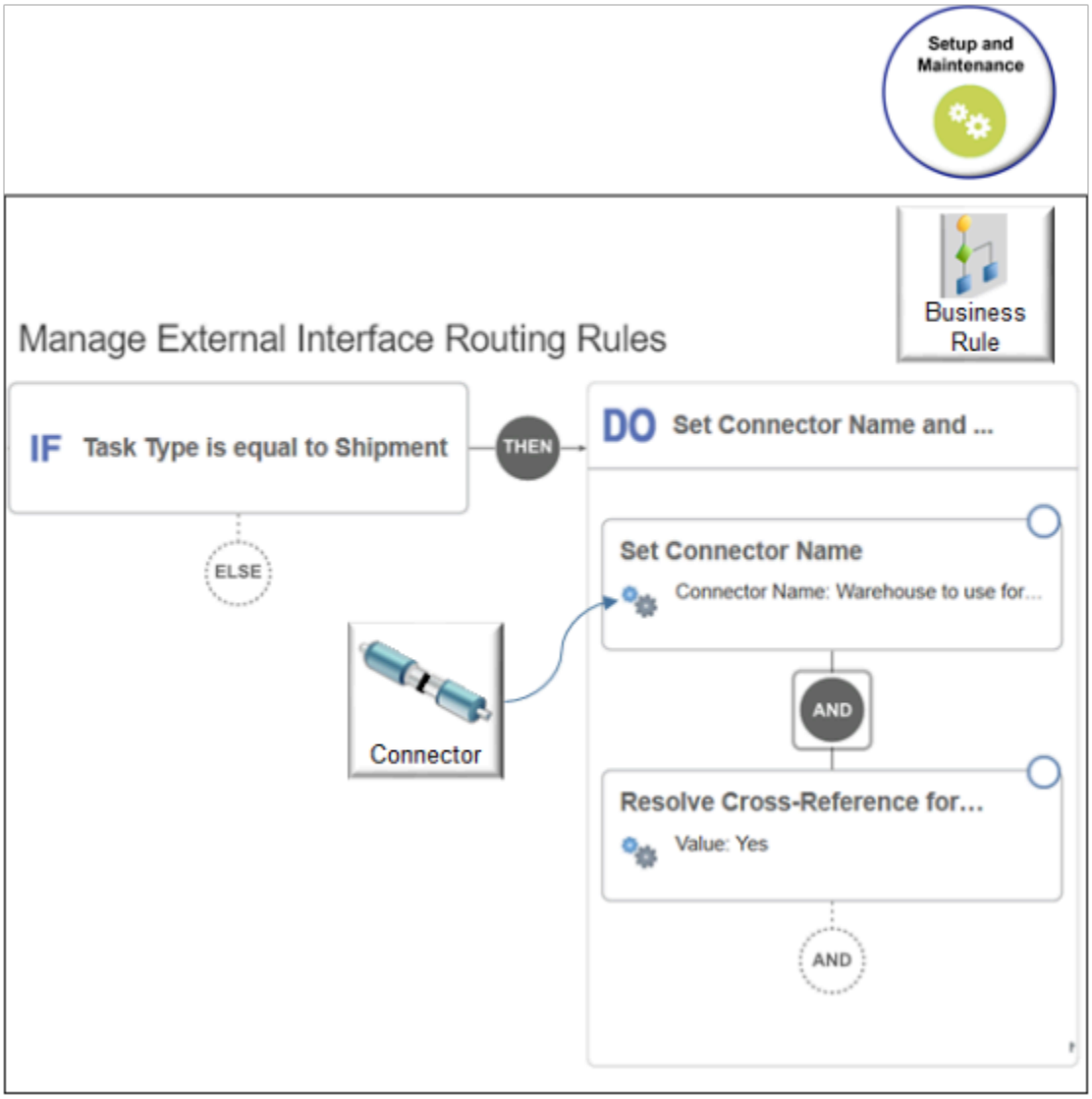
Integrate Order Management Without Cross-Referencing Customer Attributes

Integrate Order Management with your fulfillment system without creating a cross-reference to a customer attribute in the fulfillment system.

Assume you must create a routing rule.

- If the task type is FulfillOrder, and if Organization is 204, then use Oracle Application values that identify customer data to resolve cross-references to the fulfillment system.

You will create this rule.



Summary of the Set Up

1. Create the If statement.
2. Create the Then statement.

This topic uses example values. You might need different values, depending on your business requirements.

Create the If Statement

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Integration Routing Rules for Sales Orders

2. On the Manage External Interface Routing Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Route Without Customer Attributes
Description	Route fulfillment requests without using a customer attribute that resides in the fulfillment system as a cross-reference.

3. Click **New Condition**.
4. In the Create Condition dialog, enter `Task`, wait a moment, then click **Task Type (Order Header)**.
5. Click **Search**.
6. In the Search dialog, enter `FulfillOrder`, then click **Search > OK > OK**.
7. In the IF area, click **And**.
8. In the Create Condition dialog, enter `Organi`, wait for the list to populate, then click **DestinationShippingOrganizationCode**.
9. In the window under Is Equal To, enter `204`, then click **OK > Save**.

Create the Then Statement

1. On the flowchart, click **Then > Do > New Action > Perform an Action**.
2. In the Create Action dialog, choose Resolve Cross-Reference for Customer.
3. Choose a value, then click **OK**. For this example, choose Yes.

Value	Description
Yes	Use a cross-reference from an Oracle Application.
No	Use a cross-reference from the fulfillment system.

If you choose Yes, and if cross-references to attributes in your fulfillment system.

- **Don't exist**. Order Management only sends values from Oracle applications that identify customer data. It doesn't send cross-references. Use this option to send the fulfillment request when the fulfillment system doesn't contain a record for the customer. You must make sure your downstream integration creates or synchronizes customer details before Order Management sends the fulfillment request.
- . Order Management sends values from the source system and identifiers from Oracle Applications. Use this option when the fulfillment request is for a new customer who doesn't exist in the fulfillment system. Use it to create and synchronize customer details to the downstream connector before you send the fulfillment request to the fulfillment system.

4. Click **Save and Close**.
5. On the Manage External Interface Routing Rules page, click your **rule**.
6. In the dialog that displays, add a check mark to **Activate Rule**, then click **Save and Close > Publish**.

Related Topics

- Use Visual Information Builder
- Manage Routing Rules

Entities You Can Cross-Reference in Order Management

Use the Order Orchestration and Planning repository to store data that you collect from different source systems and fulfillment systems, then use them to cross-reference the entities that Order Management uses during order import or to resolve cross-references.

Here are the types of entities you can store in the repository.

Type of Entity	Description
Global	Reference more than one instance of an entity that resides in more than one source system. Data Collection collects each entity into the Order Orchestration and Planning repository, validates it, then consolidates all the entities it collects into a single record that represents all entity instances across all systems. This single record is the global entity.
Source	Don't represent the same data in more than one source system. The instances of source data are distinct. For example, one source system might consider Warehouse as an organization while another source system uses some other classification.

Data Collection collects the transactions that are related to supply and demand so Global Order Promising can do planning and promising. It collects transactions into the Order Orchestration and Planning Repository. You set up the source system so Order Management can store details that identify the source of records and manage collection. The repository uses these details so it can identify the source system where records originate.

Global Entities That Data Collection Can Collect

- Activity Types
- Conversion Types
- Currencies
- Demand Classes
- Document Categories
- Free on Board
- Freight Carriers
- Freight Terms
- Invoicing and Accounting Rules
- Payment Terms
- Receipt Methods
- Return Reasons

- Sales Credit Types
- Ship Class of Service
- Ship Mode of Transport
- Shipment Priority
- Tax Classification Codes
- Tax Exemption Reason
- Unit of Measure

Data Collection can collect these source entities.

- Reason
- Warehouse

Related Topics

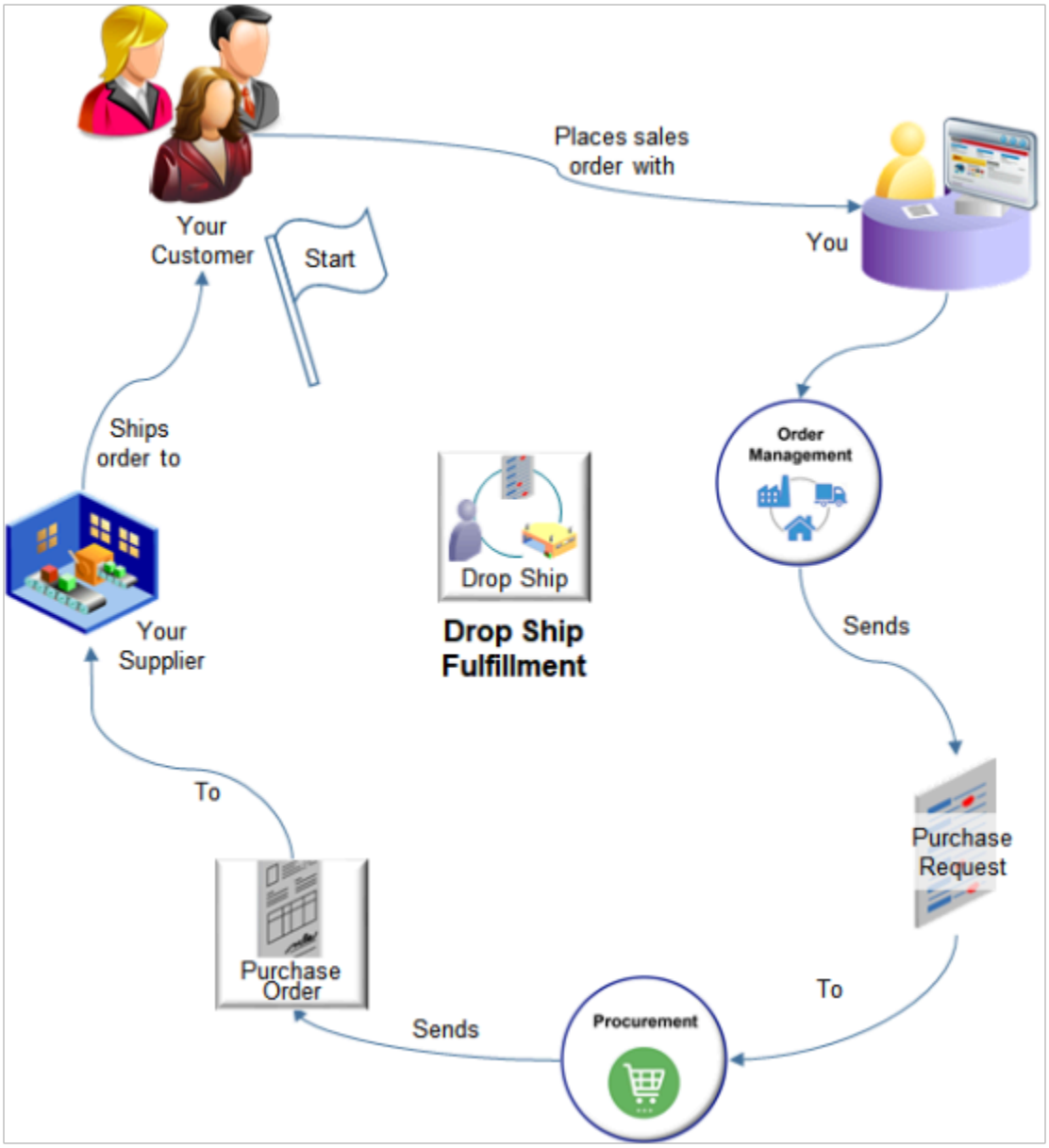
- How the Order Orchestration and Order Promising Processes Use the Collected Planning Data
- Data Collection Sequence

Drop Ship

Overview

Overview of Drop Ship in Order Management

Drop ship is a supply chain management technique where the seller relies on a supplier or contract manufacturer to build, store, and ship an item to your customer. You can use Order Management to automate this process.



Note

- Assume your customer places a sales order with you and you want to drop ship it.
- Order Management sends a purchase request to Oracle Procurement, which places a purchase order with your supplier, then your supplier ships directly to your customer.
- You provide a purchase order for the item and instructions that describe how to ship directly to the customer.
- The supplier or contract manufacturer ships the item, and your company earns a profit.

For a more detailed flow, including screen prints of various applications that you use in the flow, see *Order to Cash Drop Ship Order Flow (Doc ID 2278649.1)*.

Use drop ship to get results.

- Reduce costs for holding inventory.

- Let you supplier manage part of the supply chain, such as fulfilling and shipping the item, instead of you having to do it.
- Forecast and plan for future demand.
- Let your customer place an order with you, and you promise a ship date.
- Automatically place an order with your supplier.
- Let your supplier ship directly to your customer.
- Receive notification from your supplier when your shipment has shipped.
- Let the buyer modify the purchase order.
- Combine more than one sales order into a single purchase order, then fulfill them together.
- Modify a sales order after you create the purchase order.
- Modify the sales order or purchase order after your supplier ships part of the purchase order.
- Manage change orders.

Note

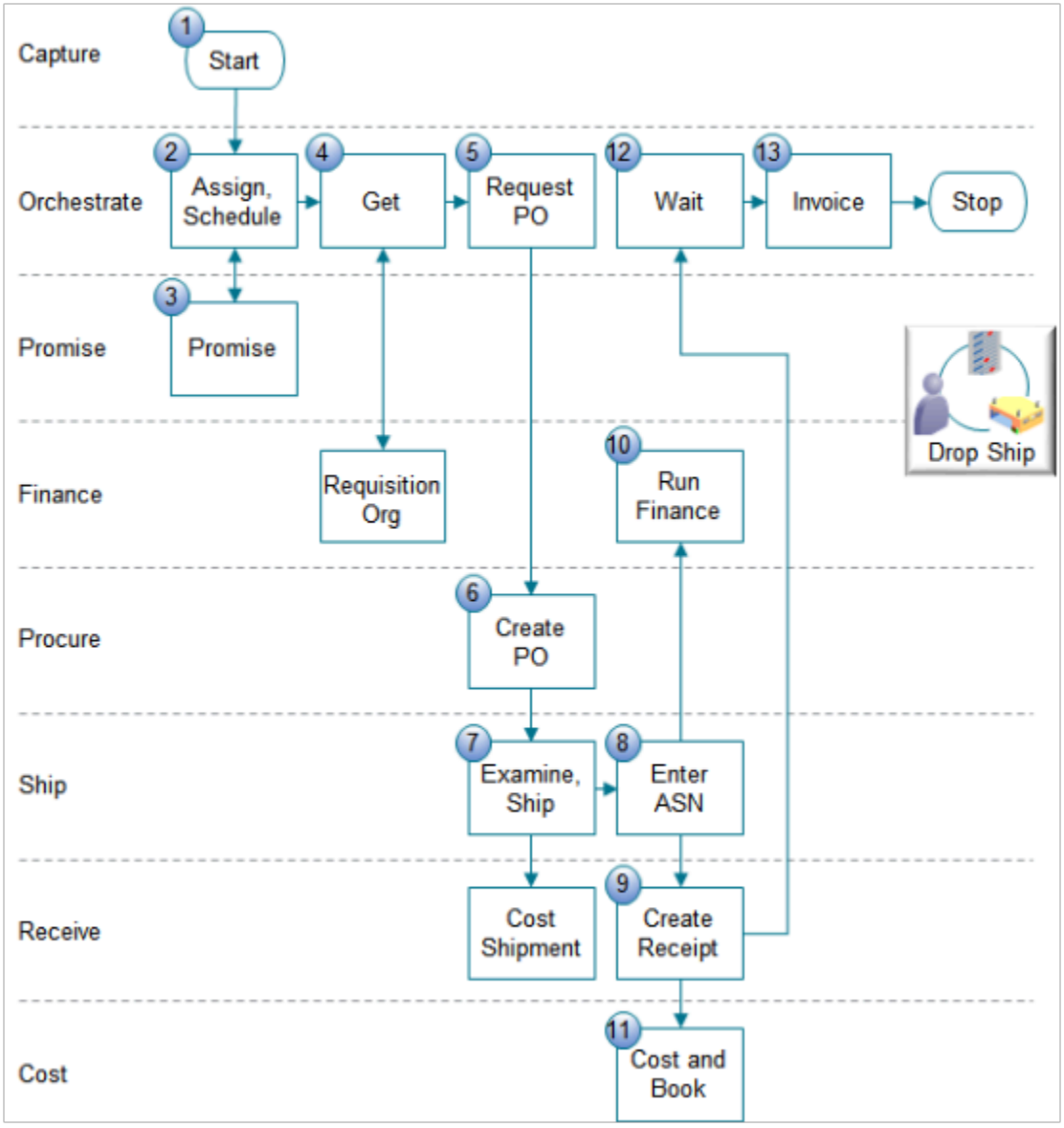
- You can drop ship a standard item, configured item, kit, configuration model, pick-to-order item, or assemble-to-order item.
- You can't drop ship a subscription or coverage.
- Your customer can return a drop ship order only to your warehouse. Your customer can't return a drop ship order directly to the supplier. When you receive the item in the warehouse, use the Return Receipts action in Inventory Management so Inventory Management can deplete the received quantity.
- You can't use a third party application in your drop ship flow.

Related Topics

- How Drop Ship Works in Order Management
- Set Up Drop Ship in Order Management
- Schedule Fulfillment Lines Manually

How Drop Ship Works in Order Management

Use Order Management to manage your drop ship flow.



Note

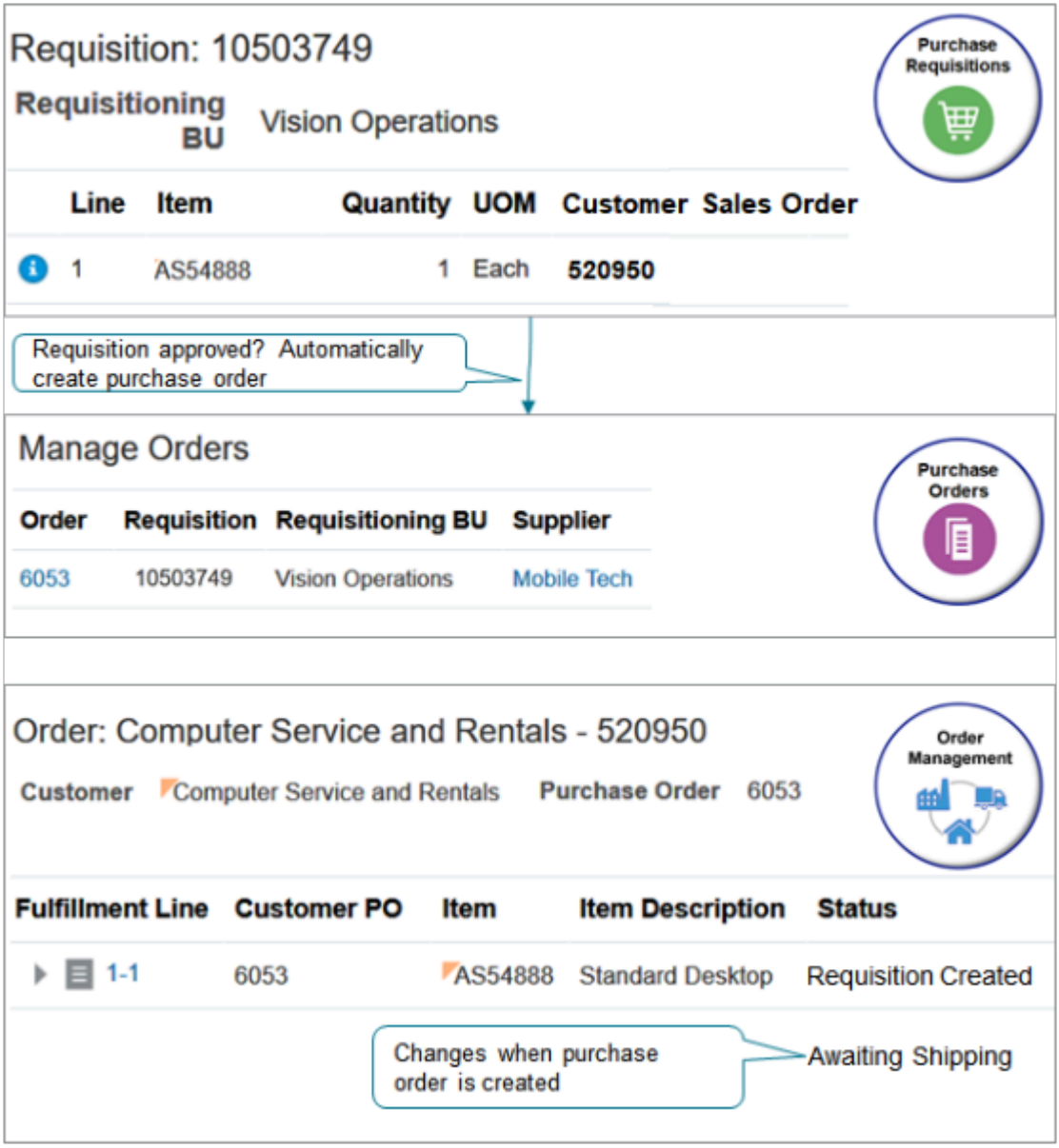
1. An order capture system captures a source order.
    - Each horizontal row in the diagram represents a system or application.
    - The drop ship flow is a variation of the order-to-cash flow.
    - Order Management comes predefined to capture sales orders in the Order Management work area. You can also capture source orders in an order capture system that resides outside of Order Management.
2. Order Management assigns an orchestration process to the sales order, uses the Schedule Order task to start scheduling the sales order, then sends a scheduling request to Global Order Promising.
3. Global Order Promising considers sourcing rules, supplier calendar, capacity, and so on to identify the supplier and supplier site that can fulfill the order in the most efficient way. If the sales order specifies a supplier, then Global Order Promising uses this supplier.
4. The orchestration process gets the requisition organization from Supply Chain Financial Orchestration so it can create a purchase request.

5. Order Management sends a purchase request to Oracle Procurement.

   Note

   - Procurement doesn't use the Product Category or Product Fiscal Classification to create a purchase order.
   - Order Management doesn't set a default value for the Product Category attribute on the order line. The Order Entry Specialist can specify this value when setting tax determinants, but Order Management doesn't send it to Procurement, so the purchase order doesn't contain a value for Product Category. For details, see *Edit Tax on Order Lines*.
   - Order Management sets a default value for the Product Fiscal Classification in the sales order. The purchase order in Procurement also has a Product Fiscal Classification attribute and Procurement sets it to the same value that's on the sales order, but it isn't required.

6. Procurement creates a purchase requisition, approves the requisition, creates a purchase order, then sends the purchase order to the supplier.



Note

This Attribute	References This Attribute
Customer Sales Order attribute in the Purchase Requisitions work area	Number attribute on the sales order in the Order Management work area
Requisition attribute in the Purchase Orders work area	Requisition Number attribute in Purchase Requisitions

This Attribute	References This Attribute
Drop Shipment PO attribute on the Supply Details tab of the fulfillment line in the Order Management work area	Purchase order
Requisition attribute on the Supply Details tab of the fulfillment line in the Order Management work area	Purchase requisition

- Order Management sets the order line status when Procurement creates the requisition and the purchase order.

What Procurement Does	Order Line Status
Creates the requisition.	Requisition Created
Creates the purchase order.	Awaiting Shipping - Receiving can send an advance shipment notice to Order Management after the status goes to Awaiting Shipping. - Receiving can create an advance shipment notice for the entire quantity or for only part of the quantity.

- If a blanket purchase agreement exists, then Procurement might source the requisition from the agreement.
- Procurement sends responses to Order Management while the purchase order moves through its lifecycle.

Response	What Procurement Did
PO_IMPLEMENTED	Created the purchase order.
PO_CO_IMPLEMENTED	Added a change to an existing purchase order.
PO_CO_RESCINDED	Rejected a change in the purchase order.
REQ_LINE_CANCEL	Canceled a line in the purchase requisition.

7. The supplier examines the purchase order and uses your shipping fulfillment system to ship the item to the customer. The enterprise supplier also communicates with the receiving part of Inventory Management to cost the shipment.

8. The supplier uses a supplier portal to enter an ASN. Order Management sends the notice to interested parties and Financial Orchestration. For details, see *Supplier Portal Integration*.
9. The receiving part of Inventory Management creates a logical receipt in the receiving organization. It doesn't create a physical receipt because the supplier ships the item directly to your customer.
10. Financial Orchestration runs a financial flow that comes predefined to handle drop ships, and that specifies how to handle the flow that runs from the supplier to the customer.
11. Cost Management does receipt accounting and cost accounting.
12. Order Management waits to receive the advance shipment notice from receiving. The notice indicates that the item shipped and that the customer acknowledged receipt.
13. Order Management communicates with Oracle Receivables to create an invoice and process payments.

You can modify some of these steps. For example, use Manage Sourcing Rules to modify how Global Order Promising considers sourcing rules, or use Manage Drop Ship Financial Flows to modify how Supply Chain Financial Orchestration sets up a relationship between the selling business unit and the requisition organization. For details, see *Set Up Drop Ship in Order Management*.

Related Topics

- Overview of Drop Ship in Order Management
- Set Up Drop Ship in Order Management
- How Order-to-Cash Works in Order Management
- How Order-to-Cash Works with Order Capture Systems
- Oracle Supplier Portal

Guidelines

Integrate with Oracle Procurement

Use guidelines to help you integrate with Oracle Procurement when you set up drop ship for Order Management.

Send Shipping Instructions and Packing Instructions

If you use the predefined drop ship flow, then note these points.

- Order Management sends shipping instructions and packing instructions in one or more attachments on the fulfillment line because Procurement only accepts instructions in an attachment.
- Procurement only processes attachments that include an attachment category of MISC (Miscellaneous) on the fulfillment line. It ignores an attachment that contains any other value.
- An Order Entry Specialist can use the Create Order page in the Order Management work area to set the Category attribute in the Attachments dialog to any value that the drop down for the attribute displays when adding an attachment to an order line. However, Order Management sends a value of MISC to Procurement regardless of the value of Category.
- The user can enter text in the Shipping Instructions attribute and the Packing Instructions attribute in the Shipping Details area of the Create Order page. However, Order Management doesn't send this text to Procurement because Procurement only accepts them in an attachment.
- You must set the attachment category to MISC on the order line of each source order that you import.

Learn about attachment categories and how to set them up. For details, see *Overview of Integrating Attachments in Order Management*.

Calculate Ship Dates

If a supplier sends updates to Procurement, then Procurement might send an update for the scheduled arrival date to Order Management. Order Management updates the scheduled arrival date but doesn't update the scheduled ship date on the fulfillment line.

Order Management updates the scheduled ship date, scheduled arrival date, and shipping method only if you set up Global Order Promising to calculate shipping, such as how to calculate transit time. If you don't set it up, the scheduled dates and shipping method might not contain a value.

This behavior affects fulfillment, including choices that your users make when they override the schedule. For details, see *Schedule Fulfillment Lines Manually*.

If your supplier sends updates to Procurement, then Procurement might send an update for the scheduled arrival date to Order Management. Order Management updates the scheduled arrival date but doesn't update the scheduled ship date on the fulfillment line. You can use the Use Global Order Promising to Recalculate Dates in Order Management feature and Promising will automatically recalculate the dates when you change the promised ship date or promised arrival date in Procurement. For details, see *Use Order Profiles to Control Order Management Behavior*.

Send Your Own Attribute

Use an extensible flexfield to send your own attribute to a descriptive flexfield on the purchase order in Procurement.

Here's an example payload that sends an extensible flexfield named ComplianceDetails to Procurement.

Note

- Use web service Request Fulfill Order Orchestration Task Service to send your payload. For details, see *Overview of Using Extensible Flexfields in Order Management*.
- Set up profile options in Procurement Cloud.
    - Copy the requisition for the purchase order to a descriptive flexfield.
    - Copy the requisition line for the purchase order to a descriptive flexfield.
    - Copy the order header for the purchase order to a descriptive flexfield.

  For details, see *Item Profile Options* and *Overview of Using Flexfields to Integrate Order Management with Other Oracle Applications*.

Use Service Mappings

Use a service mapping to send data to Procurement, such as the supplier price or a note. For details, see *Use a Service Mapping to Integrate Order Management with Procurement*.

Related Topics

- Processing Constraints
- Overview of Integrating Attachments in Order Management
- Overview of Using Extensible Flexfields in Order Management
- Use Order Profiles to Control Order Management Behavior
- Schedule Fulfillment Lines Manually

Set Up Features, Manage Change, and More for Drop Ship

Use guidelines to help you set up drop ship in Order Management.

See a demonstration that illustrates how to set up drop ship. For details, see *Demonstration of Integrating Order Management with Other Oracle Applications*.

Set Up Features

Each feature comes predefined as already available except for Fulfill a Customer Order Through Drop Shipment and Handle Landed Cost Charges for Drop Ship Receipts. Each feature includes a predefined job role.

Feature	Setup Required
Analytics with Supplier Source	No
Automate Financial Flow	Yes
Automatic Change Management on Drop Ship Orders	No
Change Drop Ship Orders from Check Availability Page	No
Create Accounts Payable Invoices for Drop Ship Receipts	Yes
Create ASNs for Drop Ship Receipts	Yes
Create Relationship between Selling Business Unit and Requisition Business Unit	Yes
Fulfill a Customer Order Through Drop Shipment	Yes

Feature	Setup Required
Handle Landed Cost Charges for Drop Ship Receipts	Yes
Order Promising for Drop Ship	Yes
Partial Shipment on Drop Ship Orders	No
Process Accounting for Drop Ship Transactions	No
Report Gross Margins for Drop Ship Orders	No
Report In-transit Inventory for Drop Shipments	No
Use Global Order Promising to Recalculate Dates in Order Management	Yes

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Aggregate Fulfillment Lines

To improve performance, you can aggregate fulfillment lines before you send them to Procurement. The aggregator aggregates change requests for each combination of sales order and purchase order. If you modify a fulfillment line during a drop shipment, then the aggregator sends each combination as a single request to Oracle Procurement. The aggregator sends these details when a time limit expires or when it aggregates a specific number of lines.

If a sales order has a large number of order lines, then the aggregator only aggregates some of the changed lines according to a time limit. This behavior might cause the sales order to become stuck in Order Management because an error happens during the change or the buyer in Procurement makes a change at the same time while the purchase order isn't on hold.

To avoid this problem in a drop ship flow, Order Management comes predefined to use the Aggregate According to Number of Order Lines That Changed profile option, and the aggregator aggregates according to the number of order lines that you change in the sales order's current revision, by default. If you don't want to aggregate according to change, but instead want to aggregate according to time or number of lines, then set this profile option to No, and set a value for the Aggregator Hold Timeout Period in Minutes profile option. For details, see *Use Order Profiles to Control Order Management Behavior*.

For more, see *Actions That You Can Set When Routing Requests to Fulfillment Systems* and *Aggregate Requests That Order Management Sends to Your Fulfillment System*.

Add Drop Ship Details to Reports

1. In Oracle Transactional Business Intelligence, on the Catalog page, click **New > Analysis**.
2. In the Select Subject Area dialog, click **Order Management Fulfillment Lines Real Time**.
3. On the Untitled page, in the Subject Areas tree, expand **Drop Shipment Details**.
4. Drag attributes from the Subject Areas tree and drop them onto the Selected Columns area.

For details, see *Use Reports and Analytics with Order Management*.

Reject an Over-Receipt

If the quantity on the advance shipment notice (ASN) exceeds the ordered quantity, then Oracle Receiving will reject the fulfillment request and display a message requesting the user to modify the quantity regardless of how you set the Over-Receipt Action attribute on the Manage Receiving Parameters page in the Setup and Maintenance work area. If receiving rejects the request, you must correct the quantity on the ASN or in receiving.

For details, see *General Receiving Parameter Options*.

Global Order Promising

Note these points if you use Global Order Promising with a drop shipment.

- Promising considers the UPTZ (User Preference Time Zone) when it calculates dates. For details, see *Time Zone Differences in Order Management*.
- Promising displays the scheduled ship date in the time zone where the supplier or supplier site is located.
- Promising displays the scheduled arrival date in the time zone where the customer site is located.
- Promising searches for and identifies supply according to the local time zone of your organization.

You can use File-Based Data Import to upload the time zones for your supplier sites. You can also specify them in the Global Order Promising work area. If you don't upload or specify them, then Promising uses UTC (Coordinated Universal Time).

Do Other Setups

Here are some other set ups that you might need to do, depending on your drop ship requirements.

- Create a separate requisition and purchase order for each order line.
- If you use a source system as part of your drop ship flow, then you might need to specify a Drop Ship Validation Organization. You can also set up transit times, and a drop ship plan. For details, see *Drop Shipments in Supply Planning*.
- Set up a promising rule. For details, see *Set Up Promising Rules and Sourcing Rules for Order Management*.
- Use Global Order Promising to recalculate dates in Order Management. For details, see *Use Order Profiles to Control Order Management Behavior*.

Related Topics

- Processing Constraints
- Overview of Integrating Attachments in Order Management
- Overview of Using Extensible Flexfields in Order Management
- Use Order Profiles to Control Order Management Behavior
- Time Zone Differences in Order Management

Manage Processing Constraints for Drop Ship

Learn about the predefined processing constraints that Oracle Order Management uses to help keep your drop ship flows running smoothly.

Examine the predefined constraints. Assume you want to get details about the predefined DOO_CREATE_PURCHASE_REQUEST_VALIDATION constraint:

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, search for the value.

Attribute	Value
Constraint Name	DOO_CREATE_PURCHASE_REQUEST_VALIDATION

3. Examine the attribute values. Notice that they prevent you from doing a validation operation on the fulfillment line, for example, when the arrival dates on the fulfillment line don't contain a value.
4. Examine the Message attribute to view the message that this constraint displays at run time.

For details about how constraints work, see *Processing Constraints*.

Predefined Constraints That You Can Enable

These constraints come predefined as disabled. Order Management applies them in your drop ship flows. You can enable them, but you can't modify them. You must not delete them.

Constraint Name	Description
DOO_CREATE_PURCHASE_REQUEST_VALIDATION Display Name: Create Purchase Request Validation	If the Scheduled Arrival Date attribute on the fulfillment line is empty, then this constraint makes sure: • The Requested Arrival Date or the Scheduled Arrival Date on the fulfillment line contains a value. • The ordered quantity on the fulfillment line is greater than zero. • The Record Set contains the values that the Update Purchase Request fulfillment task requires. If you Don't Use Global Order Promising, and if you: • Enable the buyer to manage transportation for the supplier agreement, then Order Management requires the ship date. • Don't enable the buyer to manage transportation for the supplier agreement, then Order Management requires the arrival date. Order Management makes sure the required date contains a value when you submit the sales order.
DOO_UPDATE_PURCHASE_REQUEST_VALIDATION Display Name: Update Purchase Request Validation	If the Scheduled Arrival Date attribute on the fulfillment line is empty, then this constraint makes sure: • The Requested Arrival Date or the Scheduled Arrival Date on the fulfillment line contains a value. • The ordered quantity on the fulfillment line is greater than zero. • The Record Set contains the values that the Update Purchase Request fulfillment task requires. If you Don't Use Global Order Promising, and if you: • Enable the buyer to manage transportation for the supplier agreement, then Order Management requires the ship date.

	• Don't enable the buyer to manage transportation for the supplier agreement, then Order Management requires the arrival date. Order Management makes sure the required date contains a value when you submit the sales order.
DOO_DS_FULFILLMENT_LINE_CANCEL Display Name: Cancel Fulfillment Line That Drop Ships DOO_DS_FULFILLMENT_LINE_UPDATE Display Name: Update Fulfillment Line That Drop Ships	Prevents you from updating or canceling the fulfillment line when the purchase request in Oracle Procurement is at the Purchase Order stage. For example, if you attempt to update, then Order Management displays a message. Order management can't update the fulfillment line because the purchase request in the procurement application is at the Purchase Order stage. Order Management applies these constraints until the received quantity or invoiced quantity equals the quantity on the order line. These constraints help to keep the sales order in Order Management and the purchase order in Oracle Procurement synchronized with each other. If you disable these constraints but Procurement can't accept an update or cancel because the supplier rejects the change, then you must create and enable your own constraint that prevents the user from updating or canceling the line. If you must allow your users to cancel a fulfillment line that's at the Awaiting Shipping status, and if DOO_DS_FULFILLMENT_LINE_CANCEL is enabled, then tell your users to cancel the purchase order, wait until Order Management receives the updated status for the purchase order, revise the sales order, cancel the order line, and then submit the revision. If you disable DOO_DS_FULFILLMENT_LINE_CANCEL, and if you haven't created your own constraint that prevents the user from canceling, then the user can use the Order Management work area to cancel a fulfillment line that's at the Awaiting Shipping status in a drop shipment. Procurement will also cancel the purchase order and the purchase requisition for the fulfillment line.

Predefined Constraints That You Can’t Disable

These constraints come predefined as enabled. Order Management applies them in your drop ship flows. You can’t disable them or modify them. They help to make sure you don’t inadvertently introduce errors into your drop ship flow.

Attribute	Value
DOO_FULFILLMENT_LINE Display Name: Fulfillment Line Update	If the order line’s status is Requisition Created, then you can’t revise the line, but you can cancel the line.
DOO_FULFILLMENT_LINE_CANCEL Display Name: Cancel Fulfillment Line	If the order line’s status is Requisition Created, and if it’s part of a configured item or shipment set, then you can’t cancel the line. Instead, you must return the line in Procurement, and then reschedule or cancel it in Order Management.
DOO_SUPPLIER_SITE_CHANGE Display Name: Change Supplier Site	If Procurement splits a fulfillment line during a change, then you can’t modify the supplier site on that line.
DOO_SUPPLIER_CHANGE Display Name: Change Supplier	If Procurement splits a fulfillment line during a change, then you can’t modify the supplier on that line.
DOO_FulfillmentLine_DropShip_PO_Received_Cancel	Prevents you from cancelling a fulfillment line when any of these conditions exist:

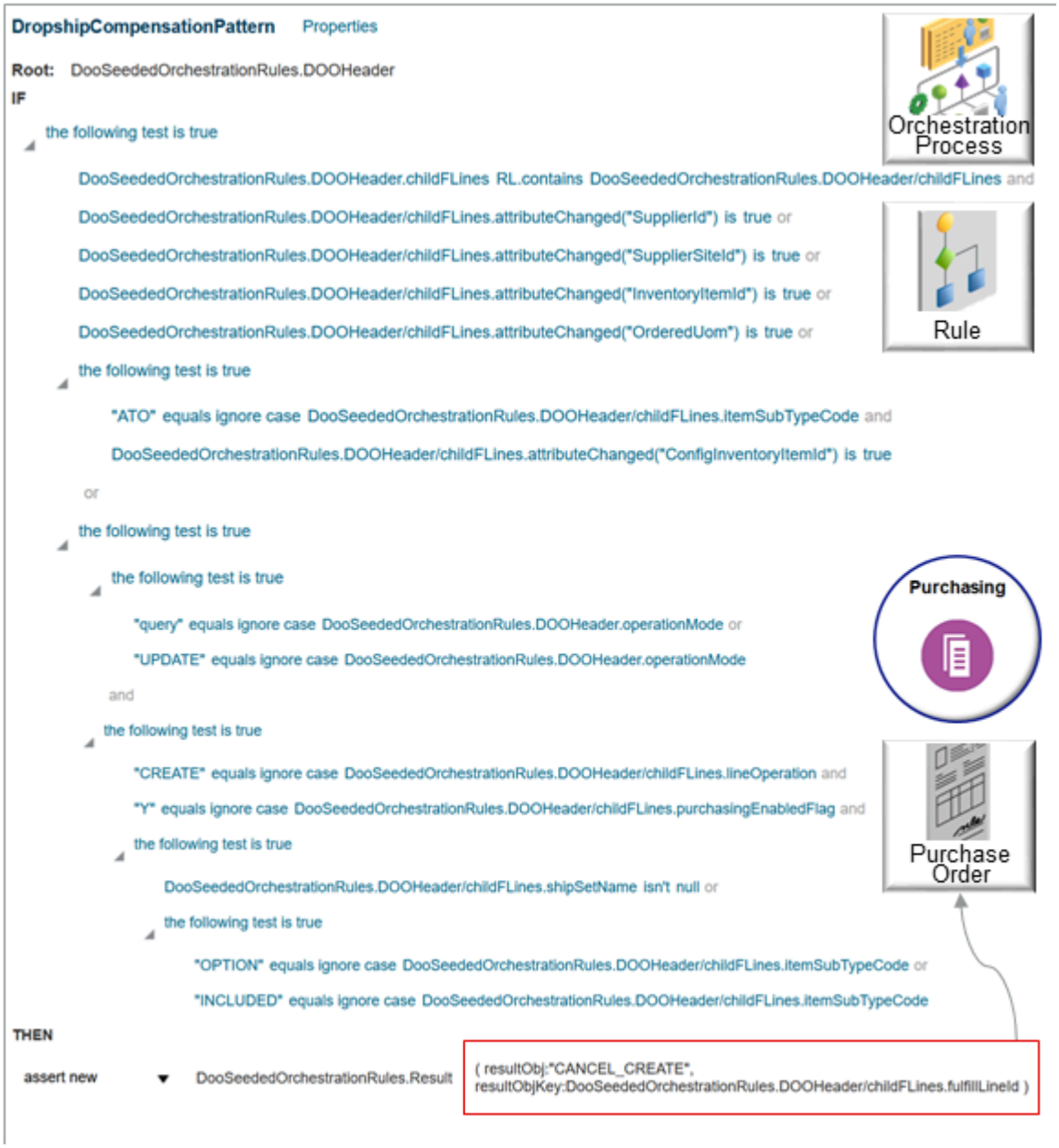
Attribute	Value
Display Name: Cancel Fulfillment Line When Purchase Order Isn't Available	• The purchase order for the line is frozen or is on hold. • The quantity that Order Management shipped doesn't match the quantity that Procurement received or billed.
DOO_FulfillmentLine_DropShip_PO_ReceiveD_Update Display Name:Update Fulfillment Line When Purchase Order Isn't Available	Prevents you from revising a fulfillment line when any of these conditions exist: • The purchase order for the line is frozen or is on hold. • The quantity received or the quantity billed doesn't equal the quantity shipped. • You ship part of a line and your customer returns some or all of the quantity on the shipped line. If the quantity that Order Management shipped doesn't match the quantity that Procurement received or billed, then you can't change the value for: • Any attribute that's eligible for a constraint. • An attachment, extensible flexfield, tax determinant, payment term, or accounting detail. • An attribute in Order Management that isn't mapped to Procurement, but that's eligible for a constraint. This condition applies until the quantity that Order Management shipped matches the quantity that Procurement received or billed. Order Management prevents these changes at run time on order lines that it has shipped and on lines that it hasn't shipped. You can modify the value of an attribute that isn't eligible for a constraint regardless of whether it's mapped between Order Management and Procurement.
DOO_UPDATE_EFF_FULFILLMENTLINE_DROPSHIP Display Name: Submit Changes to Extensible Flexfield on Fulfillment Line During Requisition	Prevents you from modifying an extensible flexfield on a fulfillment line when the line is at the requisition stage in Procurement.

Use Compensation Patterns to Manage Purchase Orders

You can add a compensation pattern to your orchestration process to manage purchase orders when you don't use Oracle Global Order Promising.

Assume an order line contains an item that you drop ship and you already submitted the line to fulfillment. You revise the sales order, the line's status is Awaiting Shipping, you change the supplier or the supplier site, and then submit the revision. You now need to cancel the old purchase order and create a new one that has your revised supplier or supplier site.

If you use Global Order Promising, then it will automatically send a request to Oracle Purchasing to manage the purchase order. If you don't use Global Order Promising, then you can use this compensation pattern to send the request instead:



Here's what you need to do:

1. Use the Manage Orchestration Process Definitions task in the Setup and Maintenance work area to open your orchestration process for editing.
2. Add the compensation pattern to the Create Purchase Request step. If your orchestration process doesn't have a step that creates a purchase request, then add one. For an example that has a Create Purchase Request step, examine the predefined OrderFulfillmentGenericProcess orchestration process.
3. Validate, release, and deploy your orchestration process.

If all the tests evaluate to true at run time, then the assert new action creates a new CANCEL_CREATE object, and the orchestration process sends that object to Oracle Purchasing with a request to cancel the old purchase and create a new one that has your revised supplier or supplier site. The orchestration process also sends other data that Purchasing needs to manage the purchase order, such as the SupplierId, SupplierSiteId, and so on.

To simplify your life, we've created an orchestration process named OrderFulfillmentGenericProcessWithoutSchedule for you that already has this compensation pattern. You can upload it to your Order Management environment. Here's how:

1. Go to *Technical Reference for Order Management (Doc ID 2051639.1)*.
2. Download the Payloads and Files attachment.
3. Open the file that you downloaded, locate the Implementation Project-40_1_20230601_2143.zip file, then save that zip file to your computer.
4. Upload the Implementation Project-40_1_20230601_2143.zip file to your Order Management environment. For details, see *Migrate Orchestration Processes in Order Management*.
5. Use the Manage Orchestration Process Definitions task to search for and open OrderFulfillmentGenericProcessWithoutSchedule.
6. Examine the compensation pattern on the Create Purchase Request step.

For details about compensation patterns, including guidelines and how to set them up, see *Compensate Sales Orders That Change*.

Here's a text version of the compensation pattern.

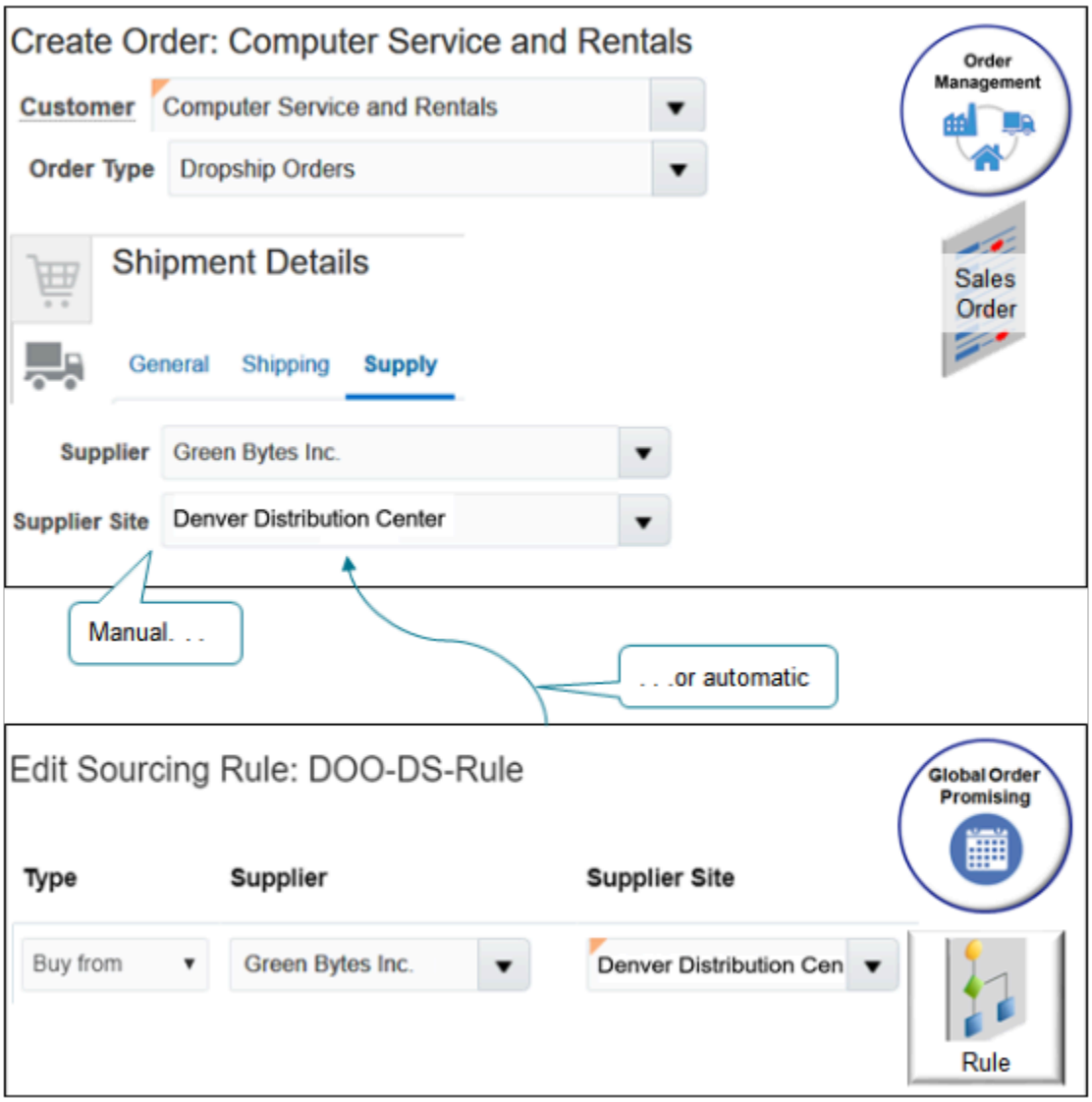
```
Root: DooSeededOrchestrationRules.DOOHeader
 IF the following test is true
 DooSeededOrchestrationRules.DOOHeader.childFLines RL.contains DooSeededOrchestrationRules.DOOHeader/
childFLines and
 DooSeededOrchestrationRules.DOOHeader/childFLines.attributeChanged("SupplierId") is true or
 DooSeededOrchestrationRules.DOOHeader/childFLines.attributeChanged("SupplierSiteId") is true or
 DooSeededOrchestrationRules.DOOHeader/childFLines.attributeChanged("InventoryItemId") is true or
 DooSeededOrchestrationRules.DOOHeader/childFLines.attributeChanged("OrderedUom")is true or
 the following test is true
 "ATO" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.itemSubTypeCode and
 DooSeededOrchestrationRules.DOOHeader/childFLines.attributeChanged("ConfigInventoryItemId") is tr
 or
 the following test is true
 the following test is true
 "query" equals ignore case DooSeededOrchestrationRules.DOOHeader.operationMode or
 "UPDATE" equals ignore case DooSeededOrchestrationRules.DOOHeader.operationMode
 and
 the following test is true
 "CREATE" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.lineOperation and
 "Y" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.purchasingEnabledFlag and
 the following test is true
 DooSeededOrchestrationRules.DOOHeader/childFLines.shipSetName isn't null or
 the following test is true
 "OPTION" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.itemSubTypeCode or
 "INCLUDED" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.itemSubTypeCode
THEN
 assert new DooSeededOrchestrationRules.Result ( resultObj:"CANCEL_CREATE",
 resultObjKey:DooSeededOrchestrationRules.DOOHeader/childFLines.fulfillLineId )
```

Manage Sales Orders and Purchase Orders

Use guidelines to help you manage sales orders and purchase orders in the drop ship flow that you use with Order Management.

Create Sales Orders

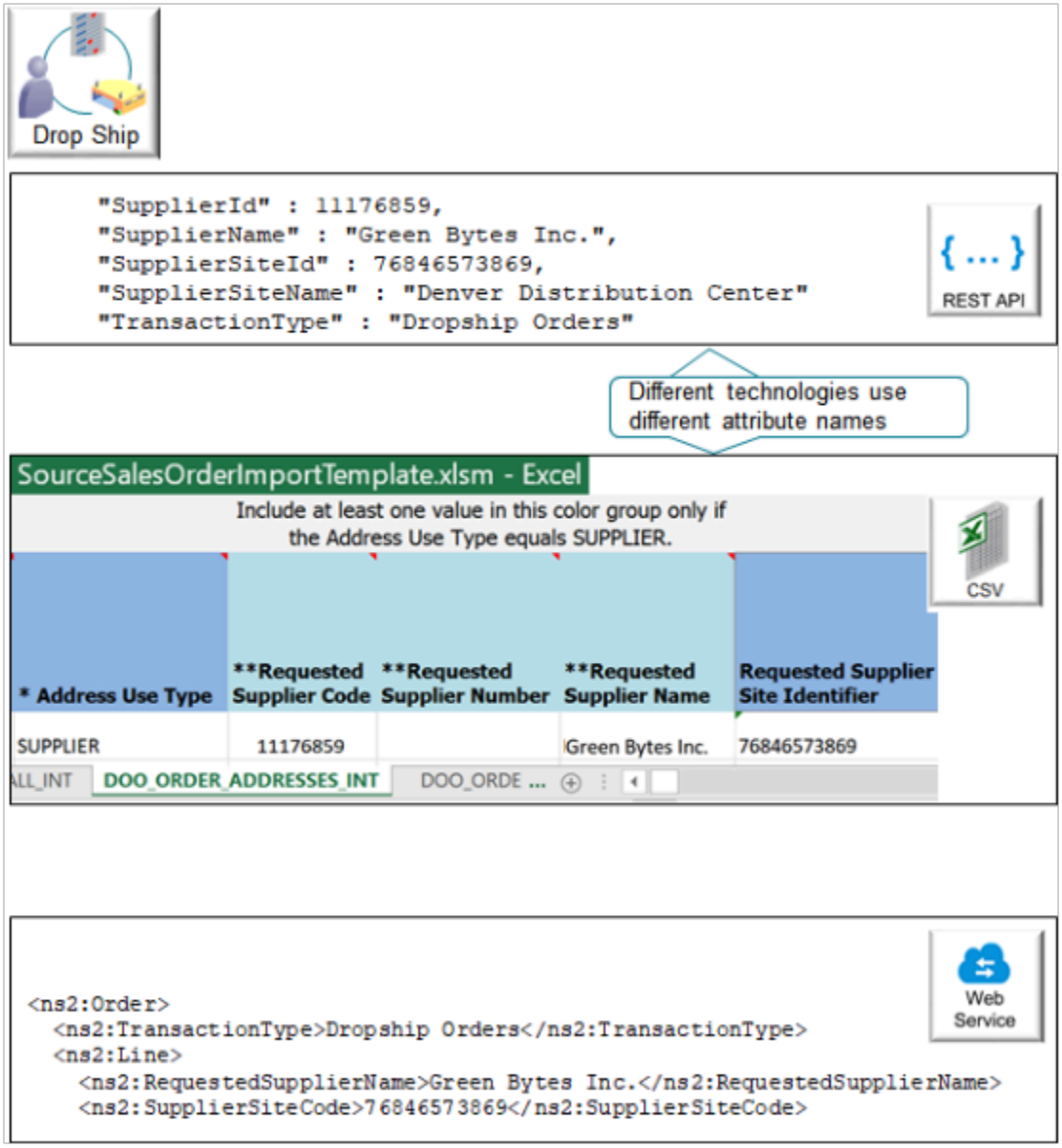
You can use different technologies to provide supplier details when you create the sales order, depending on your implementation requirements.



Note

- Specify a value in the Supplier attribute to implicitly start the drop ship flow.
- Manually set Supplier and Supplier Site attributes on the sales order, or create a sourcing rule that does it automatically.

Use REST API, file-based data import, or a web service to create the order.



Different technologies use different attribute names to represent the same data.

Technology	Description
Sales order in the Order Management work area	• As an option, set the Order Type attribute to Drop Ship Orders on the order header. • Set the Supplier attribute and Supplier Site attribute on the Supply tab of the order line. If you don't specify a value for supplier site, then Oracle Procurement sets it according to rules in Procurement.
Sourcing rule in the Order Promising work area	Use the Manage Sourcing Rules page in the Order Promising work area. Set the attributes. • Supplier • Supplier Site

Technology	Description

REST API	Use the salesOrdersforOrderHub REST resource. In the request payload, provide.
	• SupplierId
	• SupplierName (optional)
	• SupplierSiteId
	• SupplierSiteName (optional)
	• TransactionType or TransactionTypeCode
	For example:
	`{`
	`"SourceTransactionNumber":"87956",`
	`"SourceTransactionSystem":"GPR",`
	`"SourceTransactionId":"87956",`
	`"BusinessUnitName":"Vision Operations",`
	`"BuyingPartyName":"Computer Service and Rentals",`
	`"BuyingPartyContactName":"Brian Smith",`
	`"TransactionType":"Standard Orders",`
	`"RequestedShipDate":"2019-10-19T20:49:12+00:00",`
	`.`
	`.`
	`.`
	`"SupplierId" : 11176859,`
	`"SupplierName" : "Green Bytes Inc.",`
	`"SupplierSiteId" : 76846573869,`
	`"SupplierSiteName" : "Denver Distribution Center"`
	`"TransactionType" : "Dropship Orders"`
	`"TransactionTypeCode" : "STD_DS"`
	`.`
	`.`
	`.`
	`}`
	For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

File-Based Data Import	Use the SourceSalesOrderImportTemplate.xlsm file.
	On the DOO_ORDER_HEADERS_ALL_INT worksheet, set the Transaction Type Code to STD_DS.
	On the DOO_ORDER_ADDRESSES_INT worksheet.
	• Set the Address Use Type attribute to SUPPLIER.
	• Specify a value for at least one of these attributes.
	  ○ Requested Supplier Code
	  ○ Requested Supplier Number
	  ○ Requested Supplier Name
	• Specify a value in the Requested Supplier Site Identifier attribute.

Web Service	• Specify TransactionType or TransactionTypeCode on the order header in a web service payload. • Specify RequestedSupplierName and SupplierSiteCode on the order line. • Use the Create Order Operation of the OrderImportService web service.

Technology	Description

	For example:
	
	`<ns2:Order>`
	`.`
	`.`
	`.`
	
	`<ns2:TransactionType>Dropship Orders</ns2:TransactionType>`
	`.`
	`.`
	`.`
	`<ns2:Line>`
	`.`
	`.`
	`.`
	`<ns2:RequestedSupplierName>Green Bytes Inc.</ns2:RequestedSupplierName>`
	`<ns2:SupplierSiteCode>76846573869</ns2:SupplierSiteCode>`
	`.`
	`.`
	`.`

Note

- If you don't specify the supplier and supplier site in a sourcing rule, then you must include the Schedule Ship Date on the order line in the payload or FBDI template.
- To verify the values that you must use for TransactionType or TransactionTypeCode, go to the Setup and Maintenance work area, select the Order Management offering, open the Manage Order Lookups task, then search for the ORA_DOO_ORDER_TYPES lookup type. Use a value from the:
  - Meaning column for the TransactionType attribute
  - Lookup Code column for the TransactionTypeCode attribute

Modify Purchase Orders

Here's what you can and can't do.

Modification	Description
Ship a pick-to-order item, or a line that's part of a kit or shipment set.	• You can revise the promised ship date, promised delivery date, or shipment method. You can cancel the order or split the schedule for a line that has a pick-to-order item or has an item that's part of shipment set or kit in a purchase order. You can do these actions even when the supplier shipped part of the line. • You can change the shipment date, delivery date, or shipment method only if the new value that you provide on the purchase order line is the same for all components that are part of the pick-to-order item, all items that are part of a kit, or all items that are part of a shipment set. • if the item is a pick-to-order configured item, a kit, or if it's part of a shipment set, then you can't cancel the purchase order schedule or purchase order line. • Don't use a shipment set to group sales order lines into a single purchase order. Procurement doesn't support shipment sets for this usage.
Modify purchase requisition.	The buyer must not: • Modify or split a requisition line.

Modification	Description
	• Modify a pick-to-order or assemble-to-order configured item. • Modify a kit.
Release a hold.	If Order Management applies a hold on a sales order, then Procurement also applies a hold on the corresponding purchase order. If the buyer has the PO_RELEASE_PO_REQ_INIT_HOLD_FREEZE_PRIV (Release Purchase Order Hold and Freeze Initiated by Requester) privilege, then the buyer can use the Purchase Orders work area to remove the hold.
Modify other attributes	The buyer can change values in attributes that don't affect the drop ship flow.

Here's what you can and can't do regarding combining and splitting.

Modification	Description
Split the schedule.	The buyer can split the schedule, but don't split the schedule and also change the supplier at the same time.
Combine sales orders into one purchase order.	If the flow creates one purchase order for requisition lines that reference more than one sales order, and if you revise these sales orders, then the flow places a hold on one of the sales orders and revises the purchase order. It doesn't revise the other sales orders and they fail. To fix this problem, go to the Order Management work area, delete the sales order revisions that failed, wait for Order Management to finish processing the sales order it placed on hold, then revise the other sales orders. You can only combine sales orders for the same customer or for the same supplier site. You can't consolidate sales orders across customers or across supplier sites.
Combine sales order lines into purchase orders.	Manually combine sales order lines into one or more purchase orders. You can create one purchase order for each order line, resulting in several purchase orders for one sales order.
Assign a supplier.	If you don't assign a new supplier on the requisition, then don't split the schedule on the draft purchase order.

Here's what you can and can't do regarding quantity.

Modification	Description
Modify the quantity.	If you split the purchase order schedule across more than one delivery date, then make sure the total quantity across your split schedules equals the ordered quantity on the fulfillment line. If shipping has received the ASN, then you can't modify the quantity.
Set the quantity for advance shipment notice.	Make sure you use a single order line to specify the quantity for an advance shipment notice. Don't split the quantity across more than one line.
Modify the receipt quantity.	The drop ship flow creates a receipt when it receives an invoice from accounts payable or an advance shipment notice. The flow then moves to the next orchestration process step in Order Management. Order Management won't display a subsequent change you make to the receipt quantity.

Modification	Description

Create Purchase Orders

You must group all lines that are part of a configured item, kit, or shipment set into a single purchase order. You can't spread them across different purchase orders.

Assume your sales orders contains the PTO54222 phone, the phone is a pick-to-order item, and it has this hierarchy.

```
PTO54222 Phone
 Case Option Item
 Charger Included Item
 USB Cable Included Item
```

You must include the case, charger, and cable in the same purchase order.

For another example, assume your sales order has these order lines.

Order Line	Item	Shipment Set
1	AS54888, Standard Desktop Computer	S1
2	AS9000, Standard Laptop Computer	S1
3	PTO54222, Phone	S1

All of these lines are in the same shipment set, so you must include the AS54888, AS9000, and the PTO54222 in the same purchase order.

For details about how to create a purchase order, see *Create Purchase Order*.

Related Topics

- Processing Constraints
- Overview of Integrating Attachments in Order Management
- Overview of Using Extensible Flexfields in Order Management
- Use Order Profiles to Control Order Management Behavior
- Schedule Fulfillment Lines Manually

Procedures

Set Up Drop Ship in Order Management

Set up Order Management so it supports your drop ship flow.

Summary of the Set Up

1. Set up Oracle Applications.
2. Enable features.
3. Manage items.

4. Manage sourcing rules.
5. Manage financial flows. For details, see *Set Up Financial Flows for Drop Ship*.
6. Specify preparer for procurement.
7. Manage suppliers and supplier sites.
8. Manage agreements.
9. Manage orchestration processes.
10. Manage agreements, orchestration processes, and test.

Note

- Each of these set ups are required except for Manage Agreements and Manage Orchestration Processes, which are optional.
- In this example, you enable the AS54888 Standard Desktop Computer for drop ship.
- This topic uses example values. You might need different values, depending on your business requirements.

Set Up Oracle Applications

Set up various Oracle applications so they support drop ship. For example, the Order Management offering references Global Order Promising to collect supply data from more than one source.

Here are the tasks you do to get started with setting up drop ship.

1. Define Blanket Agreement and ASL in Procurement. Sign in with a privilege that you can use to administer Oracle Procurement.
2. Create Drop-Ship Validation Org in Inventory Management. Sign in with a privilege that you can use to administer Inventory Management.
3. Do tasks in Global Order Promising.
    - Sign in with a privilege that you can use to administer Global Order Promising.
    - **Manage Data Collections**.
    - **Define Item Processing Lead Times**.
    - **Manage Sourcing Rules and Manage ATP Rules**. Set up sourcing rules and ATP rules so they support drop ship. For details, see *Assignments and Promising Rules*.
4. Do tasks in the Manufacturing and Supply Chain Materials Management offering.
    - Manage Trade Operations
    - Define Shipping Network
    - Define Transit Lead Times
    - Manage Assignment Sets
    - Maintain Supply Network Model

For details, see *Oracle Fusion Cloud SCM Common Configuration Overview*.

Enable Features

1. Sign into Order Management with administrative privileges.
2. Go to the Setup and Maintenance work area, then select the Order Management offering.

   For details, see *Opt Into Features in Order Management*.
3. Click **Actions > Change Feature Selection**.
4. On the Edit Features page, in the Drop Ship row, add a check mark to **Enable**, then click **Done**.

Enabling this feature lets you access various attributes and to do the setup tasks you need to implement drop ship, such as in Financial Orchestration.

5. Click **Actions > Go to Offerings**.
6. In the Setup and Maintenance work area, go to the Procurement offering.
7. Click **Actions > Change Feature Selection**.
8. On the Edit Features page, in the Customer Sales Order Fulfillment row, add a check mark to **Enable**, then click **Done**.

   This feature allows Oracle Procurement to accept purchase requests for sales orders that a drop ship supplier fulfills, and to display some of the attributes that you use to set up drop ship.

Manage Items

Set up items so they can participate in drop ship.

1. Sign in with a privilege that you can use to administer Product Information Management.
2. Go to the Product Information Management work area, then click **Tasks > Manage Items**.
3. On the Manage Items page, search for the item you want to drop ship, such as Standard Desktop Computer.
4. In the search results, in the Item column for the item you must manage, click the **link**.

   For example, in the Standard Desktop Computer row, click **AS54888**.
5. On the Edit Item page, click **Specifications**, then click **Purchasing**.
6. In the Item Organization: Purchasing area, set the attribute.

Attribute	Description
Purchasable	Set to Yes. Allow this item to participate in a drop shipment.

7. If a blanket purchase agreement doesn't exist for the item, then, in the Pricing area, enter a number in the List Price attribute.
8. Click **Save**.

   Note

   - You specify the item as purchasable in the organization that's responsible for purchasing the item.
   - The Product Information Management work area comes predefined to set the Purchasable attribute to Yes, so you modify it only if you previously set it No.
   - Learn how to make a large number of items purchasable. For details, see *Implementing Product Management*.

Manage Sourcing Rules

Specify a sourcing rule that already includes details for the supplier and the supplier site. Global Order Promising evaluates the sourcing rule and considers the supplier calendar, supplier capacity, and supplier lead times when it promises the order.

1. Sign in with a privilege that you can use to administer Order Promising.
2. Go to the Global Order Promising work area, then click **Tasks > Manage Sourcing Rules**.
3. On the Manage Sourcing Rules page, search for the sourcing rule you must modify, such as DOO-DS-Rule.
4. In the search results, click the sourcing rule, then click **Actions > Edit**.

5. On the Edit Sourcing Rule page, verify the value.

Attribute	Description
Organization Assignment Type	Make sure the type is Global. If it isn't Global, then you can't use this sourcing rule.

6. In the Effective Start Date area, to add a sourcing rule for your drop ship flow, click **Actions > Add**, then set the values.

Attribute	Description
Organization	Leave empty. The Organization is typically the warehouse that stores inventory for a flow that doesn't include a drop ship.
Type	Select Buy From. Global Order Promising interprets Buy From to indicate that this value must come from a drop ship supplier.
Supplier	Select the supplier who will drop ship the item.
Supplier Site	Select the site that the supplier uses to store the drop ship item. The supplier ships the item from this site.

7. Click **Save and Close**.
8. Click **View Assignment Sets**.
9. In the Assignment Sets dialog, select an **assignment set**, such as AYY-OP-ASET, then click **Done**.
10. On the Edit Assignment Set page, in the Sourcing Assignments area, click **Actions > Add Row**, then assign an item to the sourcing rule.

Attribute	Description
Assignment Level	Set to Item. This setting assigns the sourcing rule to the item.
Item	Select the item that the supplier you specified in step 6 supplies. For this example, select AS54888.
Sourcing Type	Set to Sourcing Rule. This value associates the sourcing rule with the assignment set so Order Management can use the rule to assign the item.
Sourcing Rule or Bill of Distribution	Set to Drop Ship.

Sourcing Hierarchy

Click **Sourcing Hierarchy** on the Edit Assignment Set page to see how Promising decides to assign the sourcing rule. Here's the hierarchy that it uses for a drop shipment.

Rank	Description
1	Item, Customer or customer site
2	Item, Customer
3	Item, Demand Class
4	Item, Region
5	Category, Customer or customer site
6	Category, Customer
7	Category, Demand Class
8	Item
9	Category, Region
10	Category
11	Customer or customer site
12	Customer
13	Demand Class
14	Region
15	Global

1 is the highest ranking, and 15 is the lowest ranking. Promising assigns from 1 through 15, sequentially.

- Promising assigns the sourcing rule to the item's customer or customer site.
- If Promising can't assign the sourcing rule to the item's customer or customer site, then it assigns the rule to the item's customer.
- If Promising can't assign the sourcing rule to the item's customer, then it assigns the rule to the item's demand class.
- And so on.

Attributes for the Supplier

You can also specify other attributes for the supplier.

Attribute	Description
Supplier Calendar	Specify the working days for the supplier. • Modify the calendar for each supplier site. • The calendar you specify can be different from the calendar that the Supplier Site uses. • Global Order Promising considers only working days when it calculates and incorporates lead times.
Supplier Capacity	Specify the supplier capacity according to item, supplier, and supplier site. Order Promising measures the supplier capacity that exists on the arrival date.
Supplier Lead Times	You specify and collect the lead times for item processing on the item master in Product Information Management. You can specify a lead time for each supplier in Global Order Promising.

Specify Preparer for Procurement

Specify an Order Management user who the buyer can contact to help resolve a problem that might happen with a fulfillment line that involves a drop ship supplier.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Parameters
2. On the Manage Order Management Parameters page, click **Preparer for Procurement**.
3. In the Preparer for Procurement list, add a preparer, then click **Save and Close**.

    Order Management doesn't use the value for any other attribute as the preparer. For example, it doesn't use the Created By attribute on the order header or order line. It uses only the value you specify in Preparer for Procurement. For details, see *Manage Order Management Parameters*.

Manage Suppliers and Supplier Sites

Make sure the supplier data that you're providing is valid.

For example, assume your supplier's name is Penny Pack Systems:

1. Go to the Suppliers work area, then click **Manage Suppliers**.
2. On the Manage Suppliers page, search for Penny Pack Systems. Make sure the supplier exists and is active.
3. Make sure you have the privileges that you need to manage Accounts Receivable, go to the Accounts Receivable work area, then click **Tasks > Manage Customers**.

4. On the Manage Customers page, set the values, then click **Search**.

Attribute	Value
Customer Type	Organization Your supplier must be an organization.
Organization Name	Penny Pack Systems

5. Verify that the search results contains Penny Pack Systems. If the results don't include your supplier, then click **Actions > Create** to add it.
6. In the Sites section, make sure the supplier site exists and that the Purpose attribute contains Purchasing or contains Sourcing.
7. If the Purpose doesn't contain Purchasing or Sourcing, then edit the site and set these values.

Attribute	Value
Procurement BU	Vision Operations You must use the same business unit that the supplier uses. For this example, assume it's Vision Operations.
Status	Active The site must be active.
Purchasing	Contains a check mark.

For background, see the Supplier Sites and Supplier Site Assignments topic in *Using Procurement*.

Manage Agreements, Orchestration Processes, and Test

Manage Agreements

Oracle Purchasing allows your buyer to create a blanket purchase agreement for items it will drop ship from the supplier. You must define an agreement for each supplier and supplier site, and associate one or more items with the agreement. You do this work when you set up purchasing.

This topic describes how to modify the setup so it supports drop ship. Purchasing uses the prices that the agreement specifies to set default values in the purchase documents. For details, see the Blanket Purchase Agreement Lines topic in *Using Procurement*.

You manage agreements differently for a drop ship that includes a configured item. For details, see *Set Up Drop Ship for Configured Items*.

Manage agreements.

1. Sign in with a privilege that lets you access the Purchasing work area.

2. Go to the Purchasing work area.
3. Click **Tasks > Manage Agreements**.
4. On the Manage Agreements page, select a value in the Procurement BU attribute, such as Vision Operations, then click **Search**.
5. In the search results, in a row that includes an agreement with a supplier who will drop ship your item, in the Agreement column, click the **link**.
6. On the Agreement page, click **Actions > Edit**.
7. On the Edit Document page, click **Controls**, then set the values.

Attribute	Description
Automatically Generate Orders	Set to a value. - **Contains a check mark**. Purchasing will automatically convert each requisition that it sources from the blanket purchase agreement. It will convert each requisition to a purchase order. - **Doesn't contain a check mark**. Your buyer must do the conversion manually in the Purchasing work area. Purchasing examines eligibility according to the requisition to purchase order even if Automatically Generate Orders doesn't contain a check mark. For example, it makes sure each attribute is valid. - Buyer - Supplier - Supplier site - Source agreement - Item is purchasable in the inventory organization of the procurement business unit
Automatically Submit for Approval	Make sure this option contains a check mark. If it doesn't, then Order Management will create the order with an incomplete status, and the buyer must manually submit it for approval.
Use Customer Sales Order	Set the value. - **Contains a check mark**. Purchasing will group requisition lines that reference the same sales order number. It will group them on a single purchase order. - **Doesn't contain a check mark**. Purchasing won't group.

8. Click **Save > Save and Close**.
9. Repeat steps 5 through 8 for each supplier who participates in your drop ship flow.

Manage Orchestration Processes

Order Management comes predefined with the DOO_OrderFulfillmentGenericProcess orchestration process.

It contains branches that run under conditions.

- If the shipment is a drop shipment, then run the Create Shipment Request branch.
- If you enable the item in the inventory organization for back-to-back shipping, then run the back-to-back branch.
- If the first two conditions are false, then run the Create Reservation branch.

Use this orchestration process as the default process assignment in your Assign and Launch rule. You can also create your own orchestration process that meets your business requirements. For details, see *Set Up Orchestration Processes*.

Examine the predefined orchestration process.

1. Sign in with the privileges that you need to administer Order Management.
2. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
3. On the Manage Orchestration Process Definitions page, search for DOO_OrderFulfillmentGenericProcess.
4. Examine the steps and branches.
5. If necessary, make a copy of this process, then modify it so it meets your business requirements, or create a new orchestration process and use DOO_OrderFulfillmentGenericProcess as a starting point.

Test Your Set Up

Create a sales order that uses drop ship.

Related Topics

- How Drop Ship Works in Order Management
- Set Up Drop Ship for Configured Items
- Set Up Financial Flows for Drop Ship
- Assignments and Promising Rules
- Supplier Sites and Supplier Site Assignments

Set Up Financial Flows for Drop Ship

Set up your financial flow to create cost accounting distributions that track costs and ownership liability each time a transfer happens between parties, including the supplier, one or more organizations, and the customer.

Oracle Financial Orchestration controls the change in ownership for each item that it processes during a drop ship flow. For example, to transfer ownership from the selling business unit to the requisition business unit. It also creates an intercompany invoice for each internal ownership transfer, when necessary.

You can specify more than one requisition business unit to manage and own more than one transaction that requests the item. You can also specify the selling business unit in the legal entity that sells the item.

Here's how it works.

1. Receive events. Financial Orchestration captures the physical supply chain event each time one happens in the drop ship flow. For example, when the supplier sends the advance shipment notice to indicate that they shipped the item.
2. Identify the financial flow to run. Financial Orchestration uses your set up details to identify the financial flow to use.
    - Purchase order for the drop ship
    - Sales order details that it gets from source documents
    - Selling business unit and buying business unit
    - Financial orchestration qualifiers

- Priority of the financial orchestration flow

3. Run the financial flow.

The flows that Financial Orchestration runs depends on the number of business units that are involved.

Number of Business Units	Description
One	The selling business unit and the requisition business unit are the same unit, and Financial Orchestration uses only one financial flow for the drop ship.
More than one	Financial Orchestration might run through more than one business unit that involves procurement financial flows and shipment financial flows.

Set it Up

1. Make sure you have the privileges that you need to administer Order Management.
2. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Manufacturing and Supply Chain Materials Management
   - Functional Area: Supply Chain Financial Flows
   - Task: Manage Drop Ship Financial Flows
3. On the Manage Drop Ship Financial Flows page, in the search results, click **Actions > Create**.
4. On the Create Drop Ship Financial Flow page, set the values.

Attribute	Description
Name	Drop Ship Flow for Vision Operations You can use any text.
Priority	1

5. Set the attribute.

Attribute	Description
Supplier Ownership Change Event	Specify when to start the ownership change. - **ASN From Supplier**. Change ownership when the supplier sends an advance shipment notice (ASN) to the financial flow for the drop ship order. - **AP Invoice Match**. Change ownership when the supplier sends an invoice to the financial flow for the drop ship order. For details, see *Indicate an Ownership Change During Drop Ship*.

6. In the Selling BU to Receiving BU Relationships area, click **Actions > Add Row**, then specify the relationship.

For details, see *Specify Business Units for Drop Ship Flows*.

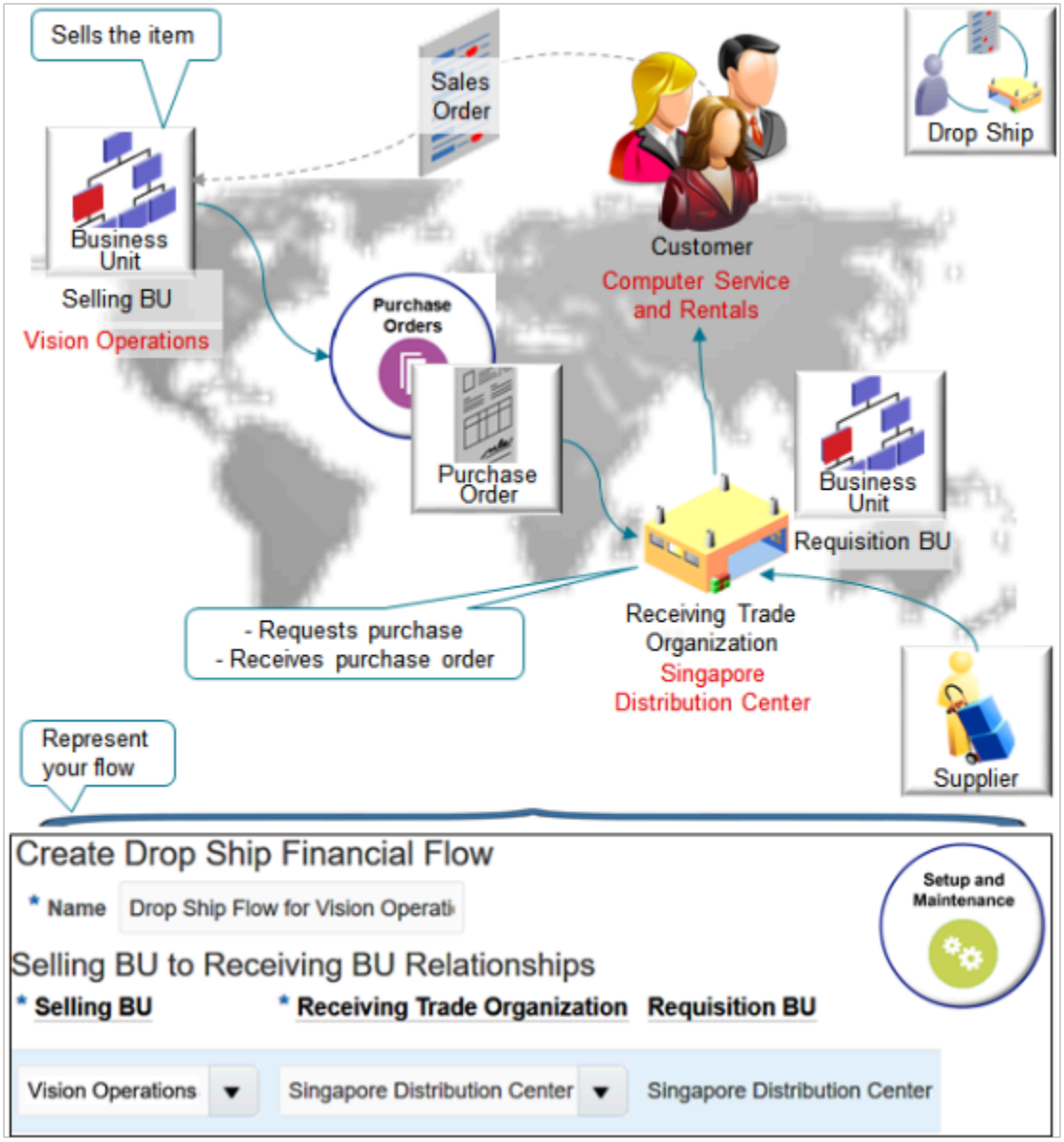
7. Repeat these steps for each selling business unit that your enterprise contains.
8. Click **Save and Close**.

Related Topics

- Set Up Drop Ship in Order Management
- Specify Business Units for Drop Ship Flows
- Indicate an Ownership Change During Drop Ship
- Financial Orchestration Flows

Specify Business Units for Drop Ship Flows

Specify the relationship between the selling business unit, requisition business unit, supplier, and customer when you set up a financial flow that includes a drop ship.



Note

- Red text indicates example values.
- Vision Operations is a legal entity. It contains a selling business unit that takes sales orders directly from your customer in its North American call center.
- Singapore Distribution Center.
    - Is the receiving trade organization. It receives the item from your supplier. The receiving trade organization also owns the requisition and receives the purchase order on behalf of the selling business unit.
    - Is the requisition business unit.
    - Supplies the item to customer Computer Service and Rentals, which is located in China.
- The supplier is a separate company and is also a legal entity.

The Create Drop Ship Financial Flow page sets the Receiving BU attribute and the Receiving Legal Entity attribute according to the value you set in the Receiving Trade Organization attribute.

Assume you set Receiving Trade Organization to Vision Operations, and you set up the application to create a requisition in the Vision Operations inventory organization for each sales order you drop ship that Vision Operations creates. Here are the attributes that will contain a value of Vision Operations.

- Selling BU
- Selling Legal Entity
- Receiving Trade Organization
- Receiving BU
- Receiving Legal Entity
- Requisition BU

Order Management gets the value for the Requisition Organization attribute and the Requisition BU attribute from the purchase requisition it creates for each drop ship order that involves a supplier, and that requires a requisition organization.

Specify the Receiving Trade Organization

Specify the Receiving Trade Organization attribute on the Create Drop Ship Financial Flow page. Select the organization that does these tasks.

- Places the requisition for the goods
- Receives the purchase order for the drop ship
- Does the receipt accounting and shipment accounting for the drop ship

Set the Selling Business Unit and the Receiving Business Unit

Use Financial Orchestration to set up and run the financial flow according to the ownership transfer that happens between the parties that are involved in the drop ship flow. Set up the relationship that exists between the selling business unit and the receiving business unit. Financial Orchestration uses this relationship when it creates a purchase requisition for the drop ship.

If you set the value for the Selling BU attribute to a value that's different from the value in the Receiving BU attribute, then Financial Orchestration determines whether it must do more financial and accounting transactions according to the procurement and shipping flows.

Set Up Different Requisition Trade Organizations

If you set the value in the Selling BU attribute to a value that's different from the value in the Receiving BU attribute, then a shipment financial flow must exist between the receiving business unit and the selling business unit. Financial Orchestration determines whether this flow exists when you create the relationship. If it doesn't, then Financial Orchestration displays an error message.

If a procurement financial flow exists in Financial Orchestration, and if Financial Orchestration can use it for the purchase order that the drop ship references, then Financial Orchestration uses the procurement financial flow when it orchestrates the drop ship.

For example, assume you must create a requisition that references the China inventory organization for each Metal item, and create a requisition that references the United States inventory organization for each Plastic item. Here's your set up.

Drop Ship Financial Flow	Inventory Organization	Financial Orchestration Qualifier
Flow 1	Resides in China as the receiving trade organization.	Use Metal as the item category.
Flow 2	Resides in the United States as the receiving trade organization.	Use Plastic as the item category.

Related Topics

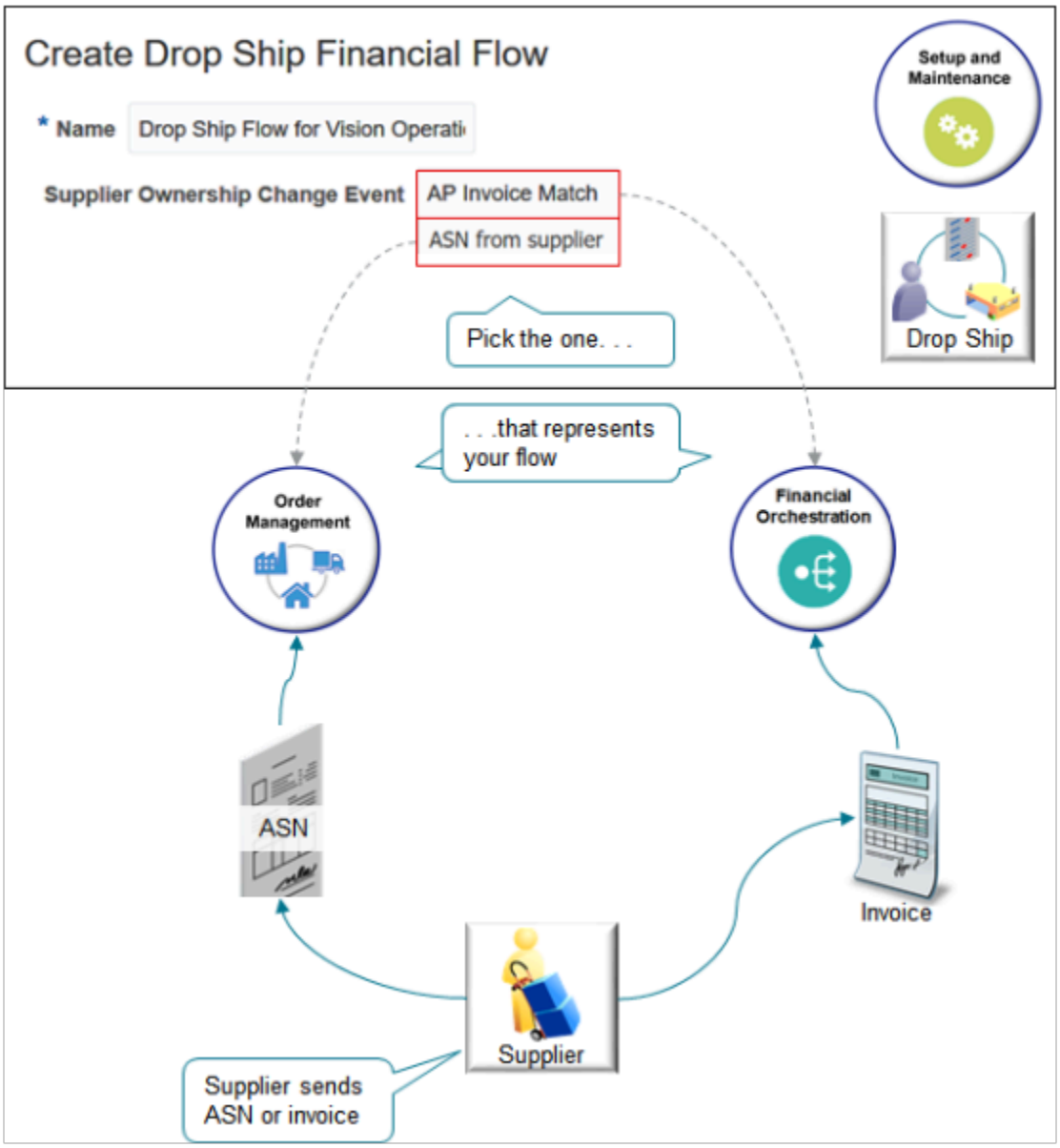
- Set Up Drop Ship in Order Management
- How Drop Ship Works in Order Management
- Overview of Drop Ship in Order Management

Indicate an Ownership Change During Drop Ship

Specify when to change ownership during a drop ship flow.

- When Order Management receives the advance shipment notice (ASN) from the supplier
- Or when Financial Orchestration receives the accounts payable invoice from the supplier

You use the Supplier Ownership Change Event attribute on the Manage Drop Ship Financial Flows page to specify when to change ownership.



ASN From Supplier

If the supplier sends an advance shipment notice (ASN), then set the Supplier Ownership Change Event attribute to ASN From Supplier.

Here's the flow.

1. The supplier sends the advanced shipment notice.
2. Receiving creates receipt and delivery transactions that record the event for the advanced shipment notice.
3. Receiving sends status and event details to Financial Orchestration and Order Management.
    - Financial Orchestration processes the trade transactions in costing, processes the receivable invoices between companies, processes payable invoices between companies, and so on.
    - Order Management sends status and event details to billing and order orchestration.

Communicate the ASN

The supplier can use actions on the supplier portal to create the advance shipment notice.

- Create ASN
- Create ASBN
- Upload ASN
- Upload ASBN

If the supplier uses an informal communication, such as email, to report the shipment of a purchase order that references a drop shipment, then the warehouse manager can also use these features on the Receipts page in the Warehouse Operations work area to manually enter the advance shipment notice. You can also upload an advance shipment notice electronically through XML or EDI. For details, see *Roadmap for Setting up Supplier Portal* and *Overview of Creating ASNs and ASBNs*.

AP Invoice Match

What happens if the supplier doesn't send an advance shipment notice, but instead sends an invoice?

1. The supplier sends an invoice to accounts payable in Financial Orchestration.
2. Financial Orchestration starts orchestration to receive costs and bill the shipment.
   The receiving process works the same way it does for ASN From Supplier, except the process doesn't create an advance shipment notice.
3. Order Management receives and delivers the drop shipment when it creates the accounts payable invoice for the purchase order. It's the same purchase order that the drop ship references.

You don't need to do any set up for AP invoice match because the Transfer Invoice Details to Supply Chain Financial Flow Orchestration scheduled process automatically does the hard work for you. It sends data about validated invoices, canceled invoices, and corrected invoices to Financial Orchestration. For details, see *Use an Improved Oracle Payables Invoice Match Flow for Drop Shipments*.

Related Topics

- Set Up Drop Ship in Order Management
- Set Up Financial Flows for Drop Ship
- Overview of Drop Ship in Order Management
- Oracle Supplier Portal
- Overview of Creating ASNs and ASBNs

Recover an Advance Shipment Notice

Receiving uses the CreateReceiptNotifications event to send shipment details for the sales order through an update on the advance shipment notice or invoice match. An error might happen during this update for a variety of reasons. You can recover from the error.

Here are some examples.

- Order Management never receives the event because there's a problem in the environment, such as a network that's down, a server isn't available, and so on.
- Order Management receives several events in a short period of time. For example, shipping creates several partial shipments and immediately sends an event for each one.
- Order Management receives an event while the sales order is in the Awaiting Shipping status, but Order Management is currently revising the order. Order Management rejects the shipment update because the wait

step on the orchestration process isn't the current step. Receiving already used the event and can't use it again until after Order Management finishes the change.

- Order Management receives an event while the sales order is in the Requisition Created status. In this example, Order Management never received an update for the purchase order, but the purchase order proceeded and shipping already created a shipping notice.
- Order Management receives an update for the advanced shipment notice but the sales order is in error recovery.
- The value in the Change Sequence Number attribute on the fulfillment line of the sales order doesn't match the Change Sequence Number on the purchase order.
- Order Management receives several events in a short period of time. For example, shipping creates several partial shipments and immediately sends an event for each one.

In some situations, Order Management doesn't receive or can't use the event, the status on the sales order remains in Awaiting Shipping, but the status on the purchase order is different depending on what fulfillment shipped.

What Fulfillment Shipped	Status on the Purchase Order
The entire sales order	Closed for Receiving
Only part of the sales order	Open

Now, the status on the sales order isn't synchronized with the status on the purchase order. Order Management will automatically attempt to fix the problem five times in 120 minutes, in 5, 10, 15, 30, and 60 minute intervals. Each attempt will try to recover from errors in the advance shipment notice or from technical errors that happen in the environment when processing the advance shipment notice.

Order Management will automatically attempt to recover errors on the order when you.

- Receive an advance shipment notice but the sales order and purchase order aren't synchronized with each other.
- Receive an advance shipment notice but the purchase order details aren't available on the sales order.
- Attempt to process an advance shipment notice while someone is revising the sales order or when a change on the purchase order hasn't finished.

Recover Manually

If these retries fail, then you can use the Recover Order action in the Order Management work area or the *Recover Errors* scheduled process to recover sales orders that fail when you update the advance shipment notice in a drop ship flow.

You can fix problems that happen when Order Management validates the advance shipment notice, such as the sales order and purchase order aren't synchronized, purchase order details aren't available on the sales order, or attempting to process the notice while someone is revising the sales order or when a change on the purchase order isn't finished.

You can only recover some errors when you use the Recover Order action. If Recover Order doesn't fix the problem, then run the Recover Errors scheduled process.

Constraints

Order Management uses constraints to avoid some of these problems.

Constraint	Description
Update Fulfillment Line When Purchase Order Isn't Available	Prevents the user from changing the Shipped Quantity in Order Management. • Makes sure the Shipped Quantity in Order Management matches the received quantity and the billed quantity in Procurement. • Makes sure the Ordered Quantity in Order Management matches the received quantity and the billed quantity in Procurement. For details, see *Overview of Drop Ship in Order Management*.
Cancel Fulfillment Line When Purchase Order Isn't Available	For details, see *Overview of Drop Ship in Order Management*.

Here are some fixes you can try if you encounter an error even when these constraints are enabled.

Trouble	Shoot
The request to revise the sales order fails because Order Management hasn't finished processing shipment details that receiving created for the sales order.	Wait a few minutes so Order Management can finish processing the shipment details, then resubmit your changes.
The request to process details on the order line for an advance shipment notice fails because someone is currently revising the sales order in Order Management. The request wasn't successful because Order Management can't find details about the orchestration group. Order Management didn't process the request because an error occurred while getting shipment details from Oracle Receiving.	Use the Recover Order action in the Order Management work area to process the shipment details. As an alternative, wait for the revision to finish, go to the Scheduled Processes work area, then run the Recover Errors scheduled process to process the shipment details. You can set parameters when you use the Recover Errors scheduled process to filter the errors you attempt to recover. For example: • To only recover errors that happen on sales order 56477, set the Orchestration Order Number parameter to 56477. • To recover all errors that are happening on sales orders for your Computer Services and Rentals customers, set the Customer Name parameter to Computer Services and Rentals.
The request to process shipment details on the order line fails because the ordered quantity on the sales order doesn't match the quantity on the purchase order for the advance shipment notice in Procurement.	Create an order revision, revise the quantity on the order line to the original ordered quantity, then submit the sales order. Next, go to the Scheduled Processes work area, then run the Recover Errors scheduled process to process the shipment details.
The request wasn't successful because Order Management can't unlock groups for the fulfillment lines. Order Management receives an advance shipment notice (ASN), then identifies each group that contains fulfillment lines for the ASN, but Order Management can't identify the groups so it can't unlock them. In some cases, Order Management might not be able to unlock groups even after it does identify them.	Use the Oracle Enterprise Manager to recover from this problem.

Related Topics

- Set Up Drop Ship in Order Management
- Set Up Financial Flows for Drop Ship
- Overview of Drop Ship in Order Management
- Oracle Supplier Portal
- Overview of Creating ASNs and ASBNs

Set Up Drop Ship for Configured Items

Set up an agreement for the configuration model that specifies the configured item and the configure options that are part of the drop ship.

1. Make sure you have the privileges that you need to administer Oracle Procurement.
2. Go to the Purchase Agreements work area.
3. Click **Tasks > Manage Agreements**.
4. Click **Actions > Create**.
5. In the Create Agreement dialog, set the values, then click **Create**.

Attribute	Value
Style	Configure to Order Blanket Purchase Agreement You must use this value for an assemble-to-order configuration model.
Supplier	Select the supplier who will drop ship the configured item.

6. On the Edit Document page, in the Lines area, add at least one line, then click **Submit**.

Note

- A hybrid configuration model is a type of configuration model that includes an assemble-to-order configuration model as the child and a purchase-to-order configuration model as the parent.

  Order Management uses one blanket purchase agreement to source the assemble-to-order options, and a different blanket purchase agreement to source the purchase-to-order options. So, Order Management issues separate purchase orders to the supplier.
- Use File-Based Data Import to import more than one agreement.

Related Topics

- Set Up Drop Ship in Order Management
- How Drop Ship Works in Order Management
- Overview of Drop Ship in Order Management
- Overview of Importing Orders Into Order Management

Use a Service Mapping to Integrate Order Management with Procurement

Use a service mapping to send data from Oracle Order Management to Oracle Procurement.

Assume you need to send details about a sales order to Procurement so Procurement can process a purchase request.

- Indicate to Procurement that we negotiated the price with the supplier when we created the sales order. We will use the NegotiatedByPreparerFlag attribute to send the indication as a Boolean value.
- Text note that we can send to the buyer. It includes details about the negotiated price, such as any discounts we provided, and why we provided them. We will use an extensible flexfield to capture these details.
- The price that we negotiated with the customer when we created the sales order. We will use the Price attribute to capture the negotiated price.
- You will set up the FLinePackShip_EFF_Custom flexfield on the fulfillment line.
- The ShippingCost_Custom attribute will capture the negotiated price.

Summary of the Set Up

1. Opt into the feature.
2. Add your entity to the integration service mapping.
3. Map your entity to the source.
4. Map entities to the service.
5. Create an integration algorithm.
6. Test your set up.

Opt into the Feature

1. Go to the Setup and Maintenance work area, select the Order Management offering, then click **Change Feature Opt In**.
2. In the row that has Order Management in the Name column, click the **pencil**.
3. In the row that has Enable Custom Payloads for Downstream Integration in the Name column, click the **pencil**.
4. In the dialog that displays, add a check mark to the Drop Ship option, then click **Save and Close > Done**.

Add Your Entity to the Integration Service Mapping

1. Make sure you have the privileges that you need to administer Order Management.
2. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Go to the Pricing Administration work area.
4. Click **Tasks**, and then, under Order Management Configuration, click **Manage Service Mappings**.
5. Open the FulfillmentIntegration service mapping for editing.

   For details, see *Use a Service Mapping to Integrate Order Management with Other Oracle Applications*.
6. Add an entity. On the Entities tab, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	FLinePackShip_EFF_Custom You must include the _Custom suffix to any entity that you add.
Description	Entity for an extensible flexfield that stores details on the fulfillment line. It describes the price we negotiated.

7. Add attributes. In the Details area, click **Actions > Add Row** to add each attribute.

Attribute	Type
FullfillLineId_Custom	Long
ShippingCost_Custom	Double

As an option, you can also add the Price, NegotiatedByPreparerFlag, and NoteToBuyer attributes, but they aren't required. These attributes come predefined on the purchase request line.

For each attribute that you add.

- Leave the Alternate Key attribute empty.
- Make sure the Allow Null attribute contains a check mark.
- Add a check mark to the Primary Key attribute only for FullfillLineId_Custom. Leave it empty for all others.

8. On the Entities tab, in the Entity column, click **PurchaseRequestLine**.
9. In the Detail area, verify that the list contains the Price attribute.

Map Your Entity to the Source

1. Click **Sources**.
2. In the Source column, click **PurchaseRequestSource**.
3. In the PurchaseRequestSource Details area, click **View > Columns**, then display columns.
   - Joined Entity
   - Joined Entity Attribute
4. Click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Entity	FLinePackShip_EFF_Custom Its the entity you added earlier in this procedure.
Type	View Object
View Object	FulfillLineEffBPackShipInstructionprivateVO
Query Type	Unique Identifier
Query Attribute	FulfillLineId

5. Click **Save**, set the values, then click **Save** again.

Attribute	Value
Joined Entity	FulfillLine
Joined Entity Attribute	FulfillLineId

6. In the FLinePackShip_EFF_Custom Details area, add new attributes, then click **Save**.

Attribute	View Object Attribute
FulfillLineId_Custom	FulfillLineId
ShippingCost_Custom	_ShippingCost

7. Map attributes to the PurchaseRequestLine entity.
    - In the PurchaseRequestSource Details area, in the Entity column, click **PurchaseRequestLine**.
    - In the PurchaseRequestLine Details area, add attributes, then click **Save**.

Attribute	View Object Attribute	Expression
NegotiatedByPreparerFlag	Empty	true
NoteToBuyer	Empty	Empty
Price	Empty	Empty
FulfillLineId	FulfillLineId	Empty

Map Entities to the Service

1. Click **Services**.
2. In the row that contains PurchaseRequestService in the Service column, set the values, then click **Save**.

Attribute	Value
Implementation Type_Custom	Algorithm
Implementation_Custom	`Integrate Order Management with Procurement Custom`

Attribute	Value
	This is the name of the algorithm you created earlier in this procedure.

3. In the Details area, add entities, then click **Save**.
    - FLinePackShip_EFF_Custom
    - Header_Custom
    - FulfillLine_Custom
    - PurchaseRequestConfig_Custom
    - PurchaseRequestHeader_Custom
    - PurchaseRequestLine_Custom

    Note

    - If you use this service to send your own custom attributes from Order Management to Procurement, then you must also add your custom entities.
    - Click **Actions > Add Row** to add each entity.

      You must include the _Custom suffix.
    - Make sure the Read attribute and Write attribute contain a check mark for each entity.
    - Leave the other attributes empty.
4. Add attributes to the FLinePackShip_EFF_Custom entity.

Attribute	Alias
FulfillLineId_Custom	FulfillLineId
ShippingCost_Custom	Leave Empty

5. Add attributes to the PurchaseRequestLine entity.
    - FulfillLineId
    - NegotiatedByPreparerFlag
    - NoteToBuyer
    - Price

    When you add attributes to these entities.

    - Make sure the Read attribute and Write attribute contain a check mark for each attribute.
    - Leave the other attributes empty.

Create an Integration Algorithm

Create an integration algorithm that you can use to call Procurement.

1. Click **Tasks**, and then, under Order Management Configuration, click **Manage Algorithms**.
2. On the Create Algorithm page, set the values, then click **Save**.

Attribute	Value
Name	Integrate Order Management with Procurement Custom
Description	Use this integration algorithm to send our custom data from Oracle Order Management to Oracle Procurement.

3. Click **Add Step > Conditional Action**.
4. In the Data Sets area, add two data sets.

Name	Variable Path	Primary	Cardinality	Data Set Join
PRCLine	PVar.PurchaseRequestLine	Contains a check mark	Leave empty	Leave empty
PRCLineEff	PVar.FLineEff	Doesn't contain a check mark	Zero or one	[FulfillLineId: {PRCLine.FulfillLineId}]

   FlineEFF is an alias for the FLinePackShip_EFF_Custom entity. You must use the FlineEFF alias or the entire FLinePackShip_EFF_Custom entity name.
5. In the Execute Condition area, click **Add Condition > Default Action**.
6. In the Edit Actions window, then enter this code.

```
PRCLine.Price = PRCLineEff!= null? PRCLineEff.ShippingCost_Custom: null
```

   As an option, add code that maps an attribute you add to the entity for your extensible flexfield in the service mapping, or that you code as a string. For example:

```
PRCLine.NoteToBuyer = "We negotiated price with the supplier"
```

7. Click **Save**.
8. Test and publish. For details see *Set Up an Integration Algorithm*.

Test Your Set Up

Assume you're on a call with a preferred, long-term supplier who orders a quantity of 10, A54888 desktop computers. To beat your competitor's price, you agree to provide a 20% discount on the order line. As part of the deal, you convince your supplier to add 4 printers to the order.

1. Create a sales order in Order Management.

2. Add an order line, then set the attribute on it.

Attribute	Value
Your Price	Enter the negotiated price.

3. Submit the sales order.
4. Examine the details in Procurement.

Related Topics

- Use a Service Mapping to Integrate Order Management with Other Oracle Applications
- Set Up an Integration Algorithm
- Create a Sandbox So You Can Edit Service Mappings

Keep the Dates on Your Sales Orders and Purchase Orders Synchronized

Keep the dates on your sales orders in Order Management synchronized with the dates on your purchase orders in Procurement.

- If you update the shipment date, delivery date, or shipment method on a purchase order that you manually create or on an open purchase order that you revise, then Order Management automatically updates these values on the sales order line and adds the reason for the change on the sales order line.
- If the item is a pick-to-order item, assemble-to-order item, shipment set, or kit, then you can change the shipment date, delivery date, or shipment method only if the new value that you provide on the purchase order line is the same for all components that are part of that item, set, or kit.
- If you Use Global Order Promising to recalculate dates in Order Management, and if you change the promised ship date or promised arrival date in Procurement, then Order Management will change these dates on the order line, and Promising will automatically recalculate them. For details, see *Use Order Profiles to Control Order Management Behavior*.

This feature applies to these types of items:

- Standard
- Pick-to-order
- Assemble-to-order
- Items that are part of a shipment set or kit

Specify the Reason

This feature comes predefined to set the reason for the update to Update from Supplier. You can specify a different reason to meet your needs.

Assume you want to change the reason to Update from Purchase Order. Here's how:

1. Go to the Setup and Maintenance work area, then go to the task.
2. Offering: Order Management
3. Functional Area: Orders
4. Task: Manage Order Lookups
5. On the Manage Order Lookups page, search the Lookup Type attribute for DOO_SCHEDULE_REASON.

6. In the Lookup Codes area, in the row that has ORA_DOO_SUPPLIER_DATE_CHANGE in the Lookup Code column, change the value.

Attribute	Value
Meaning	Update from Purchase Order

Update Attributes on Split Lines for Drop Ship

Update attributes on a split order line that Oracle Order Management creates when a supplier ships part of an order line in a drop shipment flow.

If only some of the quantity on the order line is currently available on the requested date, then your drop ship supplier might ship that quantity, and Order Management will split the order line for the remaining quantity. You can update attributes on the split fulfillment line.

Use this feature so you can update attributes on a split order line that Order Management creates when Oracle Procurement ships part of an order line in a drop shipment flow.

Use this feature to realize these benefits:

- Update attributes on an order line after you ship part of a drop shipment line.
- Avoid having to cancel split lines and manually recreate new ones when all you want to do is update attributes on the shipment line.
- Reduce the number of steps and the time you need to process change on an order line after you ship part of that line.
- Improve efficiency and accuracy of how you process change on the line after you ship part of that line.
- Improve tracking and reporting for lines that you fulfill in more than one shipment.

The usages described in this topic involve split lines that you ship in a drop shipment.

Example

A supplier can ship part of a fulfillment line that you drop ship to help meet customer demand. If only some of the quantity on the order line is currently available on the requested date, then the supplier ships the quantity that's available to ship, and Order Management will create a split order line for the remaining quantity. You now have a new fulfillment line that has the remaining quantity that the supplier still needs to ship.

You might need to update attributes on the new split line in Order Management after the supplier ships part of the order line. For example, assume you know that supplier B has enough supply to fulfill the line that hasn't shipped. If you revise the supplier and supplier site attributes on that line to supplier B and submit your revision, then Order Management will send these details to Procurement so Procurement can create a new purchase order for supplier B, and supplier B can fulfill the request.

Update Attributes in Order Management

You can update one or more attributes on an order line that you split and ship through a drop shipment:

- Update line attributes just like you do when you update attributes on a line that isn't split and when the line is awaiting shipping. You can also use actions in a fulfillment view to do this.

- As an option, use the Update Lines action to update attributes on all the lines that you select at the same time. If you opt into this feature, then you can use this action for the split lines and for the order lines that you backorder in Procurement.
- Update an attribute on the order header, and Order Management will cascade your update to all the split lines that you haven't overridden. You can also use the Override Order Line action to update an attribute on the split line.
- Use an order management extension or posttransformation rule to update an attribute on a split line.

Note

- If you ship lines in a shipment set, and if you partially ship or don't ship at least one of these lines, then Order Management removes the lines that you haven't shipped from the set. You can then update attributes on the removed lines just like you do on a line that isn't in a shipment set.
- If you update an attribute on a split through REST API, or through some other application, such as Backlog Management, Global Order Promising, and so on, but it's a split line that Order Management doesn't allow you to update, then you will receive an error.
- If you do an unschedule action for one or more split lines, and if the fulfillment tolerance on at least one of the lines that you select isn't 0, then Order Management will ask you to confirm the action. For background, see *Guidelines for Setting Up Shipment Tolerances*.

If you use an action in a fulfillment view to update the split lines, and if your selection includes:

- **Only the split lines that Order Management can't update**. Order Management will display the same error that it displays when you don't use this feature.
- **Split lines that Order Management can update**. Order Management will update them, and won't update the split lines that it can't update.

You can't update an attribute if:

- It's on a remnant line of a configured item. For more about remnants, see *Fulfill Configured Items*.
- It's on a line that Order Management created when Oracle Receiving partially received a return line.

> **Note:** You can't update an attribute if it's on a sales order that you submitted before you opted into the Update Attributes on Split Order Lines for Partial Drop Shipments feature. You can update attributes on a split line that you drop ship only on a sales order that you submit after you opt into the feature.

Update Attributes on Sales Orders

You can update attributes on split order lines:

- Modify the quantity, scheduled date, shipping location, shipping method, supplier, or supplier site.
- Modify the price of a purchase order line, through an extensible flexfield on the order line.
- Unschedule the line or substitute the item.
- Schedule or split the line when it's in Manual Scheduling Required status.

You can update an attribute's value on a split line in Order Management, and then use that value to:

- Reprice and recalculate tax.
- Fulfill, recognize revenue, and bill the split lines.

- Cost the transaction and recognize the cost of goods sold for the split lines in Oracle Cost Management.
- Orchestrate financial details and process the transaction in Oracle Supply Chain Financial Orchestration for the split lines.

Do these updates in the Order Management work area, through REST API, or through some other application, such as Oracle Backlog Management or Oracle Global Order Promising.

Update Attributes on Purchase Orders

You can update attributes on a purchase order's partially shipped line:

- Modify the promised dates or quantity.
- Split the schedule on the line.
- Cancel the purchase order.

You can update attributes on a line that Procurement never received in a purchase order, and when the line contains a configured item, a kit, or the item is part of a shipment set:

- Modify the quantity or split the schedule.
- Cancel the purchase order.

Do these updates in Procurement, through REST API, or through Oracle Supplier Portal.

How Order Management Processes Change

Order Management User Updates an Attribute

Order Management processes change on a split line similar to how it processes change on a line that isn't split and that's awaiting shipping. Order Management compensates change differently depending on the attributes that you update or the fulfillment task.

If you update an attribute on a split line, and if the orchestration process uses that attribute to identify change for a fulfillment task, then the orchestration process sends the updated attribute value on the split line to Procurement. For example, if you update the ordered quantity, shipping method, scheduled date, shipping location, and so on a split line, then Order Management compensates the procurement task and sends the updated attribute value to Procurement.

- The orchestration process updates all of its fulfillment tasks, then sends the split line to Procurement.
- You can cancel a split line on a sales order or fulfill it without updating any attribute just like you do if you don't use this feature. Procurement will close the purchase order line that fulfills the split line.
- Order Management doesn't compensate the parent line of a split line when you cancel that split line for a partial shipment.

If you substitute an item, do an unschedule action, or change the fulfillment from a drop shipment to a standard warehouse flow or to a back-to-back flow, then Order Management will send a request to Procurement to close the purchase order.

- For example, if you substitute an item, then Procurement will close the old purchase order and create a new one for the new item.
- If you unschedule a split line, then you can reschedule the line to a new supplier or warehouse. Procurement will also close the purchase order that was open for line before you unscheduled it.
- If you revise the split line to remove the supplier and add a warehouse, then Procurement will close the purchase order and Order Management will interface the line to your warehouse for fulfillment.

Buyer or Supplier in Procurement Updates an Attribute

Here are some examples of how Order Management processes an update on a purchase order when Procurement receives only part of the quantity, and a buyer or supplier updates an attribute on a split line in Procurement.

What You Do in Procurement	What Order Management Does
Reduce the quantity on the purchase order line.	If you reduce the purchase order quantity to: • **A quantity that exceeds the quantity that Procurement already received**. Order Management updates the quantity on the backordered line, creates a new fulfillment line for the remaining quantity that the supplier can't fulfill, then sets the status on the new fulfillment line to Manual Scheduling Required. Procurement continues to allow the purchase order to receive more quantity. • **The same quantity that Procurement already received**. Order Management sets the status on the backordered fulfillment line to Manual Scheduling Required. Procurement closes the purchase order and prevents it from receiving more quantity.
Cancel the schedule on the purchase order line.	If you set the Cancel Unfulfilled Demand attribute to: • **Yes**. Order Management updates the backordered fulfillment line's status to Manual Scheduling Required. • **No**. Order Management updates the backordered fulfillment line's status to Requisition Created. If your item is a configured item, kit, or shipment set, then you can cancel the entire purchase order but not the purchase order line and schedule.
Update the promised ship date or the promised delivery date on the schedule.	Update the scheduled ship date or the scheduled arrival date on the backordered fulfillment line.
Split the purchase order's schedule into more than one schedule and update the promised dates.	Split the line into more than one line and update the scheduled ship date or the scheduled arrival date on these lines.

Here's a summary of what the buyer or supplier can do on a purchase order line when Procurement has received only part of the quantity.

Type of Item	Reduce Quantity	Split Schedule	Revise Ship Date	Revise Delivery Date	Revise Ship Method	Cancel Schedule	Cancel Line	Cancel Order
Standard	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Assemble to Order	Yes	No	Yes	Yes	No	Yes	Yes	Yes
Pick to Order	Yes	Yes	Yes	Yes	No	No	No	Yes
Kit	Yes	Yes	Yes	Yes	No	No	No	Yes
Hybrid	No	No	Yes	Yes	No	No	No	No
Shipment Set	Yes	No	Yes	Yes	No	Depends	Depends	Yes

Note

- Depends means Yes, but only for standard or assemble-to-order items.

- If the item is a configured item, kit, or shipment set, and if you modify the date or shipment method, then you must apply the same date or shipment method across all the lines that are part of the configured item, kit, or set.

If Procurement hasn't received any of the quantity, then you can still make all of the same modifications that you can make when Procurement received only part of the quantity, except for these differences with a shipment set.

Type of Item	Reduce Quantity	Split Schedule	Revise Ship Date	Revise Delivery Date	Revise Ship Method	Cancel Schedule	Cancel Line	Cancel Order
Shipment Set	No	No	-	-	-	No	No	-

How Procurement Processes Change

Here are some examples of how Procurement processes an update that you make on an attribute on a split line on a sales order in Order Management when Procurement receives only part of the schedule.

What You Do in Order Management	What Procurement Does
Modify the quantity.	1. Split the current schedule. 2. Reduce the old schedule's quantity to the quantity that Procurement already received. 3. Create a new schedule that has the new quantity from Order Management.
Modify the scheduled date.	1. Split the current schedule. 2. Reduce the old schedule's quantity to the quantity that Procurement already received according to the old promised date. 3. Create a new schedule that includes the quantity that Procurement hasn't received and the revised promised date from Order Management.
Add a new price for the purchase order through an extensible flexfield on the sales order line.	1. Split the current schedule. 2. Reduce the old schedule's quantity to the quantity that Procurement already received according to the revised price. 3. Create a new schedule that includes the quantity that Procurement hasn't received and the new price from Order Management. 4. Update the price on the purchase order's line.
Modify the shipping method.	1. Split the current schedule. 2. Reduce the old schedule's quantity to the quantity that Procurement already received according to the old shipping method. 3. Create a new schedule that includes the quantity that Procurement hasn't received and the new shipping method from Order Management.
Modify the shipping location.	1. Split the current schedule. 2. Reduce the old schedule's quantity to the quantity that Procurement already received according to the old shipping location. 3. Create a new schedule that includes the quantity that Procurement hasn't received and the new shipping location from Order Management.
Unschedule the line.	Close the purchase order with the quantity that Procurement already received.
Substitute the item, or modify the supplier or supplier site.	1. Close the purchase order with the quantity that Procurement already received for the old item, or from the old supplier or supplier site. 2. Create a new requisition and a new purchase order for the new item, or for the new supplier or supplier site.

How Supply Chain Financial Orchestration and Cost Management Process Change

Assume you add a new price for the purchase order through an extensible flexfield on the order line in Order Management, and then Procurement updates the price on the purchase order line:

1. Oracle Financial Orchestration will calculate the transfer price according to the price on the updated purchase order line for each subsequent drop shipment.
2. Oracle Cost Management will use the new price when it processes each subsequent drop shipment. It will also adjust the price on each prior drop shipment according to the new price on the purchase order line.
3. If you modify the supplier or supplier site on the sales order line, then Procurement will create a new purchase order, and Financial Orchestration will create a new financial agreement for each subsequent drop shipment.

Assume on-hand supply becomes available, so you update supply details on the sales order line and ship the remaining quantity that you haven't shipped yet from the warehouse. This modification will change the sales order's flow from a drop shipment to a standard flow or to an intercompany transfer. Financial Orchestration and Cost Management will use the new flow when they process and cost each subsequent shipment.

Guidelines

- To use this feature, you must opt into the Update Attributes on Split Order Lines for Partial Drop Shipments feature.
- You can't opt out of this feature after you opt in and submit a sales order. At this point, you're committed to using the feature.
- You can update attributes on a split line that you drop ship, and that has a configured item or a kit just like you do when you update attributes on a line that isn't split and that's awaiting shipping, but you can't modify the configured item on the split line.
- If you update an attribute on a split line, then Oracle Pricing reprices the line just like it does when you update an attribute on a line that isn't split and that's awaiting shipping.
- If you drop ship a split line in a standard or in a back-to-back flow, then you can set the supplier and the supplier site on the split line.
- If you substitute the item on a split line or unschedule it, and if the fulfillment tolerance on the line isn't 0, then fulfillment will apply the fulfillment tolerance on the split order line, but it won't consider the cumulative quantity that you already shipped across all of the order line's fulfillment lines. Instead, fulfillment will consider only the quantity on the fulfillment line that you're shipping.
- You can't use a web service from Application Development Framework to update an attribute on a split line that you drop ship.

For background, see *Update Attributes on Split Order Lines for Partial Shipments*.

Access Requirements

You need these privileges to use this feature:

- Initiate Order (FOM_CREATE_ORDER_PRIV)
- Revise Order (FOM_REVISE_ORDER_PRIV)
- View Orders (FOM_VIEW_ORDERS_PRIV)
- Change Purchase Order (PO_CHANGE_PURCHASE_ORDER_PRIV)
- Maintain Supply Chain Financial Orchestration Flow (FOS_MAINTAIN_SUPPLY_CHAIN_FINANCIAL_TRADE_AGREEMENT_PRIV)

- Create Cost Accounting Distributions (CST_CREATE_COST_DISTRIBUTIONS_PRIV)
- Review Cost Distribution (CST_REVIEW_COST_DISTRIBUTIONS_PRIV)

This feature doesn't introduce any new privileges.

4 Set Up Features

Approvals

Guidelines

Overview of Setting Up Approval

Set up Order Management to route each sales order for sales order approval.

Sales order approval is a process that includes one or more approvers who must approve a sales order before Order Management sends it to order fulfillment. An approval rule determines whether to do approval. Here are some examples.

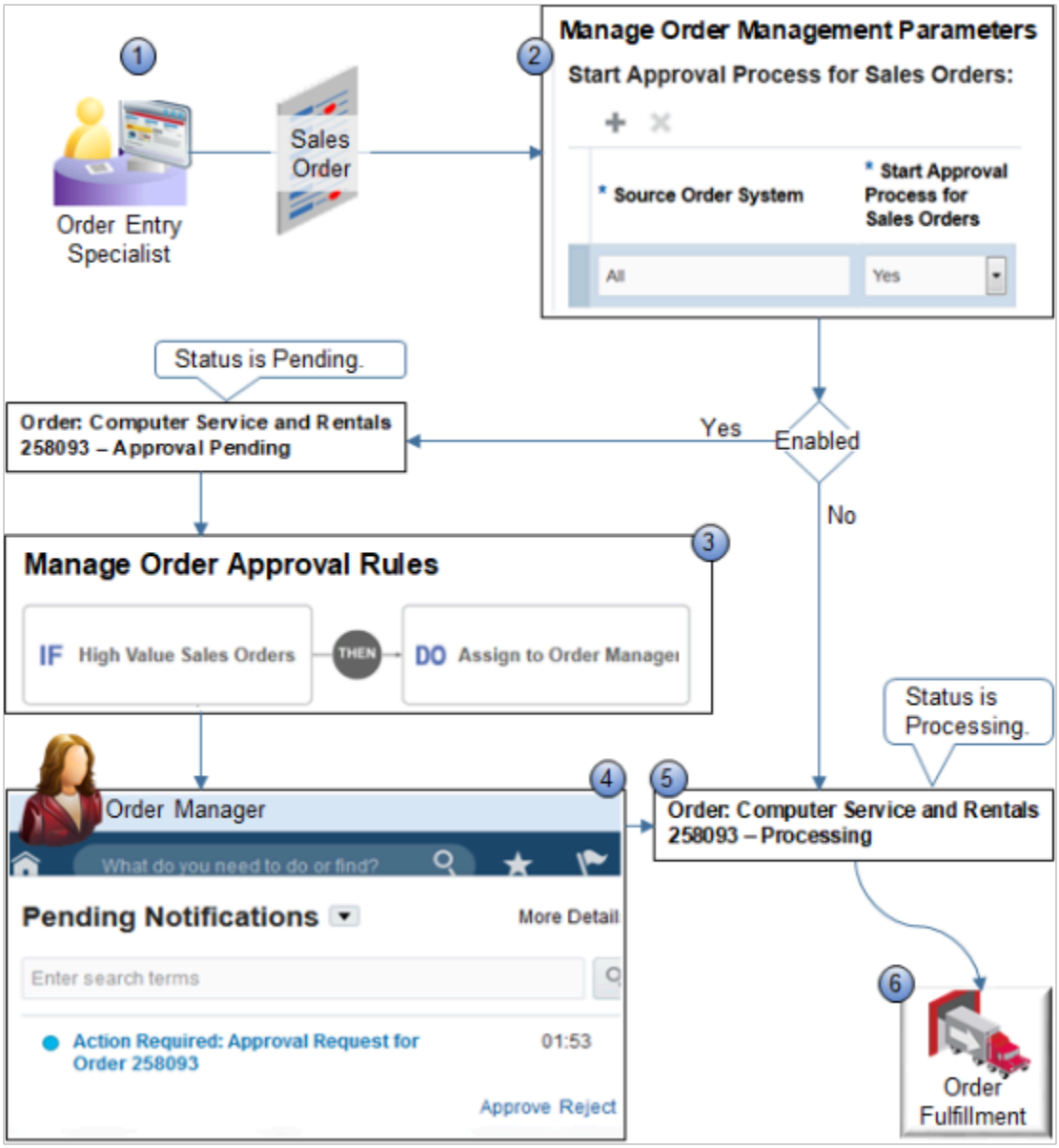
Assume your company policy limits the discount you provide to your customer to 10%. Create a rule to implement the policy.

```
If an Order Entry Specialist adds a 20% discount, then route the sales order to an Order Manager for approval.
```

Note

- You can set it up to accept or reject the sales order according to a variety of approval rules, and on a variety of sales order attributes.
- Use approval to streamline how you enforce a policy, such as to streamline communication between the Order Entry Specialist and one or more approvers, and to improve the experience for people who enter and approve the order.
- Approving the sales order before starting order fulfillment can reduce the number of sales orders that Order Management must return to Draft status because they don't meet a fulfillment requirement for some reason, which in turn reduces rework after fulfillment starts.

Approval includes at least one Order Entry Specialist and one or more approvers. Consider an example where you need approval when the sales order total exceeds $10,000.



Note

1. An Order Entry Specialist creates a sales order, and then clicks Submit.

   Alternatively, Order Management receives a source order from a source system.
2. If the Source Order System value that you set in the Start Approval Process for Sales Orders parameter matches the source order system attribute of the sales order, and if Source Order System is Yes, and if you create an approval rule:

   - **Then.** Order Management sets the order status to Approval Pending and routes the sales order for approval.
   - **Else.** Order Management doesn't evaluate any approval rules, even if defined, and proceeds to step 5.

   For details, see *Manage Order Management Parameters*.

3. The rule you create on the Manage Order Approval Rules page routes the sales order through the approval process, such as assign and route the sales order to an order manager.
4. The approver uses their Pending Notifications page or worklist to view and approve the approval request.

   If the approver rejects the approval request, then Order Management sets the sales order status to Draft, unlocks it, routes it back to the person who created the sales order, and displays a message on the Order page that describes the rejection.

   The person who creates the order can use the Approval Notes tab on the sales order to get approval details, including suggested actions the approver entered in comments when rejecting the approval request. For example:

   - `Split the sales order into two separate orders so the total amount of the sales order doesn't exceed $10,000`
5. Order Management sets the order status to Processing, then sends it to order fulfillment.
6. Order fulfillment fulfills the sales order.

Related Topics

- Set Up Approval Rules for Sales Orders
- Get Approvals for Sales Orders

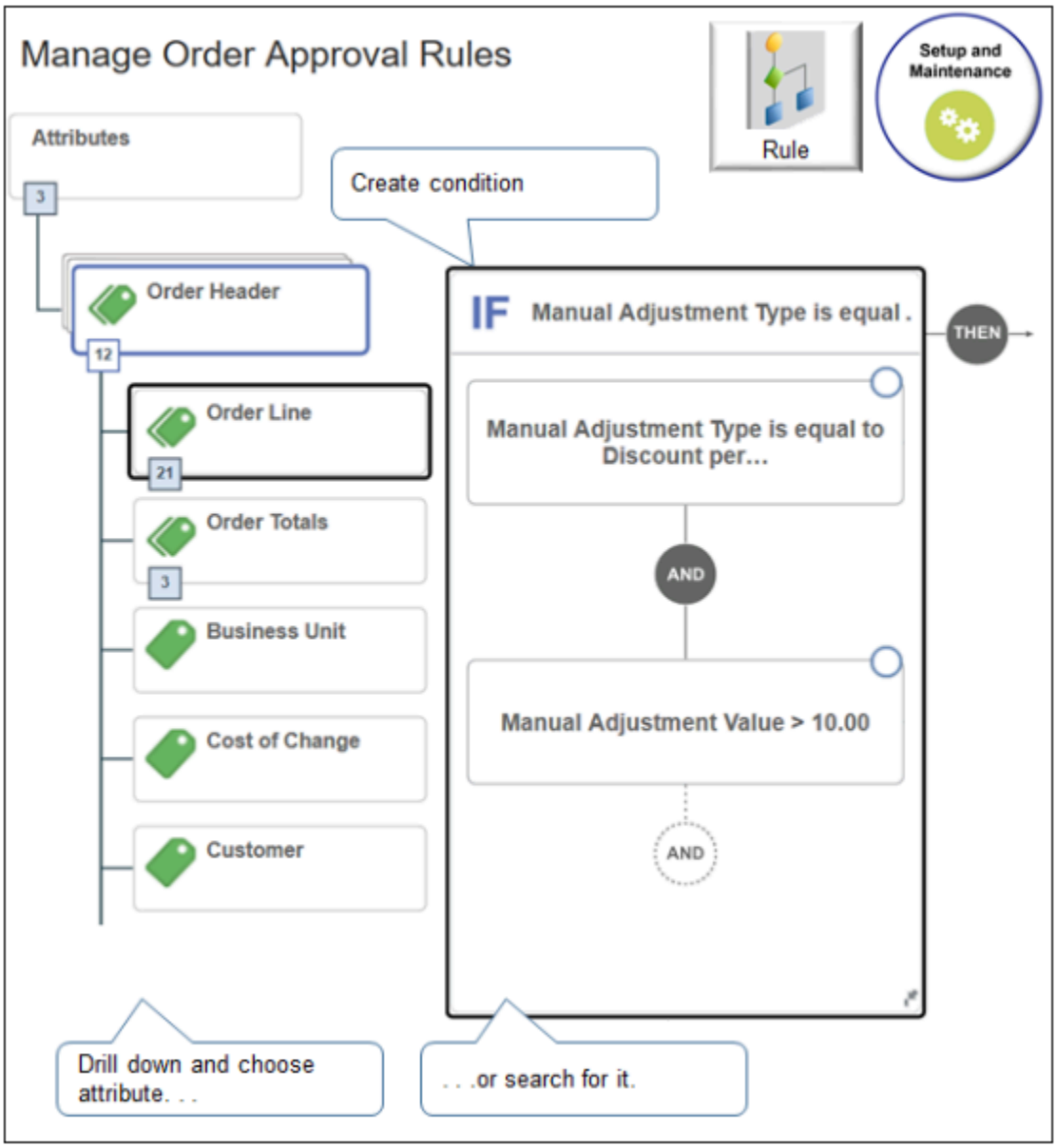
Guidelines for Setting Up Your Approval Rule

Use guidelines to help you create your approval rule.

For details about how to set up other parts of approval, see the Define Approval Management section in *Implementing Common Features for Oracle SCM*.

Create Your Rule

Search for Manage Order Approval Rules in the Setup and Maintenance work area. For details, see *Overview of Using Business Rules With Order Management*.


Manage Order Approval Rules
Rule
Setup and Maintenance
Attributes
3
Create condition
Order Header
12
Order Line
21
Order Totals
3
Business Unit
Cost of Change
Customer
IF
Manual Adjustment Type is equal .
THEN
Manual Adjustment Type is equal to Discount per...
AND
Manual Adjustment Value > 10.00
AND
Drill down and choose attribute. . .
. . .or search for it.

Create a Condition

Specify the condition that determines when to do approval according to the value of an attribute.

- Select from a wide range of attributes.
    - Order header attribute
    - Order line attribute
    - Order total
    - Order charge
    - Price adjustment
    - Validation
    - Order type attribute on the order header

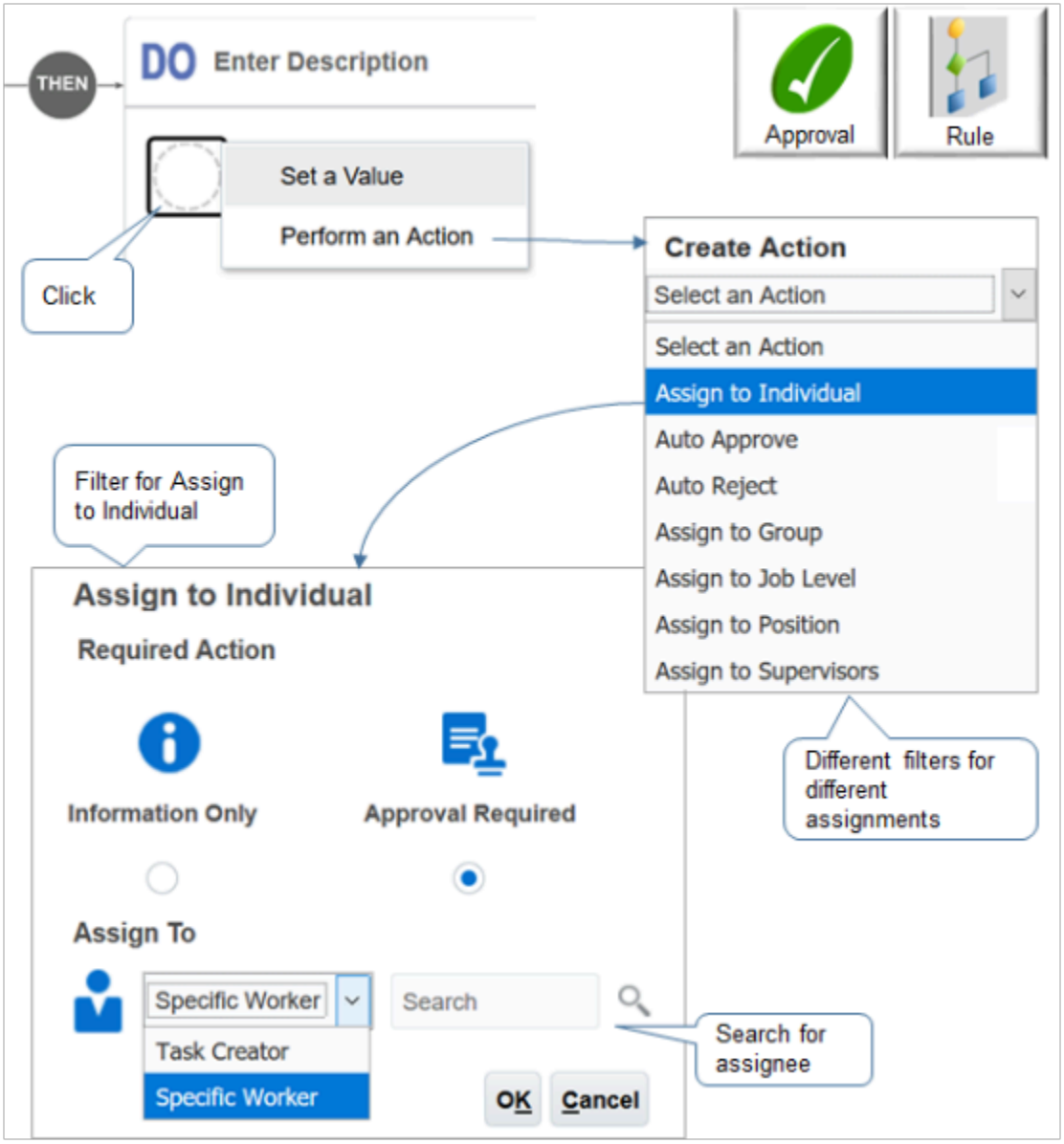
The Manage Order Approval Rules page filters the attributes you can use. Use the tree to select the attribute, or, more simply, enter it in the If statement. If you use the Tree, only use attributes under the Order Header branch. Don't use the HierarchyPrincipal or Task branches.

For example, in the If area:

- Click **New Condition** (the dashed circle).
- In the Create Condition dialog, enter `adj`, wait a moment, then click **Manual Adjustment Type**.
- Set the condition to `Manual Adjustment Type is equal to Discount Percent`, then click **OK**.
- Click **And**.
- In the Create Condition dialog, enter `adj`, wait a moment, then click **Manual Adjustment Value**.
- Set the condition to `Manual Adjustment Value > 10`, then click **OK**.

Specify the Action

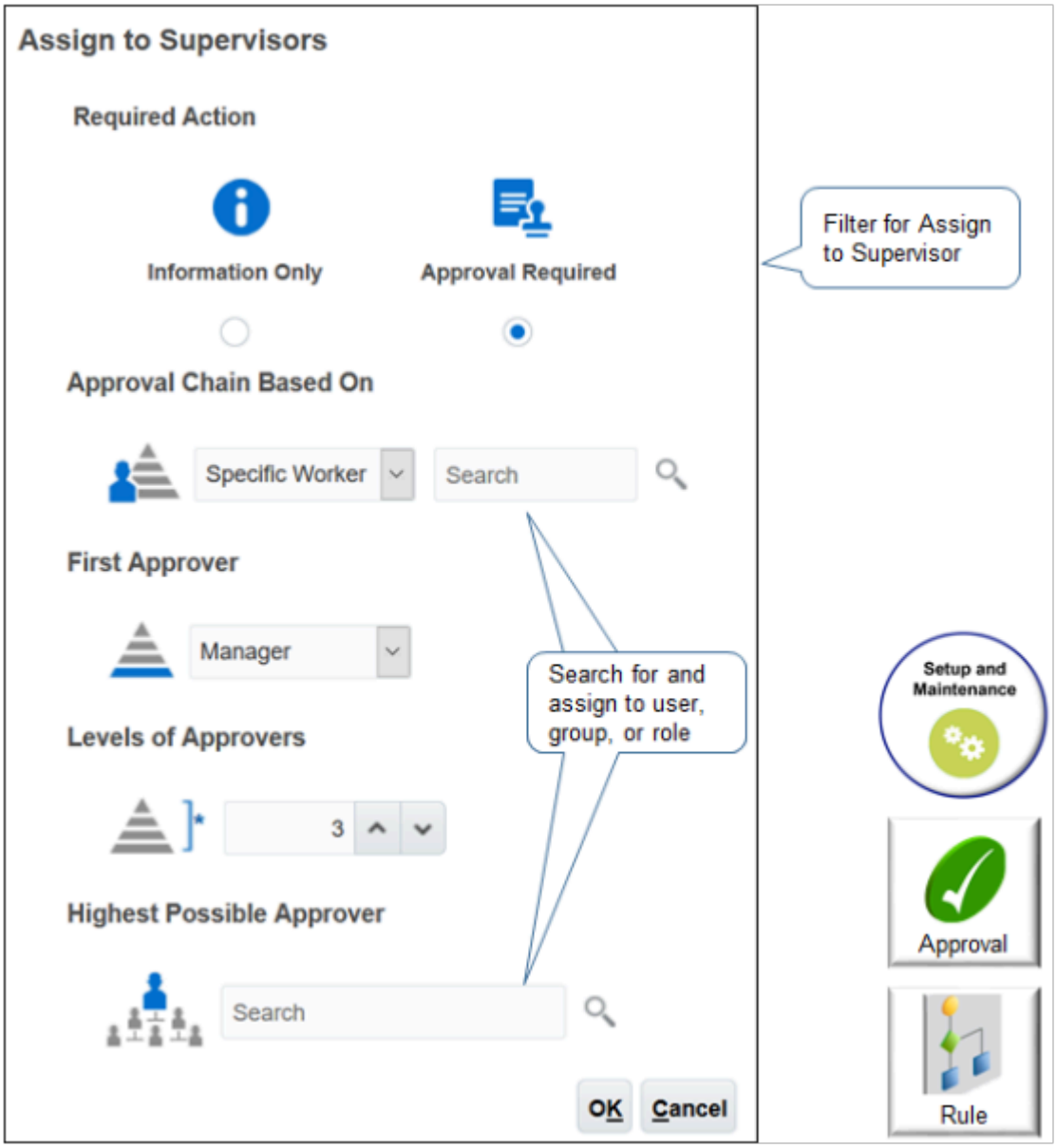
What should we do if the condition evaluates to true? You have a range of choices, such as Assign to Individual, Assign to Group, Assign to Supervisors, and so on. The rule editor filters your choices depending on the action that you select.



For example, if you select:

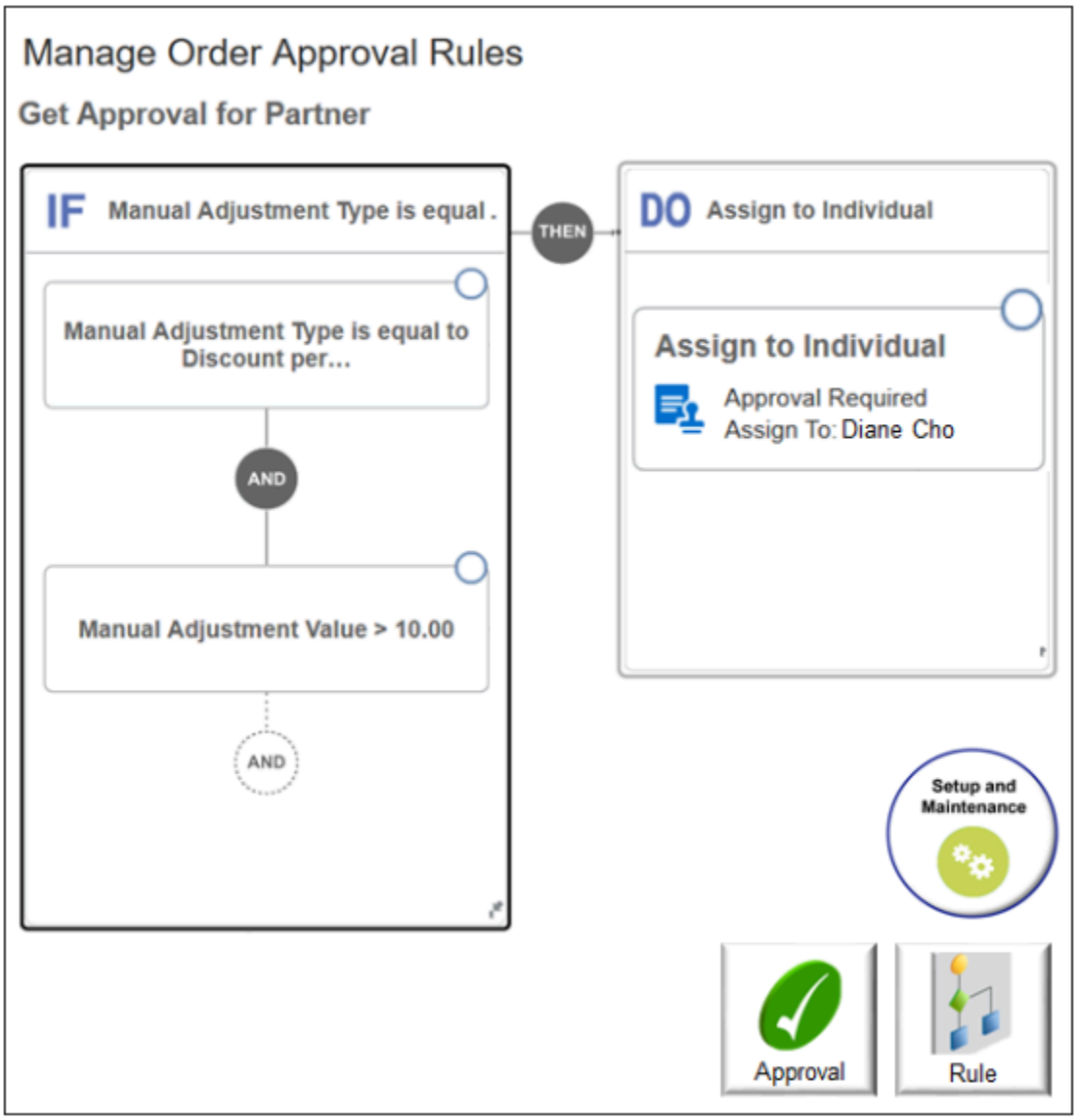
- **Assign to Individual**. A dialog displays where you can select the individual. It includes other options, such as Information Only or Approval Required.
- **Assign to Group**. A different dialog displays where you can select the group.

- **Assign to Supervisors**. A different dialog displays where you can specify the approval chain, first approver, level of supervisors, highest approver, and so on.



- Click **Then > Do**.
- Click **New Action > Perform an Action**.
- In the Create Action dialog, click **Assign to Supervisors**.
- Set the value in Approval Chain Based On first. It affects values you can chose in other attributes, such as First Approver.

Here's the rule for this example.



The rule is:

`If Manual Adjustment Type is Discount Percent, and if Manual Adjustment Value is greater than 10, then assign approval to Diane Cho.`

Use the Primary Total

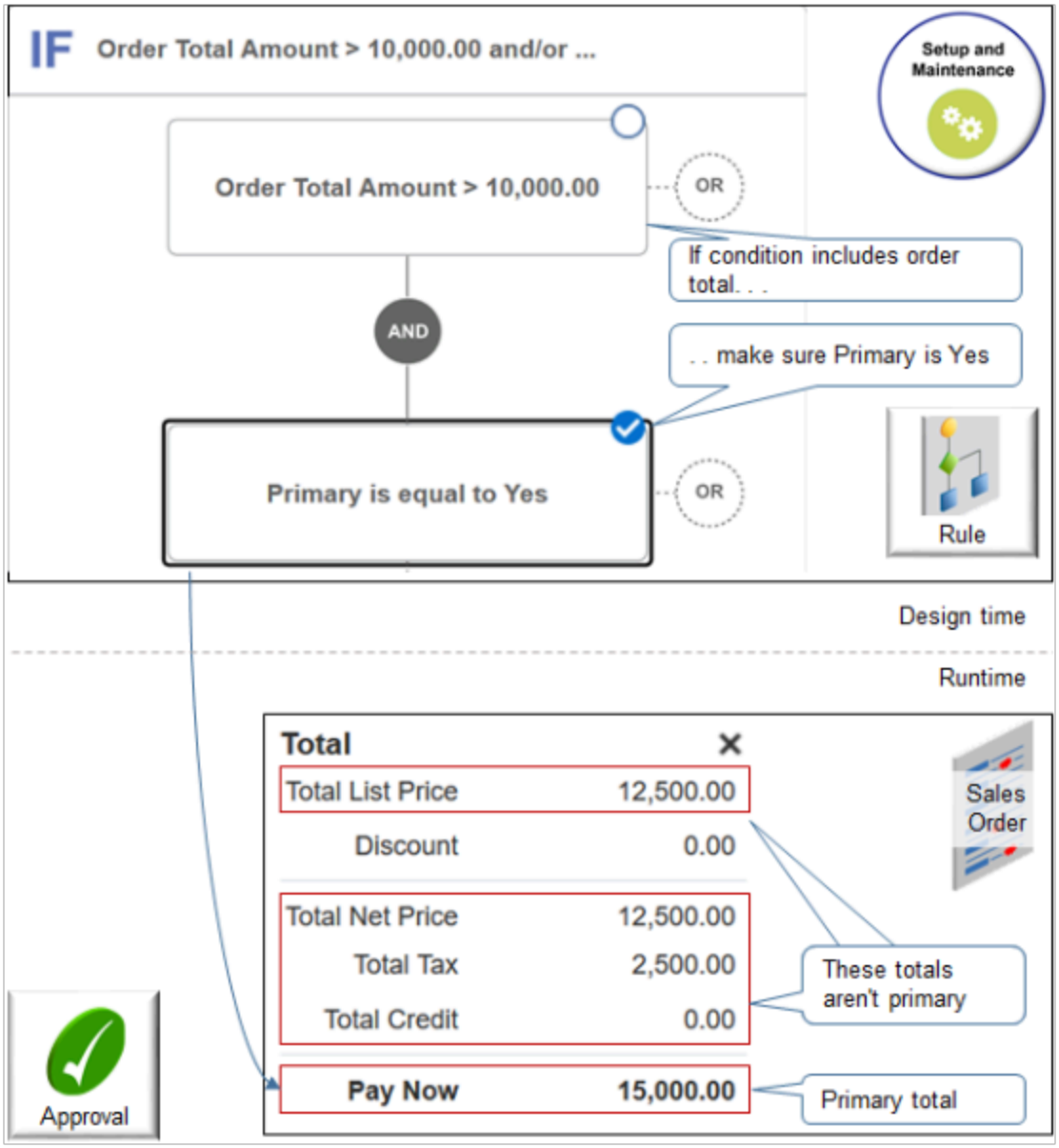
If your condition references the order total, then make sure your rule specifies the primary total.

A sales order includes more than one total. For example, it includes the header total, total list price, total net price, total tax, total credit, and so on. The header total is the primary total. In the price breakdown dialog, it displays on the Pay Now row.

Assume you create an approval rule that filters only according to **`Order Total Amount is greater than 10,000`**. At runtime, the rule will create a separate approval request for each order total that's greater than 10,000 although you require only one approval request for the sales order.

To avoid this problem, add another condition that filters approval according to `Primary equals Yes`. This way, the rule evaluates only the primary order total, ignores the other totals, and sends only one approval request.

For example:



Various Guidelines

- Order Management doesn't come with predefined approval rules because they're specific to your requirements. If you need approval, then you must create an approval rule.
- Approval evaluates each order line, one at a time, sequentially.

- If more than one rule applies for different order lines, then approval creates a separate approval flow for each line that requires approval.

  For example, if approval rule b applies to line 1, and approval rule c applies to line 2, then approval sends approval request x for line 1 and request y for line 2. The approvers in request x might be different from the approvers in request y, depending on how you set up your rules.
- You can set up a group of approvers. For example, get three separate approvals from user x, user y, and user z for a single approval request. Use the Manage Approval Groups page in the Setup and Maintenance work area, and assign the group in your rule. Approval will send notification to each user, sequentially. For example, it will send notification to user, x, and then y, and then z.
- Make sure you have the FOM_MANAGE_ORDER_APPROVAL_RULES_PRIV privilege when you set up your rule.

Limitations

- Order Management approves only the entire sales order. It can't approve only an order line.
- You can't start approval according to order status. For example, you can't start approval depending Draft status or Processing status. Approval automatically sets status to Approval Pending.
- You can't start an approval flow according to a change in an attribute value during order revision.
- You can't start an approval flow according to comparison between different versions of the sales order.
- Use an approval only before Order Management submits the sales order to order fulfillment. You can't do approval during order fulfillment.
- You can't skip approval according to changes that you make to a specific attribute on an order revision at runtime.

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Order Management Parameters
- Set Up Approval Tasks for Sales Orders
- More Setup for Workflow Email Notifications

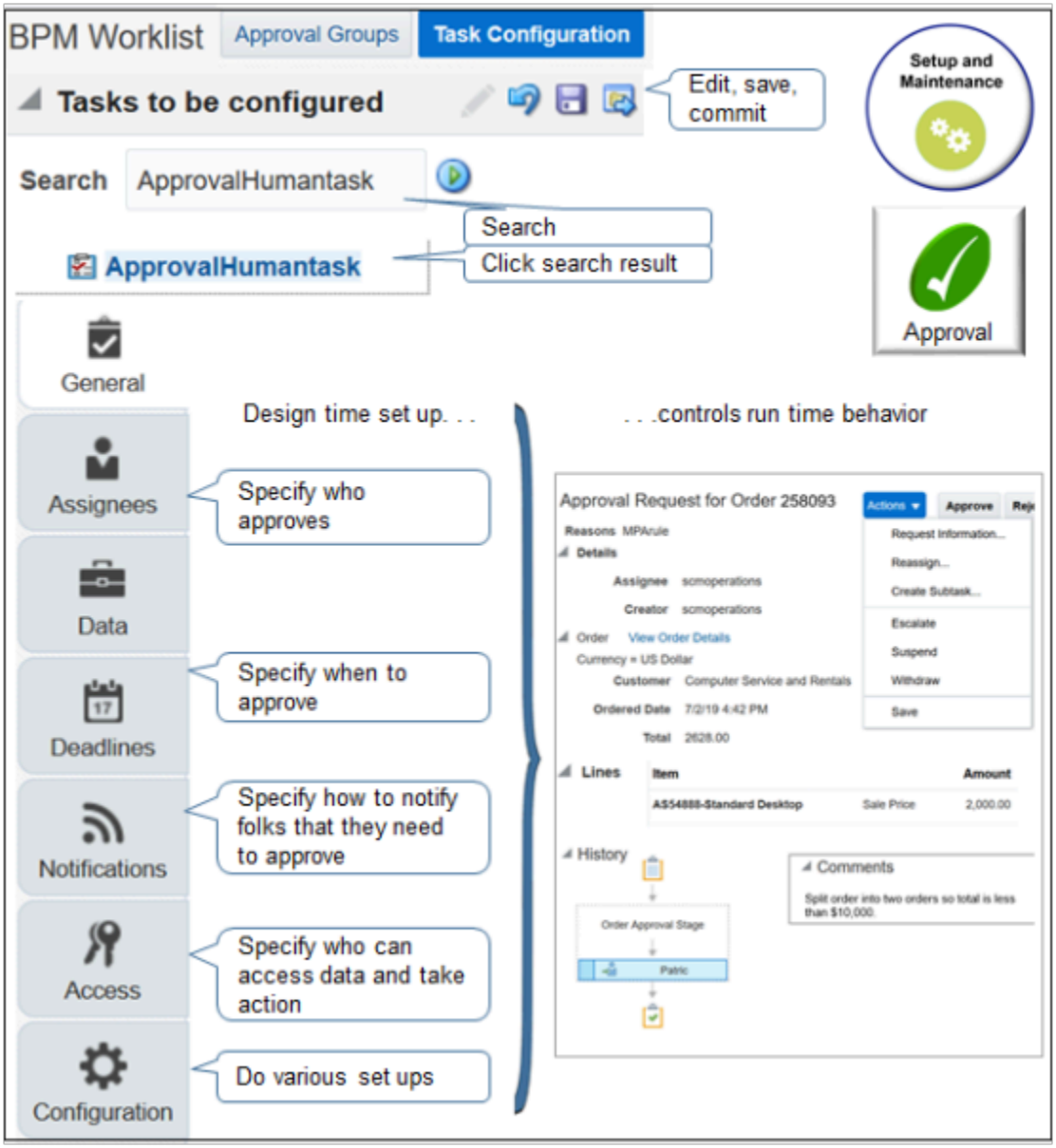
Guidelines for Setting Up Your Approval Task

Use guidelines to help you set up your approval task.

For details about how to set up other parts of approval, see the Define Approval Management section in *Implementing Common Features for Oracle SCM*.

Set Up Approval Task

An approval is a type of human task. Set it up to control runtime behavior.



Note

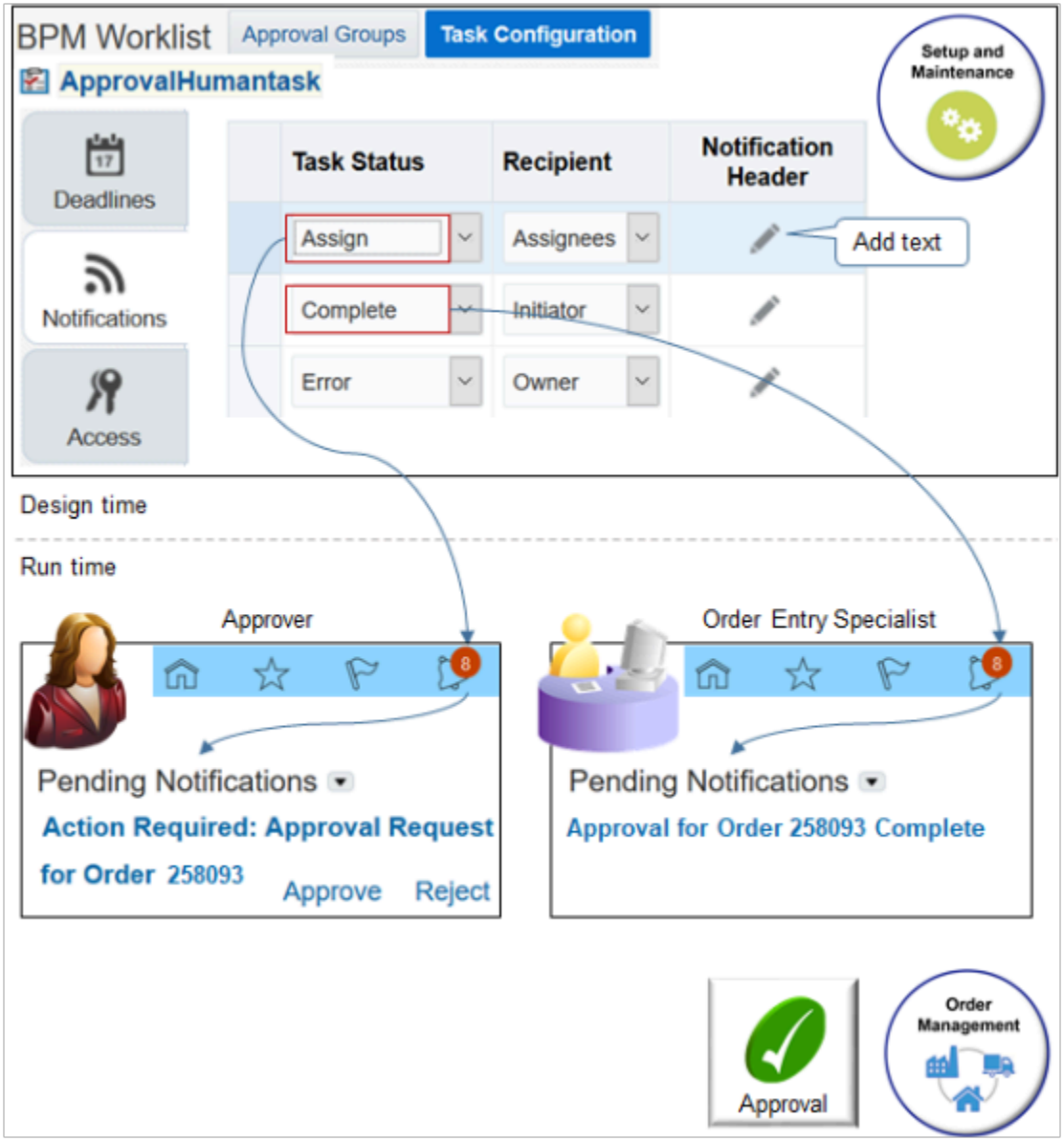
- Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional area: Customers
    - Task: Manage Task Configurations for Supply Chain Management
- On the BPM Worklist page, search for and edit task ApprovalHumantask. Don't edit the TaskApproval task because its for supply orders, not sales orders. Don't edit ApproversTask. Its for another type of flow.

- Use the tabs to specify a range of behavior.

Tab	Description
Assignees	Who approves.
Deadlines	When to approve.
Notifications	How to notify approvers and others in the approval flow. Make sure you only add users you have already set up in an Oracle Application.
Access	◦ Who can access data ◦ Who can take what actions
Configuration	Specify other configurations.

- Use participant when you set up the worklist. Don't use other levels, such as position.

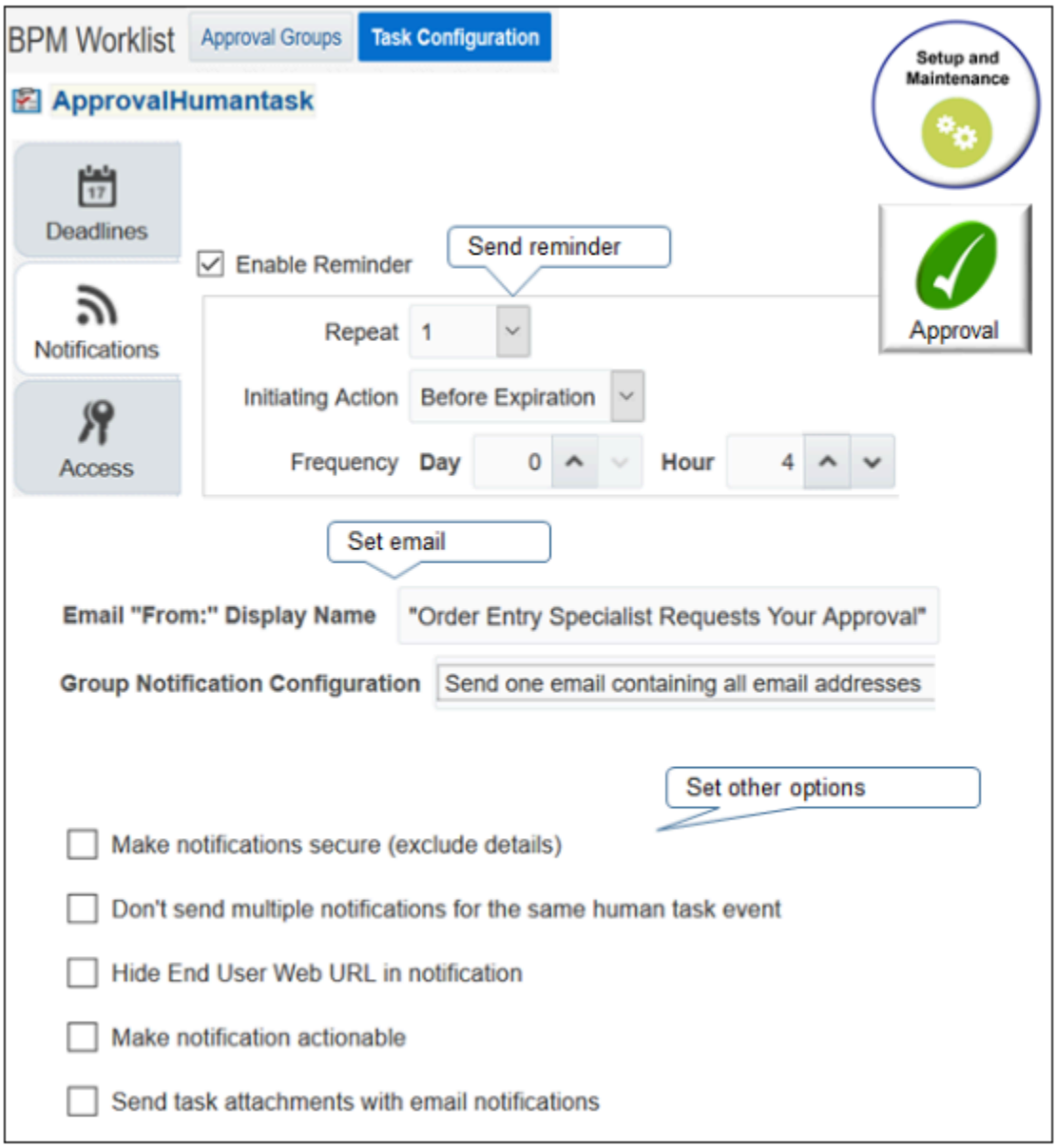
Set Up Notifications



Note

- Use the Task Status list to specify when to send a notification when the approval status changes.
- Example statuses include Assign, Complete, Error, Request Info, Suspend, Withdraw, Resume, and so on.
- Specify who to send the notification to, such as Assignees, Initiator, Approvers, and so on.
- The initiator is the person who creates the sales order and requests approval. The task assignee is the person who approves the request.
- The list comes predefined to send a notification to the task assignee when the flow changes the status to Assign, and to send a notification to the initiator when the flow changes the status to Complete.
- You can change the status, add new statuses, and delete statuses. For example, an approver can Request Information, Suspend, Withdraw, or Resume an approval request. You can send a notification each time the approver takes one of these actions.
- As an option, add text to include in the notification header.

Set Up Other Notifications



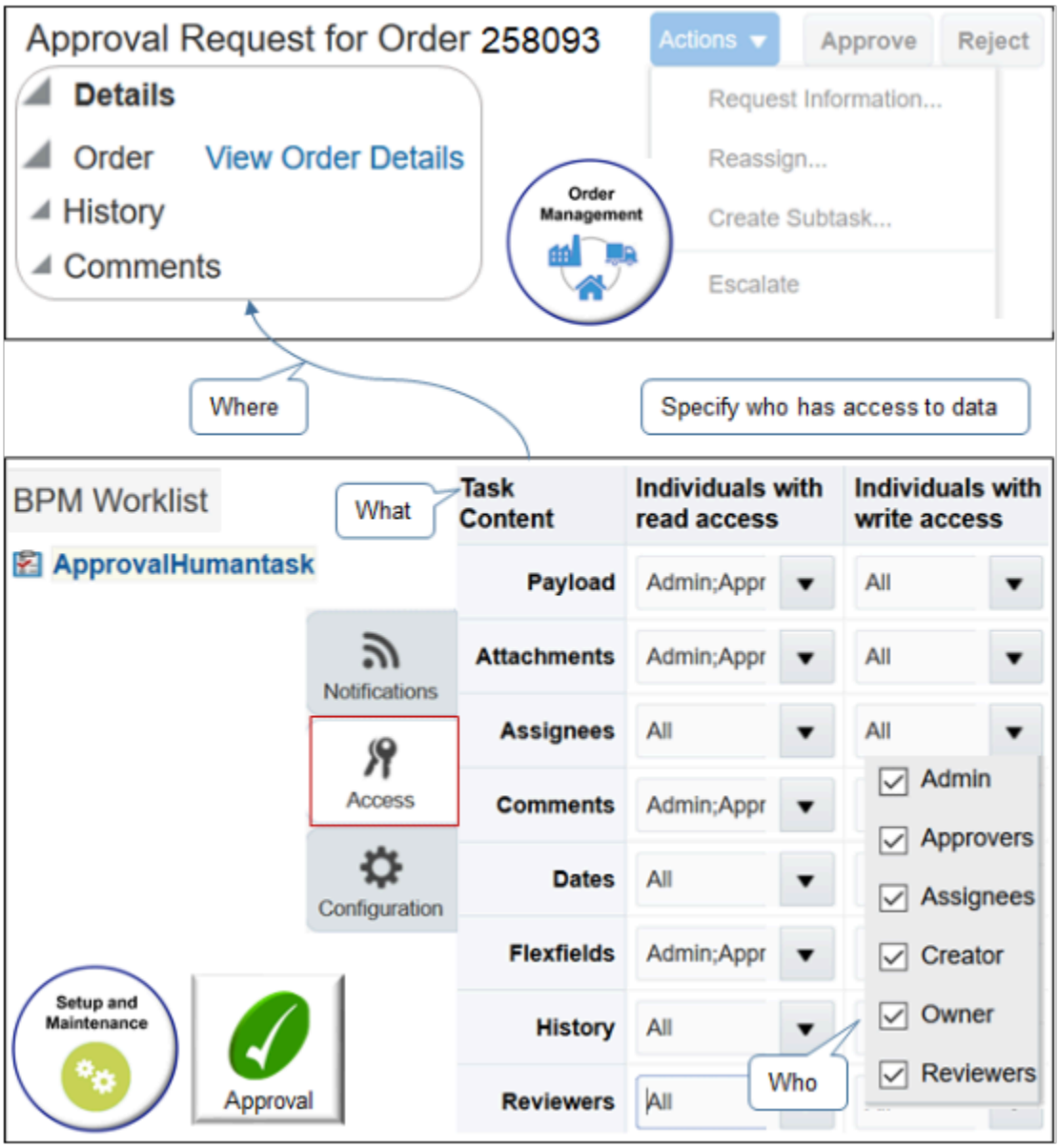
Note

Attribute	Value
Email "From" Display Name	Specify text to include in the From line of the email message. For details, see *Overview of Configurable Email Notifications in SCM*.
Group Notification Configuration	Specify whether to send a separate email to each recipient or to send only one email to all recipients.

Attribute	Value
Make Notifications Secure (Exclude Details)	Set to a value. • **Contains a check mark**. Include only the task number in the notification. • **Doesn't contain a check mark**. Include the task number, task details, approval history, HTML details, attachments, actionable links, and an Approval button in the notification.
Don't Send Multiple Notifications for the Same Human Task Event	Send only one approval request for each sales order. Assume your order includes two approvals. • Rule x on line 1 sends approval to Diane Cho. • Rule y on line 2 also sends approval to Diane. Enable this attribute so approval sends only a single request for the entire order instead of sending a separate request for each rule.
Make Notification Actionable	Set to a value. • **Contains a check mark**. Let the recipient take action directly in the notification. • **Doesn't contain a check mark**. Don't let the recipient take action directly in the notification.

Specify Access

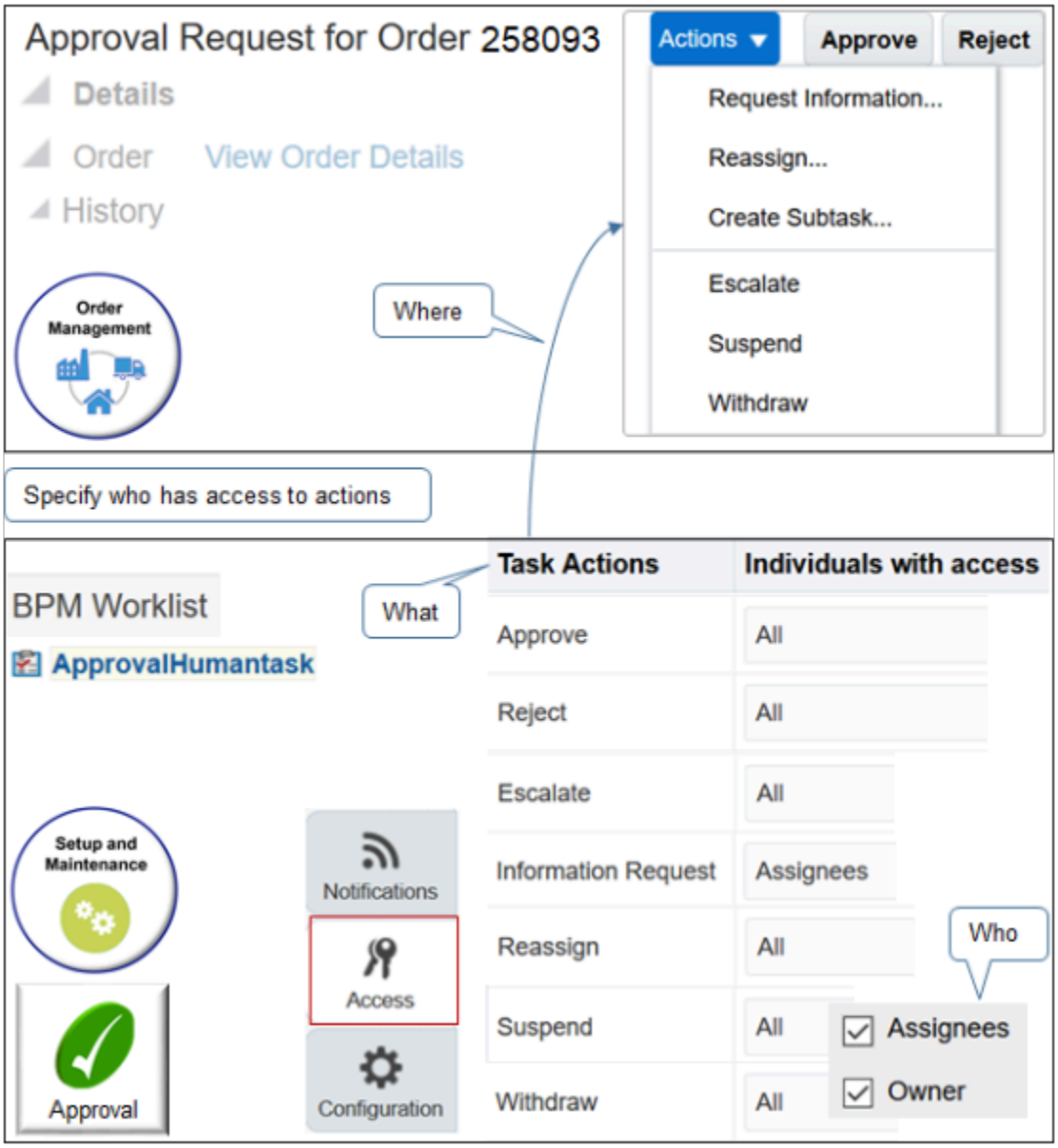
Use the Access tab to specify who can see what, where.



Note

- Access determines visibility of data, such as the history section and comment section of the approval request in Order Management.
- Specify who, such as administrator, approver, assignee, creator, owner, or reviewer.
- Specify data, such as attachments, assignees, comments, dates, and history.
- Specify what the user can see, such as read attachments or read comments.

Use the Actions area of the Access tab to specify who can do what, where.



Note

- Access determines visibility of actions in the approval request.
- Specify who, such as assignee or owner.
- Specify what the user can do, such as approve, reject, request information, reassign, escalate, suspend, or withdraw.

For details, see *Set Up Approval Tasks for Sales Orders* and *Overview of Setting Up Approval*.

Specify the Configuration

Attribute	Description
Allow All Participants to Route Task to Other Participants	Allow a participant to send the approval request to another user. Assume you're an approver and have a large backlog of approval requests. You need to reassign the request to another approver. Enable this attribute so the approver can route the approval to another participant.
Allow Participants to Edit Future Participants	Allow a participant in the approval flow to add another downstream participant.
Allow Initiator to Add Participants	Allow the person who creates the sales order to add an approver or reviewer to the approval request.
Mandate Comments Before Updating These Outcomes	Make comments required and use them to establish an audit trail.
Complete Parallel Subtasks Early Complete Parent Tasks of Early Completing Subtasks	Use a subtask to separate an approval request into different parts. Assume you work at company headquarters and receive a large order with a quantity of 1,000. You are authorized to approve large quantities for the customer but you first must make sure you can deliver the quantity by the delivery date. Create a subtask and approval will send a request to someone in your warehouse or factory to verify they can meet the delivery deadline.

Related Topics

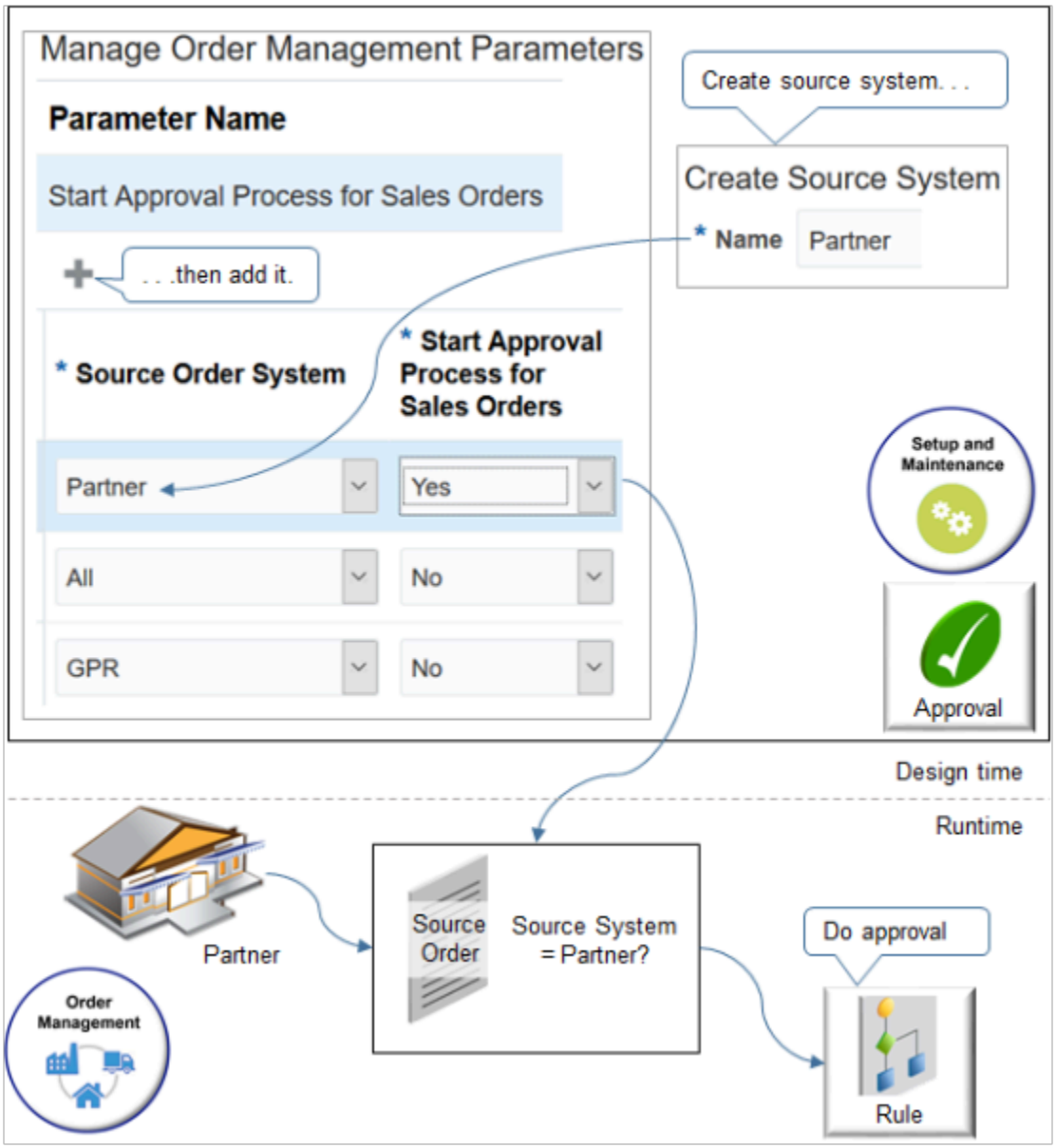
- Overview of Using Business Rules With Order Management
- Manage Order Management Parameters
- Set Up Approval Tasks for Sales Orders
- More Setup for Workflow Email Notifications

More Guidelines for Setting Up Approval

Use guidelines to help you set up approval.

Set Up the Order Management Parameter

Use the Start Approval Process for Sales Orders parameter to enable approval and specify the source system where you need approval.



For example, assume your in house staff has authority to set discounts that exceed 10%, and they use the Order Management work area to create sales orders. You also receive orders from a partner, but the partner isn't authorized to provide these discounts.

Note

- Go to the Setup and Maintenance work area.
- Search for the Manage Source Systems task, then use the Create Source System page to create your source system.

  For details, see *Integrate Order Management with Source Systems*.
- Use the Manage Order Management Parameters page to add your source system to the Start Approval Process for Sales Orders parameter. For details, see *Manage Order Management Parameters*.

- At run time, Order Management will do approval for each source order where the Source System attribute equals Partner.
- In this example, the All source system and GPR source system are each No, so the rule runs only when the source system is Partner.

  If you set the Start Approval Process for Sales Orders attribute to No, then the flow doesn't do approval and proceeds directly to fulfillment.

Trade Compliance

- You can create a condition for trade compliance, such as trade compliance status, screening type, compliance type, result, or reason.
- If you need approval for trade compliance, and if the trade compliance status is Hold or Failed, then, as an option, you can route the sales order for approval. Its a business decision whether to reject the sales order or release it to order fulfillment.

  For example, if the hold happens because of a licensing effective date, then the Order Manager can select to release the sales order to fulfillment, then manage the license problem at some later time.

Information Only

Use the Information Only action to keep someone who needs to know about the approval informed but doesn't actually approve. For example, assume user June Tsai and user Lily Cox need to approve the sales order, and user Diane Cho doesn't approve but needs to get updates about the approval. Here's what you need to do:

1. Create a group and assign June and Lily to the group. Assume you name the group MyGroup.
2. Do assignments in the Do statement of your approval rule.

```
Assign to Group MyGroup, Approval Required
Assign to Individual, Information Only, Assign to Diane Cho
```

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Order Management Parameters
- Set Up Approval Tasks for Sales Orders
- More Setup for Workflow Email Notifications

Procedures

Set Up Approval Rules for Sales Orders

Set up an approval rule that routes a sales order to one or more approvers.

Assume you must need to create this routing rule:

- If the order total is greater than $10,000, and if the Primary is yes, and if the customer is Computer Service and Rentals, then request and get approval from a manager.

Here's the rule that you will create.

Summary of the Set Up

1. Create the approval rule and the IF.
2. Create the first AND.
3. Create the second AND.
4. Create the DO.
5. Activate and publish your rule.
6. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Create the Approval Rule and the IF

1. Make sure you have the privileges that you need to administer Order Management.
2. Set the Start Approval Process for Sales Orders parameter to Yes for all source systems.

   For details, see *Manage Order Management Parameters*.

3. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Approval Rules
4. On the Manage Order Approval Rules page, click **Create New Rule**.
5. Change the rule name.

Current Name	New Name
New Rule	Get Approval for High Value Sales Orders

6. Name the If statement so it reflects the meaning of the condition.
    - Click **Enter Description**.
    - In the Enter Description dialog, enter `High Value Sales Orders`, then click **OK**.
7. Click **New Condition** (the dashed circle in the IF area).
8. In the Create Condition dialog, enter `order total`, wait a moment, then click **Order Total Amount (Order Header > Order Totals)**.

    Notice that the dialog displays suggestions when you wait a moment after you finish typing.
9. In the Create Condition dialog, set the values, then click **OK**.

Field	Value
Attribute	Order Total Amount
Condition	greater than (>)
Value	10000

    Notice that the IF area now displays the condition.
    - `Order Total Amount > 10,000`
10. Click **Save**.

Create the First AND

1. Click **AND**.
2. In the Create Condition dialog, enter `primary`, wait a moment, then click **Primary (Order Header > Order Totals)**.

3. In the Create Condition dialog, set the values, then click **OK**.

Field	Value
Attribute	Primary
Condition	Is equal to
Value	Yes

Notice that the IF area now displays the condition.

- `Primary is equal to Yes`

4. Click **Save**.

Create the Second AND

1. Below `Primary is equal to Yes`, click **AND**.
2. In the Create Condition dialog, enter `customer`, wait a moment, then click **Customer (Order Header)**.
3. In the Create Condition dialog, set the values.

Field	Value
Attribute	Customer
Condition	Equals (=)

4. Click **Search**.
5. In the Search dialog, in the Name attribute, enter `Computer Service and Rentals`, then click **Search**.
6. In the search results, click **Computer Service and Rentals**, then click **OK**.
7. In the Create Condition dialog, click **OK**.
   Notice that the IF area now displays the condition.
   - `Customer = Computer Service and Rentals`
8. Click **Save**.

Create the DO

1. Click **THEN > DO**.
2. Name the DO so it reflects the meaning of the action.
   - Click **Enter Description**.
   - In the Enter Description dialog, enter `Assign to Order Manager`, then click **OK**.
3. Click **New Action** (the dashed circle in the DO area), then click **Perform an Action**.
4. In the Create Action dialog, set the action to `Assign to Job Level`.

5. In the Create Action dialog, verify the values.

Field	Value
Action	Assign to Job Level
Required Action	Approval Required
Approval Chain Based On	Task Creator
First Approver	Manager
Minimum Job Level of Top Approver	1
Utilized Approvers	All Levels

6. Click **Search**.
7. In the Search dialog, next to Users, enter `manager`, then click **Search**.
8. In the Search for Highest Possible Approver dialog, in the manager row, click the **radio button**, then click **OK**.
9. Verify that Highest Possible Approver contains `manager`, then click **OK**.

   Notice that the DO area now displays the condition.

   - `Assign to Job Level`
10. Click **Save and Close**.

Activate and Publish Your Rule

1. On the Manage Order Approval Rules page, notice that the Get Approval for High Value Sales Orders rule is grey.
2. Click **Get Approval for High Value Sales Orders**.
3. In the Get Approval for High Value Sales Orders dialog, add a check mark to the Activate Rule option, then click **Save and Close**.

   Notice that the Active indicator for Get Approval for High Value Sales Orders is green.
4. Click **Publish**.

   Order Management publishes the active rules that display on the Manage Order Approval Rules page.

Test Your Set Up

1. Make sure you have the privileges that you need to create sales orders, go to the Order Management work area, create a sales order, then click **Submit**.

Attribute	Value
Customer	Computer Service and Rentals

Attribute	Value
Amount	Add items to an order line so that the Amount exceeds $10,000.

2. Verify that Order Management updates the order status to Approval Pending.
3. Click **View Approval Information**, then verify the approval details reflect the approval rule you created.

4. In a separate browser session, Make sure you have the privileges that you need to manage sales orders.
5. Open the Pending Notifications page, locate the link for the approval request, such as **Action Required: Approval Request for Order 258093**, then click **Approve**.
6. Navigate to the browser session for the Order Entry Specialist, refresh the page, then verify that the order status is now Processing.

Attributes That You Can Use with Approval Rules

If you enter the first few letters of an attribute, then the rules editor will automatically populate the list with the attributes that start with those letters. However, you might find it useful to see all the attributes that you can use with an approval rule. We hope you find this information useful.

Order Header and Order Line

You can use these attributes on the order header and the order line.

Order Header	Order Line
Business Unit	Asset Tracked (Flag)
Cost of Change	Cancellation Effective Date
Created By	Configuration Trade Compliance Status
Customer	Configured (Flag)
Freeze Price (Flag)	Customer Item
Open (Flag)	Inventory Transaction (Flag)
Order Change Version	Item
Order Type	Item Type
Ordered Date	Line Type
Packing Instructions	Number for Covered Customer Product
Payment Term	Number for Covered Product
Pricing Strategy Name	Order Line Category
Primary Salesperson	Order Line Trade Compliance Status
Purchase Order	Ordered Quantity
Sales Agreement	Ordered UOM
Sales Channel	Over Fulfilment Tolerance
Shipment Priority	Payment Term
Shipping Instructions	Priced in Secondary UOM
Source Document Type	Primary Salesperson
Source Order System	Product Category
Subinventory	Project Number
Trade Compliance Status	Project Record Indicator
Transaction Currency	Purchase Order
Warehouse	Purchase Order Line
	Requested Arrival Date
	Requested Ship Date
	Return Reason
	Return Type
	Return without a reference order
	Returnable (Flag)
	Sales Agreement
	Sales Product Type

Order Header	Order Line
	Secondary Ordered Quantity Secondary UOM Selling Profit Center Business Unit Shipment Priority Source Business Unit Subinventory Subscription Profile Tax Classification Code Under Fulfillment Tolerance Warehouse

Order Line Details and Order Totals

You can use these attributes for the order line details and order totals.

Order Line Details	Order Totals
Trade Compliance Comment Trade Compliance Status Trade Compliance Type Trade Control Type	Order Total Amount Order Total Type Primary

Pricing on the Order Line

You can use these attributes to specify pricing on the order line.

Charges	Charge Components	Price Adjustments	Price Validation
Line Charge Name Line Charge Periodicity Line Rollup Charge	Extended Charge Amount Price Element Name Price Source Type Unit Charge Amount	Charge Charge Type Manual Adjustment Basis Manual Adjustment Reason Manual Adjustment Type Manual Adjustment Value Periodicity Rollup Charge Flag	Price Validation Code Validation Charge Component or Total Validation Charge Periodicity Validation Charge Type Validation Rollup Charge Flag

Charges	Charge Components	Price Adjustments	Price Validation

Related Topics

- Manage Order Management Parameters
- Overview of Setting Up Approval
- Get Approvals for Sales Orders

Set Up Approval Tasks for Sales Orders

An approval is a type of task. You can set it up so it meets your needs.

Here's your scenario.

- Diane Cho needs to approve each approval request and Li Yu needs to review it.
- The approver must approve the request within one day.
- Send a notification to the.
    - Assignee when approval assigns the task.
    - Initiator when approval is done.
    - Owner when an error happens.

  Include text in the notification that approval sends to the approver.

  ```
  Please approve the request for this sales order.
  ```
- Send a reminder to the approver one hour before the approval request expires.
- Include text in the From line of the email notification.

  ```
  Order Entry Specialist Requests Your Approval
  ```
- Send only one email to all users who participate in the approval request. Don't send separate emails.
- Allow everyone to read attachments and comments, but don't allow reviewers to edit them.
- Allow the assignee and owner to send an information request, but allow only the assignee to approve it.
- Assume your company policy requires an audit trail, so make sure each approval and each rejection contains a comment.

Summary of the Setup

1. Set up assignees and deadlines.
2. Set up notifications.
3. Set up access.
4. Set up configuration.

Set up Assignees and Deadlines

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management

- Functional Area: Customers
- Task: Manage Task Configurations for Supply Chain Management

2. On the BPM Worklist page, search for ApprovalHumantask.
3. In the search results, click **ApprovalHumantask**, then click **Edit Task**.
4. Click **Assignees**, then set the values.

Attribute	Value
Task Owner	Diane Cho
Reviewers	Li Yu

5. Click **Deadlines**, then set the values.

Attribute	Value
Due Date	Contains a check mark
Day	1 The approval request will expire one day after approval sends the request to the approver.
Hour	0
Minutes	0

Set Up Notifications

1. Click **Notifications**.

   Notice the Task Status list comes predefined with a separate row for three different statuses.

Task Status	Recipient
Assign	Assignees
Complete	Initiator
Error	Owner

2. Add a notification message.
    - In the assign row, in the Notification Header column, click the **pencil**.
    - In the Edit Notification Message dialog, enter text, then click **OK**.

      ```
      Please approve the request for this sales order.
      ```

      The notification that approval sends to the approver will include this text.
3. Add a reminder.
    - Click **Enable Reminder**, set the values, then click **Save**.

Attribute	Value
Repeat	1
Initiating Action	Before Expiration Approval will send the reminder before the deadline that you set earlier in this procedure expires.
Frequency	Day 0 Hour 4 Minutes 0 Approval will send the reminder four hours before the deadline expires. You set the deadline at 1 day, so approval will send the reminder 20 hours after approval sends the request to the approver.

4. Expand **More**, then set the values.

Attribute	Value
Email "From:" Display Name	Add text. `"Order Entry Specialist Requests Your Approval"` Approval will add your text to the From line of the email notification that it sends to the approver. You must enclose your text with double quotation marks (" ").
Group Notification Configuration	Specify how to display the email message. You can. - Send individual emails.

Attribute	Value
	○ Send individual emails and include a form in each one. ○ Send one email to all recipients.

Set Up Access

1. Click **Access**, then set the values.

Task Content	Individuals with Read Access	Individuals with Write Access
Attachments Comments	Make sure each option contains a check mark. ○ Admin ○ Approvers ○ Assignees ○ Creator ○ Owner ○ Reviewers	Make sure only the Reviewers option doesn't contain a check mark.

2. Expand **Actions**, then set the values.

Task Action	Individuals with Access
Information Request	Make sure each option contains a check mark. ○ Assignees ○ Owners
Approve	Make sure only the Assignees option contains a check mark.

Set Up Configuration

1. Click **Configuration**, then set the values.

Attribute	Value
Mandate Comments Before Updating These Outcomes	Make sure each option contains a check mark. ○ Approve ○ Reject

2. click **Save**, then click **Commit Task**.

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Order Management Parameters
- More Setup for Workflow Email Notifications

Set Up Parallel Approval Requests for Sales Orders

Set up approvals so you can send an approval request to all members of an approval group at the same time.

As an alternative, you can use the first approver who replies to represent the entire group. If the first reply:

- Approves the request. Order Management proceeds with fulfillment.
- Rejects the request. Order Management changes the order status to Draft and discards the approval request.

You can also use a voting percent according to the response from the first person who replies to the approval request or according to a consensus of responses from more than one approver.

- Reduce the time you need to approve a request. Its particularly helpful when you have a group of folks who can do the approval. You'll no longer have to wait for each approver to reply sequentially before proceeding to the next approver. Instead, do it in parallel.
- Allow the first person who replies to approve the request for the entire group. It can significantly reduce the time needed to get the approval done.
- You can use any attribute that you use in a sequential approval rule in a parallel rule.

Create a Staged, Parallel Flow

Assume you need to do approval in three stages:

- Stage one starts the approval process.
- Stage two needs a simple majority consensus of approvers to move to stage three. Approvers in stage two can include any person that has the privileges from the Supply Chain Operations Manager, Order Entry Specialist, or Order Administrator job role.
- Anyone in stage two can approve independently of anybody else, approval can be done in parallel, but approval won't move to stage three until at least 51% of the job roles approve. In this example you have three job roles in stage two, so if two of them approve, that's a 66% approval rating, and approval will move to stage three.
- Stage three only needs one person to approve. Approvers in stage three can include any person that has the Buyer or the Order Manager Operations job role. As soon as one person approves, the entire request is approved.

Assume you enter the text Requires Approval in the Purchase Order attribute on the order header to indicate that the order requires approval.

Summary of the Setup

1. Create your approval groups.
2. Add your stages.
3. Create your approval rules.
4. Test your setup.

Note

- You will set up a workflow. There's a lot to know. For details, go to *Implementing Applications*, then expand **Workflow Approvals and Notifications > Configure Workflow Approvals and Notifications**.
- This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.
- This topic uses example values. You might use different values, depending on what you need.

Create your Approval Groups

You create one group for stage two and another one for stage three.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Customers
    - Task: Manage Task Configurations for Supply Chain Management
2. On the BPM Worklist page, enter `order` in the search window, click **Search Task Types**, wait for the search to finish, then click **ApprovalHumantask**. For details, see *Guidelines for Setting Up Your Approval Task*.
3. Click **Assignees**, then click **Approval Groups**.
4. In the Groups Area, click **Create Approval Group** (the plus icon), then set the value.

Attribute	Value
Name	Order Managers

5. Add a member.
    - Click **Add Member** (the plus icon on the far right side).
    - In the Add to Group dialog, search for the value, then click **OK**.

Attribute	Value
ID	supply_chain_operations_manager Use advanced search, as necessary.

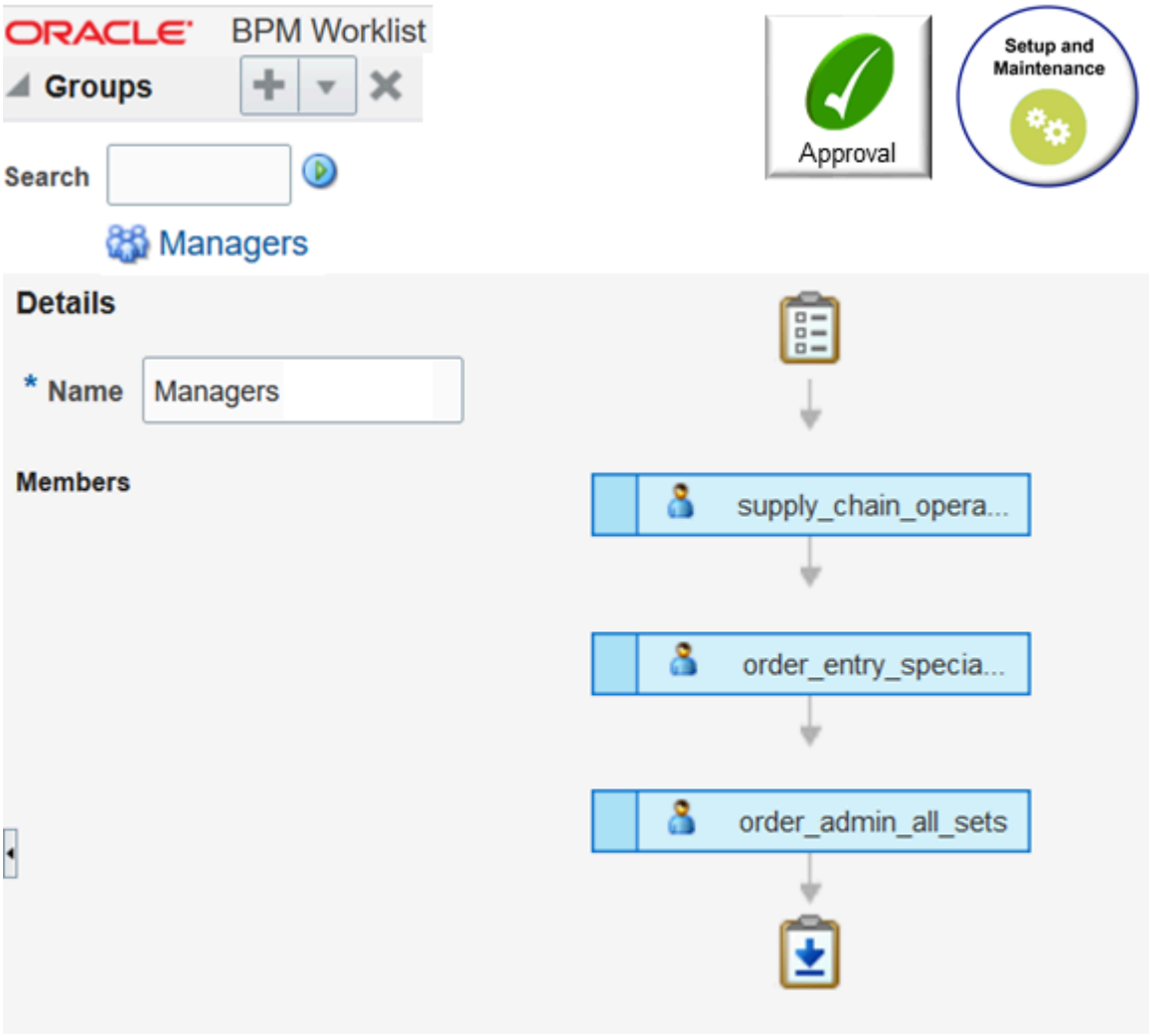
6. Add another member. Repeat the step you just did.

Attribute	Value
ID	order_entry_specialist

7. Add another member.

Attribute	Value
ID	order_admin_all_sets

8. Click **Save**, then verify your group. It should look like this.



9. Create another group.
   - Create the group.
Attribute	Value
Name	Buyers
   - Add members to the Buyers group, then click **Save**.
     - cv_buyer
     - order_mgr_operations

Add Your Stages

1. Click **Task Configuration**.
2. In the Tasks to be Configured area, enter `order`, then click **Search Task Types**.
3. In the search results, click **ApprovalHumanTask**, then click **Edit Task** (the pencil next to Tasks to be Configured).
4. Add a stage.
    - Click **Assignees**.
    - In the Assignees area, notice that it displays one stage, and its name is Order Approval. Don't mess with this stage.
    - Notice that the stage has an inner box and an outer box. The inner box is a participant and the outer box is the stage. You click the inner box to add a participant. You click the outer box to add a stage.
    - Click the **participant** in the new stage (the inner box), then give it a name in the details area.

Attribute	Value
Participant Name	Temporary This attribute doesn't have a label. There's only a text box. Replace the text in the box. For example, replace New.Participant294 with Temporary.

    - Click the **participant** in the new stage (the inner box), click the **pencil**, then click **Add Participant > Add Sequential > Parallel**. Notice that your stage now has two participants, they're sequential, and that details of the New.Particpant area has a Voting tab.
    - Name the new participant in the details area.

Attribute	Value
Participant Name	Managers This attribute doesn't have a label. There's only a text box. Replace the text in the box. For example, replace New.Participant626 with Order Managers.

    - Click the **Temporary** participant in your new stage, click the **pencil**, then click **Delete**. You created this participant only so you could add the second, sequential participant. You have to add at least one sequential participant to get the Voting tab to display.
    - Click the **Order Managers** participant, then, in the Order_Managers area, set the values.

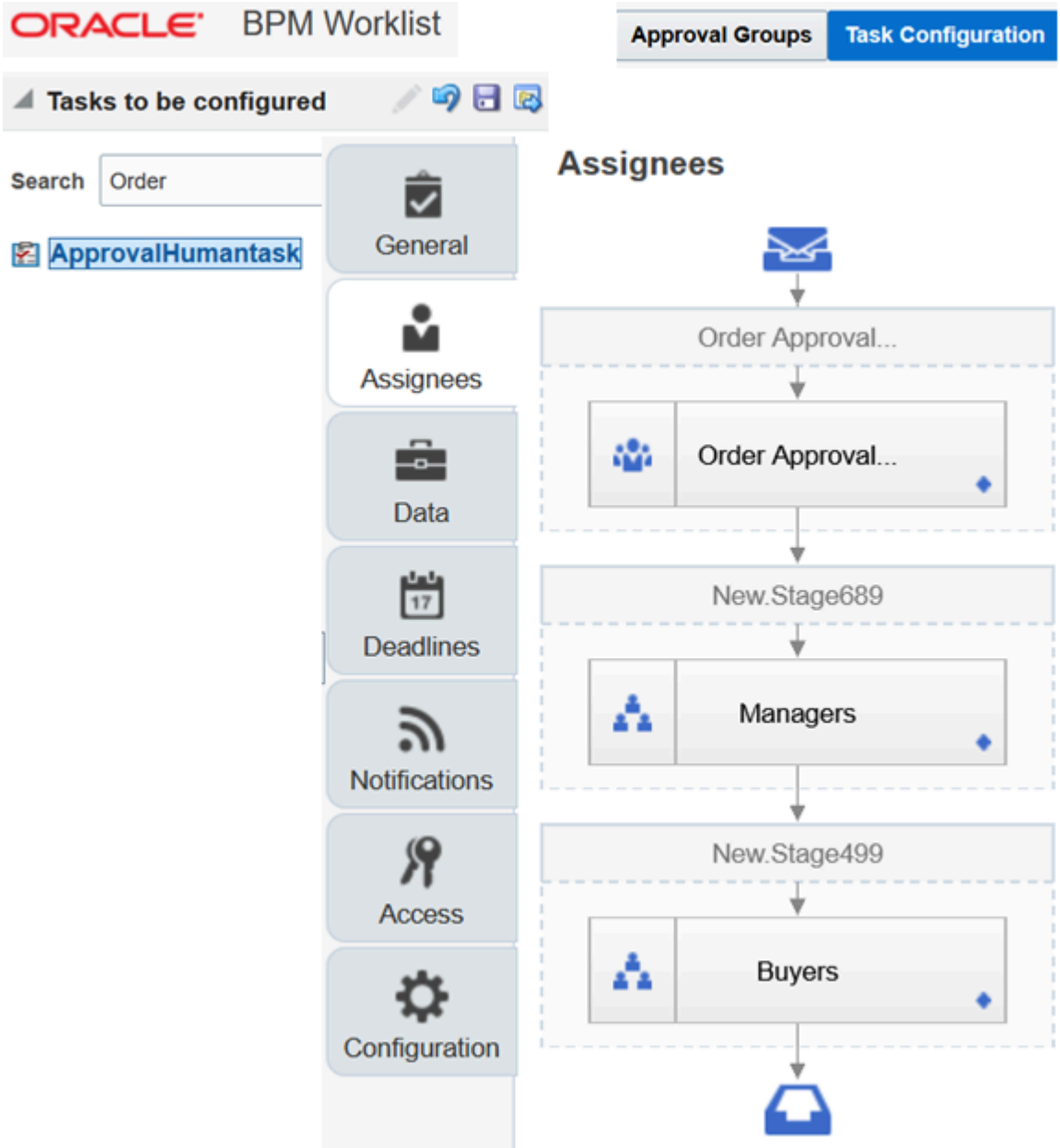
Attribute	Value
Assignees Based On	Rule-based
Business Rule	Rule_for_Managers You can enter any text. Click Create Rule after you enter the text.

- Click **Voting**, set the values, then click **Save**.

Attribute	Value
Any	51%
Default Outcome	Approve
Outcome Trigger Policy	Immediately Trigger Voted Outcome When Minimum Percentage is Met

5. Add another stage.
   - Click the **outer box** of the stage you just created, click the **pencil**, then click **Add Stage > Add Sequential Stage**.
   - Repeat the steps you did to add the first stage.
   - Make sure to add a temporary participant, add a sequential one after it, then delete the temporary.
   - Set the Participant Name to Buyers instead of Managers.
   - Set the name of the rule to Rule_for_Buyers instead of Rule_for_Managers
   - Set the voting to 1% instead of 51%.

Your flow should look like this.



Note

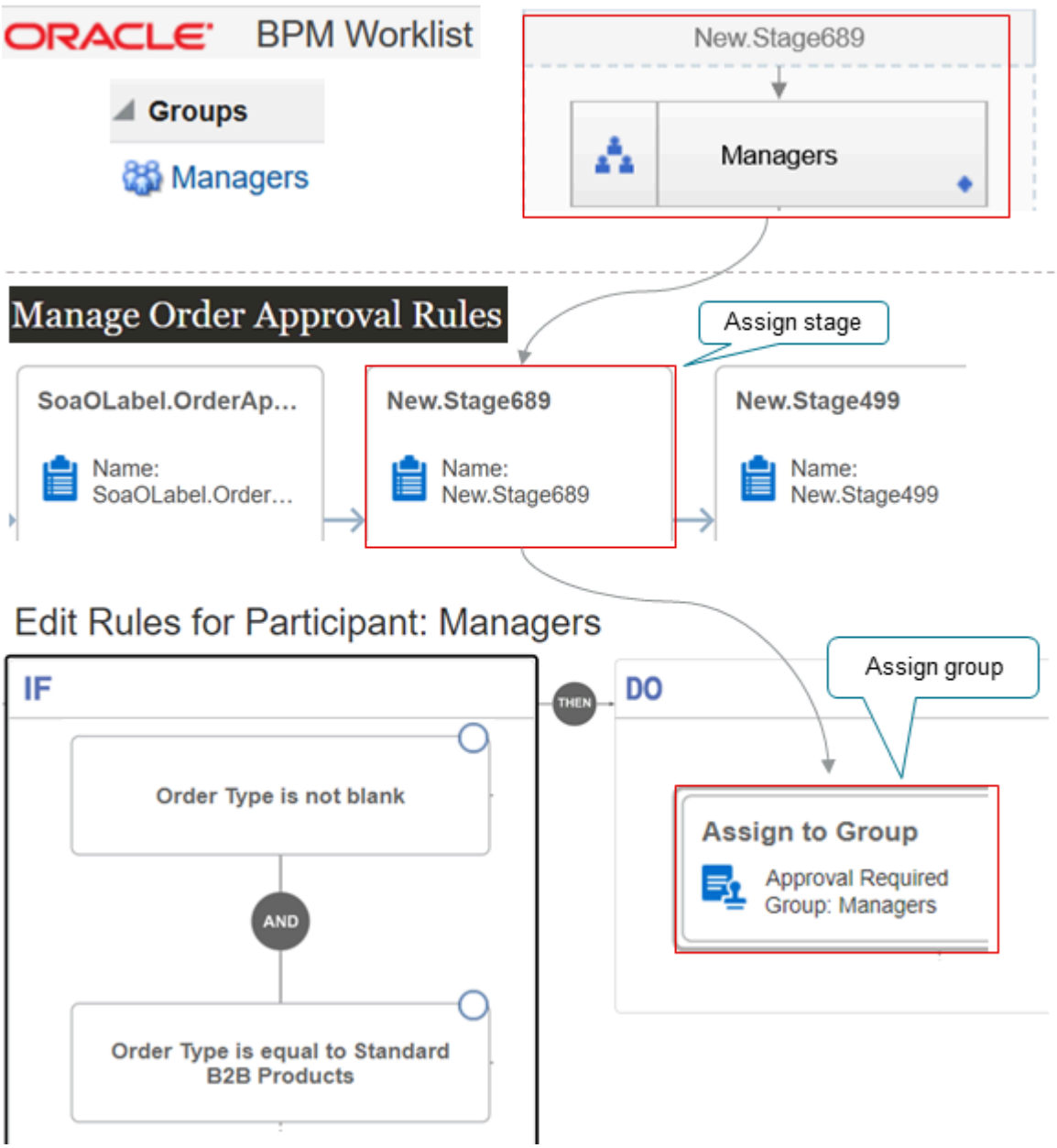
- PBM Worklist creates a unique name for each stage. Notice the stage names in this example, such as New.Stage689 and New.Stage499. You will reference them in the next section of this procedure.
- To get the flow to display the name of the approver, you might have to add a temporary approver after the Buyers approver, then delete that temporary.
- We're displaying the flow vertically so it fits this page and you can read it. You can also display it horizontally.

6. Click **Commit Task**.

You must commit so you can reference your stages and groups in the approval rules that you create next.

Create Your Approval Rules

Next, set up an approval that references the group and the stage that you just created.



Note

- You assign Stage689 to the Managers group.
- You assign Stage499 to the Buyers group.

At runtime, Order Management runs Stage689 first. It proceeds to Stage499 only if 51% of the managers in Stage689 approve the request.

- Always make sure an attribute isn't empty before you examine it. You do this to avoid a null pointer error. If it turns out that it is empty, and you attempt to examine, then you might get that runtime error.
- We're using the Standard B2B Products order type to illustrate the flow. You might need a different type.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Customers
    - Task: Manage Order Approval Rules
2. On the Manage Order Approval Rules page, notice that it has three stages:

Stage	Description
SoaOLabel.OrderApprovalStage.SalesOr	This is always the first stage. You'll see it for any flow that you set up. You use it to create the condition that starts the approval flow.
New.Stage689	This is the manager stage that you created in BPM Workflow.
New.Stage499	This is the buyer stage that you created in BPM Workflow.

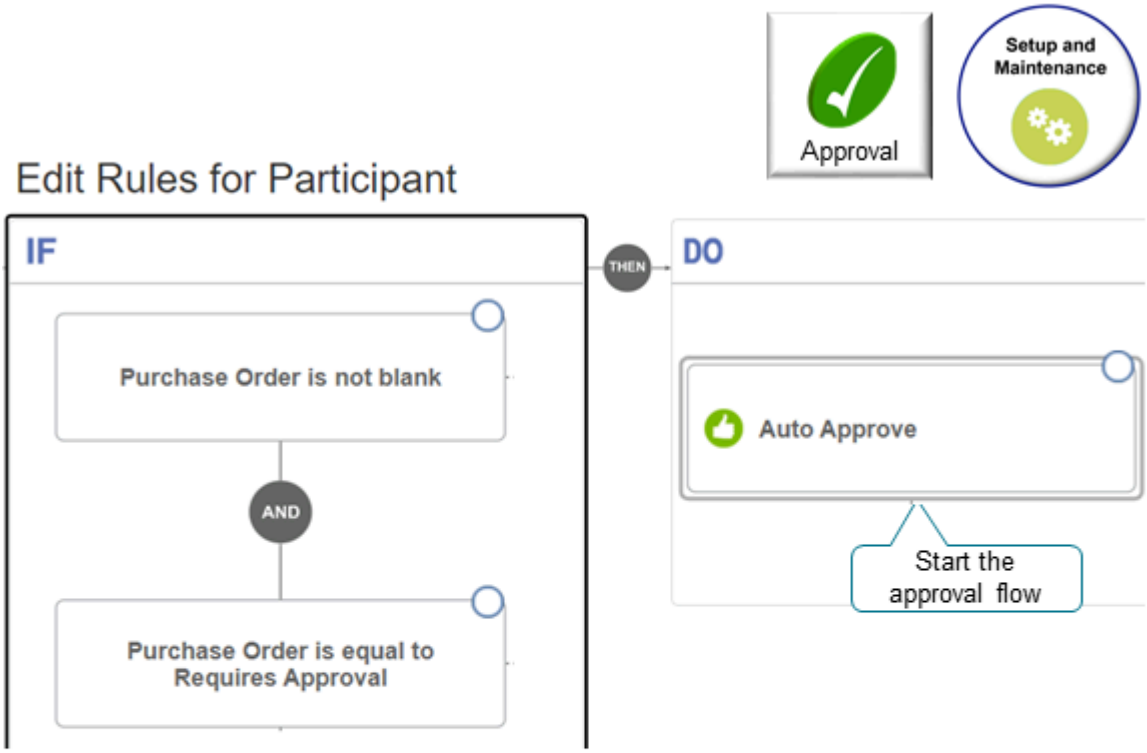
3. Create the condition that starts the approval flow.
    - Click **SoaOLabel.OrderApprovalStage.SalesOrderApprovalStage**, and watch it expand.
    - In the expanded step, click **ParticipantType**.
    - In the dialog that displays, click **Actions > Edit Rules**.
    - On the Edit Rules for Participant page, click **Create New Rule**, then set the value.

Attribute	Value
Name	Condition That Starts the Approval Flow

    - Create your rule.

```
If the Purchase Order attribute on the order header isn't blank, and if the Purchase Order attribute on the order header is equal to Requires Approval, then do Perform an Auto Approve
```

      The Auto Approve action starts the approval flow. For example:


Approval
Setup and Maintenance
Edit Rules for Participant
IF
THEN
DO
Purchase Order is not blank
AND
Purchase Order is equal to Requires Approval
Auto Approve
Start the approval flow

4. Create a rule for the order manager stage.
    - Click **New.Stage689**, and watch it expand.
    - In the expanded step, click **ParticipantType**.
    - In the dialog that displays, click **Actions > Edit Rules**.
    - On the Edit Rules for Participant page, click **Create New Rule**, then set the value.

Attribute	Value
Name	Approval Rule for the Managers Stage

    - Create your new rule.

      ```
      If the Order Type attribute on the order header isn't blank, and if the Order Type attribute on the order header is equal to Standard B2B Products, then do Perform an Action Assign to Group Managers
      ```

      See the diagram above for details.
    - Click Save and Close, then activate your rule.
5. Create a rule for the buyer stage. Do the same setup you did to create a rule for the managers but with these differences.
    - Click **New.Stage499** instead of New.Stage689.
    - Set the Name of the rule to Approval Rule for the Buyers Stage.
    - Assign the rule to the Buyers group.
6. Click **Publish**.
7. Enable the Start Approval Process for Sales Orders parameter for your source system. For details, see *Manage Order Management Parameters*.

Note

- If you don't see Actions > Edit Rules when you attempt to add a rule, the check your setup in BPM Worklist and make sure you added a rule to the stage.
- If you modify your setup in BPM Worklist and commit your work, then the Manage Order Approval Rules page deletes your rules and you'll have to create them again.

Test Your Setup

1. Create a sales order.
    - Sign into Oracle Applications as a user that has the Order Entry Specialist job role.
    - Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Sales and Rental
Purchase Order	Requires Approval
Order Type	Standard B2B Products

- Add an order line, then click **Submit**.

  Notice the status in the title bar. For example, Computer Service and Rentals 521966 Approval Pending.

2. Add an approval.
   - Sign out, then sign in with the Supply Chain Operations Manager job role.
   - On the Home page, click **Notifications** (the bell).
   - In the Notifications dialog, verify that you see Approval Request for Order 521966, the status is Action Required, and that there's an Approve button for the request.
   - Click **Approve** to approve the request.
3. Repeat step 2 but sign in with the Order Administrator job role, then sign out.
4. Sign in with the Supply Chain Operations Manager job role, but this time notice that the status for the approval request in the Notifications dialog is Withdrawn, and that there's no Approve button for the request. This means the request is approved for stage two because two other job roles in the group approved request, 66% of the job roles approved, and that exceeds the 51% threshold required to approve.
5. Repeat step 2 but sign in with the Buyer job role. You're now in stage 3, so you should see the Action Required status. Click **Approve**, then sign out.
6. Sign in with the Order Manager Operations job role, go to the Order Management work area, query for sales order 521966, then confirm that the status in the title bar says Processing. Recall that you set the voting percent for stage three to 1%, there are only two job roles in the group that you created for stage three, so 50% of the job roles in the group approved.

Migrate Approval Rules Between Instances of Order Management

Migrate approval rules that you create in one instance of Order Management to another instance of Oracle Order Management.

- You can migrate your rules from one instance of Order Management to another instance of Order Management. For example, you can migrate from your test instance to your production instance.
- Each instance must be on the same update.
- Migration deletes all rules in the target instance, then copies all rules in the source instance to the target. Migration deletes rules in the target even if there's no matching rule in the source.
- Migration copies only the If condition of each rule. It doesn't copy the Then part of the rule. During testing, you typically use the Then part to assign approval to a specific test user. However, you need to replace the test user with an actual user in your production environment, so you need to redo the Then statement after you migrate.
- If even one rule contains an error, then migration fails and it won't migrate any rule.

This topic uses example values. You might need different values, depending on your business requirements.

Migrate approval rules between instances of Order Management.

1. Sign into the Order Management instance where you must create the approval rule, then create the approval rule that you must export.

   Make sure you activate and publish your approval rule.
2. Create the implementation project.
   - Go to the Setup and Maintenance work area.
   - On the Setup page, click **Tasks > Manage Implementation Projects**.
   - On the Implementation Projects page, click **Actions > Create**.
   - On the Basic Information page, set the value, then click **Next**.

Attribute	Value
Name	Project to Migrate Approval Rules You can use any text. Tab out of the Name attribute after you enter the value to populate other attributes.

- On the Select Offerings to Implement page, in the Order Management row, add a check mark to **Include**, then click **Save and Open Project**.
- In the Task Lists and Tasks area, click **Actions > Select and Add**.
- In the Select and Add dialog, set Search to Task, search for Manage Order Approval Rules, then click **Apply > Done**.
- On the Implementation Project page, click **Done > Done**.

3. Create the configuration package you will use to implement the project that you created in step 2.

- On the Setup page, click **Tasks > Manage Configuration Packages**.
- On the Manage Configuration Packages page, click **Actions > Create**.
- On the Create Configuration Package page, in the Source Implementation Project area, set the values.

Attribute	Value
Name	Project to Migrate Approval Rules
Export	Setup Task List and Setup Data

- In the Configuration Package Details area, make a note of the value that displays, then click **Next > Submit**.

Attribute	Value
Name	Project to Migrate Approval Rules_1

- In the Warning dialog that displays, click **Yes**.
- On the Manage Configuration Packages page, in the Project to Migrate Approval Rules_1 area, monitor the status. Click **Refresh** until the value displays.

Attribute	Value
Status	Completed Successfully

Attribute	Value

- In the Download column, click **Download**, then click **Download Configuration Package**.
- Save the file to a location on your computer. Make a note of the location and the file name, such as Project to Export Approval Rules_1_20180310_1805.zip.
- Click **Done**, then sign out of Order Management.

4. Import the rule dictionary that you downloaded in step 4.

- Sign into the Order Management instance where you must import the approval rules.
- Go to the Setup and Maintenance work area.
- On the Setup page, click **Tasks > Manage Configuration Packages**.
- On the Manage Configuration Packages page, click **Upload**, locate the file you downloaded in step 4, click **Get Details > Submit**.
- In the Search Results area, click the **row** that includes the value.

Attribute	Value
Name	Project to Migrate Approval Rules_1

- In the Project to Migrate Approval Rules_1 area, verify the values.

Attribute	Value
Name	Project to Migrate Approval Rules_1
Type	Upload
Status	Completed Successfully

- Click **Import Setup Data**.
- On the Import Setup Data page, in the Import Options area, note the value, then click **Submit**.

Attribute	Value
Process Name	Project to Migrate Approval Rules_1

Attribute	Value

- In the Search Results area, click the **row** that includes the value.

Attribute	Value
Name	Project to Migrate Approval Rules_1

- In the Project to Migrate Approval Rules_1 area, verify values.

Attribute	Value
Name	Project to Migrate Approval Rules_1
Type	Import Setup Data
Status	Completed Successfully

5. Navigate to the Manage Order Approval Rules page, then verify that your approval rules imported successfully.

Related Topics

- Manage Order Management Parameters
- Overview of Setting Up Approval
- Get Approvals for Sales Orders

Get and Display Approval Details for Sales Orders

You can get approval details and use them for your specific needs, such as in an audit report.

Run an SQL query to get the value of the Approved Date attribute and the Approved By attribute.

Assume you need to get details for sales order 514517. To start, get the header ID.

```
select header_id from doo_headers_all where order_number='514517';
```

Assume the query returns a header ID value of 300100246104588. Next, get the task ID.

```
select taskid from FA_FUSION_SOAINFRA.WFTASK where identificationkey like '%300100246104588%';
```

Assume the query returns task ID cdfdb0b2-639d-4cc8-acce-df5031154d8f. Use the task ID to get approval details, such as Approval Date, Comments, Approved By, and so on.

```
select WFCOMMENT from FA_FUSION_SOAINFRA.WFCOMMENTS; where taskid ='cdfdb0b2-639d-4cc8-acce-df5031154d8f'
select * from FA_FUSION_SOAINFRA.WFTASKHISTORY where roottaskid='cdfdb0b2-639d-4cc8-acce-df5031154d8f';
```

Display Approval Details in Your Custom Report

Use the Reports and Analytics work area to run a query.

```
select h.order_number order_number,h.creation_date ,t.ASSIGNEES
assignee,t.version ,t.CREATEDDATE,t.UPDATEDDATE,t.state,t.outcome from
doo_headers_all h,fa_fusion_soainfra.wftask t where
t.identificationkey='DOO'||h.header_id||h.approval_sequence_number
union
select
h.order_number,h.creation_date,a.ASSIGNEE,a.version,a.CREATION_DATE
CREATEDDATE,a.LAST_UPDATE_DATE UPDATEDDATE,t.STATUS_CODE state,t.OUTCOME_CODE
outcome from fnd_bpm_task_assignee a,FND_BPM_TASK_B t,doo_headers_all h where
a.task_id=t.task_id and
t.identification_key='DOO'||h.header_id||h.approval_sequence_number
select h.order_number,h.creation_date,c.COMMENT_TEXT,c.comment_by
from FND_BPM_TASK_COMMENT c,FND_BPM_TASK_B t,doo_headers_all h where
c.task_id=t.task_id and
t.identification_key='DOO'||h.header_id||h.approval_sequence_number
union
select
h.order_number,h.creation_date,c.WFCOMMENT,c.UPDATEDBYDISPLAYNAME
from fa_fusion_soainfra.wfcomments c,fa_fusion_soainfra.wftask
t,doo_headers_all h where c.taskid=t.taskid and
t.identificationkey='DOO'||h.header_id||h.approval_sequence_number
```

Get details about the approver.

```
SELECT dha.order_number order_number,
 dha.header_id ,
 dha.approval_sequence_number ,
 t.IDENTIFICATIONKEY
 ||
 '#####' ,
 dha.creation_date ,
 t.ASSIGNEES assignee ,
 t.version ,
 t.CREATEDDATE ,
 t.UPDATEDDATE ,
 t.state ,
 t.outcome
FROM fusion.doo_headers_all dha,
 fa_fusion_soainfra.wftask t
WHERE t.IDENTIFICATIONKEY LIKE'DOO'
 ||
 dha.header_id
 ||
 '%'
 AND dha.submitted_flag = 'Y'
 AND dha.source_order_number = '&SOURCE_ORDER_NUMBER'
UNION
SELECT dha.order_number order_number,
 dha.header_id ,
 dha.approval_sequence_number ,
 t.IDENTIFICATION_KEY
 ||
 '#####' ,
 dha.creation_date ,
 a.ASSIGNEE ,
 a.version ,
 a.CREATION_DATE CREATEDDATE ,
 a.LAST_UPDATE_DATE UPDATEDDATE,
 t.STATUS_CODE state ,
 t.OUTCOME_CODE outcome
FROM fusion.fnd_bpm_task_assignee a,
 fusion.FND_BPM_TASK_B t ,
 fusion.doo_headers_all dha
WHERE a.task_id =t.task_id
```

```
 AND t.identification_key LIKE'DOO'
 ||
 dha.header_id
 ||
 '%'
 AND dha.submitted_flag = 'Y'
 AND dha.source_order_number = '&SOURCE_ORDER_NUMBER'
```

Get the approver's comments.

```
SELECT dha.order_number order_number,
 dha.header_id ,
 dha.approval_sequence_number ,
 t.IDENTIFICATIONKEY
 ||
 '#####' ,
 dha.creation_date,
 c.WFCOMMENT ,
 c.UPDATEDBYDISPLAYNAME
FROM fa_fusion_soainfra.wfcomments c,
 fa_fusion_soainfra.wftask t ,
 fusion.doo_headers_all dha
WHERE c.taskid (+) =t.taskid
 AND t.identificationkey LIKE'DOO'
 ||
 dha.header_id
 ||
 '%'
 AND dha.submitted_flag = 'Y'
 AND dha.source_order_number = '&SOURCE_ORDER_NUMBER'
UNION
SELECT dha.order_number order_number,
 dha.header_id ,
 dha.approval_sequence_number ,
 t.IDENTIFICATION_KEY
 ||
 '#####' ,
 dha.creation_date,
 c.COMMENT_TEXT ,
 c.comment_by
FROM fusion.FND_BPM_TASK_COMMENT c,
 fusion.FND_BPM_TASK_B t ,
 fusion.doo_headers_all dha
WHERE c.task_id (+) =t.task_id
 AND t.identification_key LIKE'DOO'
 ||
 dha.header_id
 ||
 '%'
 AND dha.submitted_flag = 'Y'
 AND dha.source_order_number = '&SOURCE_ORDER_NUMBER'
```

Related Topics

- Manage Order Management Parameters
- Overview of Setting Up Approval
- Use SQL to Query Order Management Data
- Get Approvals for Sales Orders

Use Extensible Flexfields In Approval Rules

Use an extensible flexfield as part of the condition in an approval rule so you can meet your specific requirements and improve the efficiency of your order approval flow.

Assume some of your customers are government organizations and some aren't. You need to indicate whether the customer is a government organization when you create the sales order, so you create an extensible flexfield segment named Government Customer on the order header.

You then set up an approval rule that does a compliance check to make sure the order meet's the government's various purchasing requirements.

```
If the governmentCustomer attribute on the order header is equal to Yes, then do compliance.
```

Try it.

1. Create an extensible flexfield named Government Customer. For details, see *Overview of Using Extensible Flexfields in Order Management*.
2. Set up approvals.

   Make sure you set the Start Approval Process for Sales Orders parameter to Yes. For details, see *Overview of Setting Up Approval* and *Manage Order Management Parameters*.
3. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Customers
   - Task: Manage Task Configurations for Supply Chain Management
4. In the BPM Worklist section, search for SalesOrder, then click **ApprovalHumantask**.
5. Click **EditTask** (the pencil icon). If you have created or modified your flexfield set up, then you will see a warning message.
6. Click **Start Synchronization** in the message and wait for the warning to disappear.

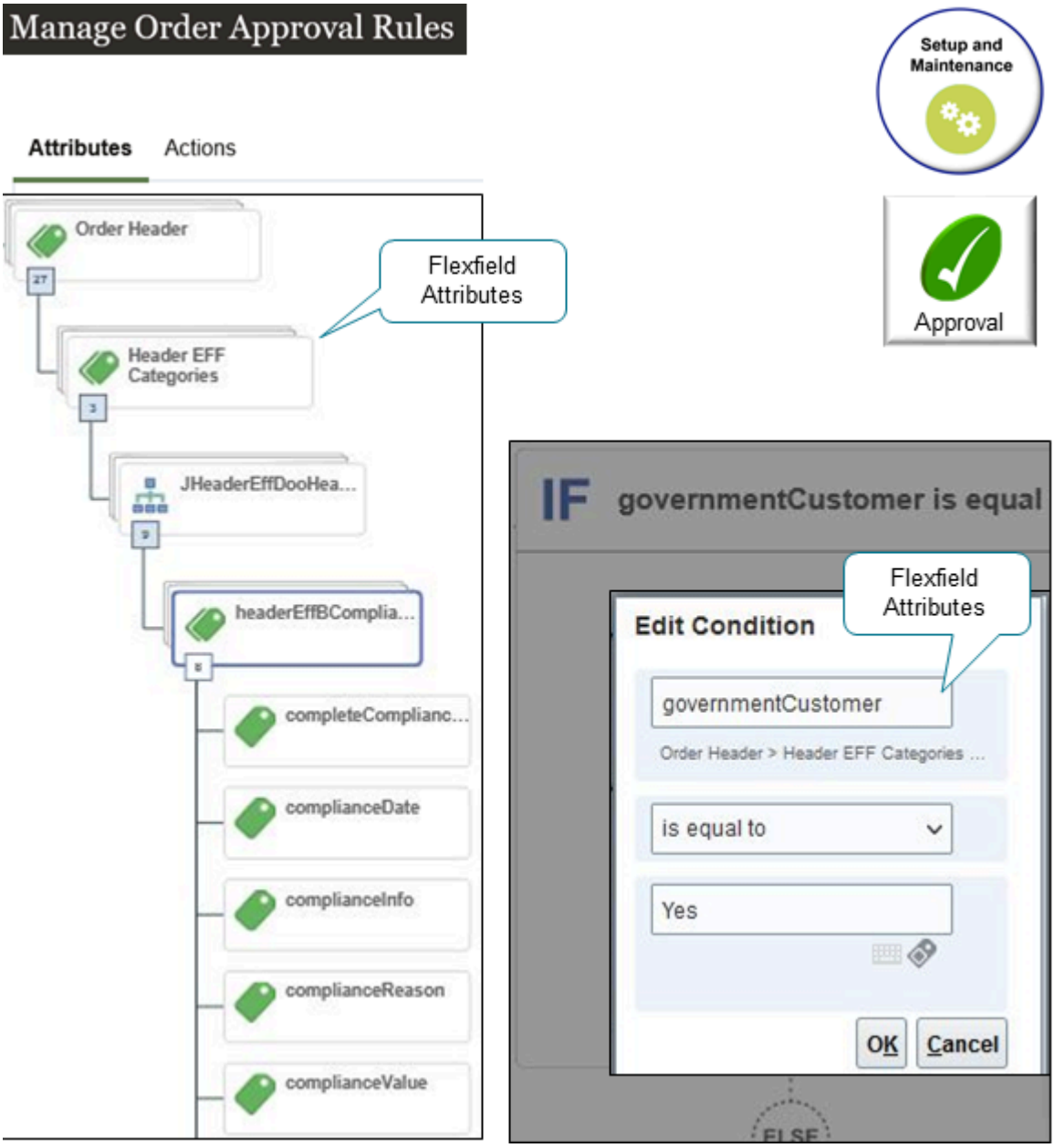
   The BPM Worklist area displays the warning only if you have created or modified a flexfield. You must synchronize your flexfield so the rules editor has the latest flexfield definitions.

7. Create a new approval rule or modify an existing one.

   Assume you're adding a flexfield on the order header.

   - Go to the Setup and Maintenance work area, then open the Manage Order Approval Rules task.
   - On the Manage Order Approval Rules page, click **Create New Rule**.
   - In the Attributes area, click **Order Header > Header EFF Categories**.

     Notice that the Header EFF Categories tree contains the categories that you created in step 1.



   - Click **New Condition**.
   - In the Create Condition dialog, enter `Header`.

- Click **HeaderId (Order Header > Header EFF Categories)**.
- Notice how the dialog contains the categories that you created in step 1.

8. Test your set up.
   - Go to the Order Management work area and create a sales order.
   - Verify that the order header displays the Government Customer attribute.
   - Add a check mark to the Government Customer attribute, add an order line, then click **Submit**.
   - Verify that Order Management starts the approval process.

Credit

Credit Check

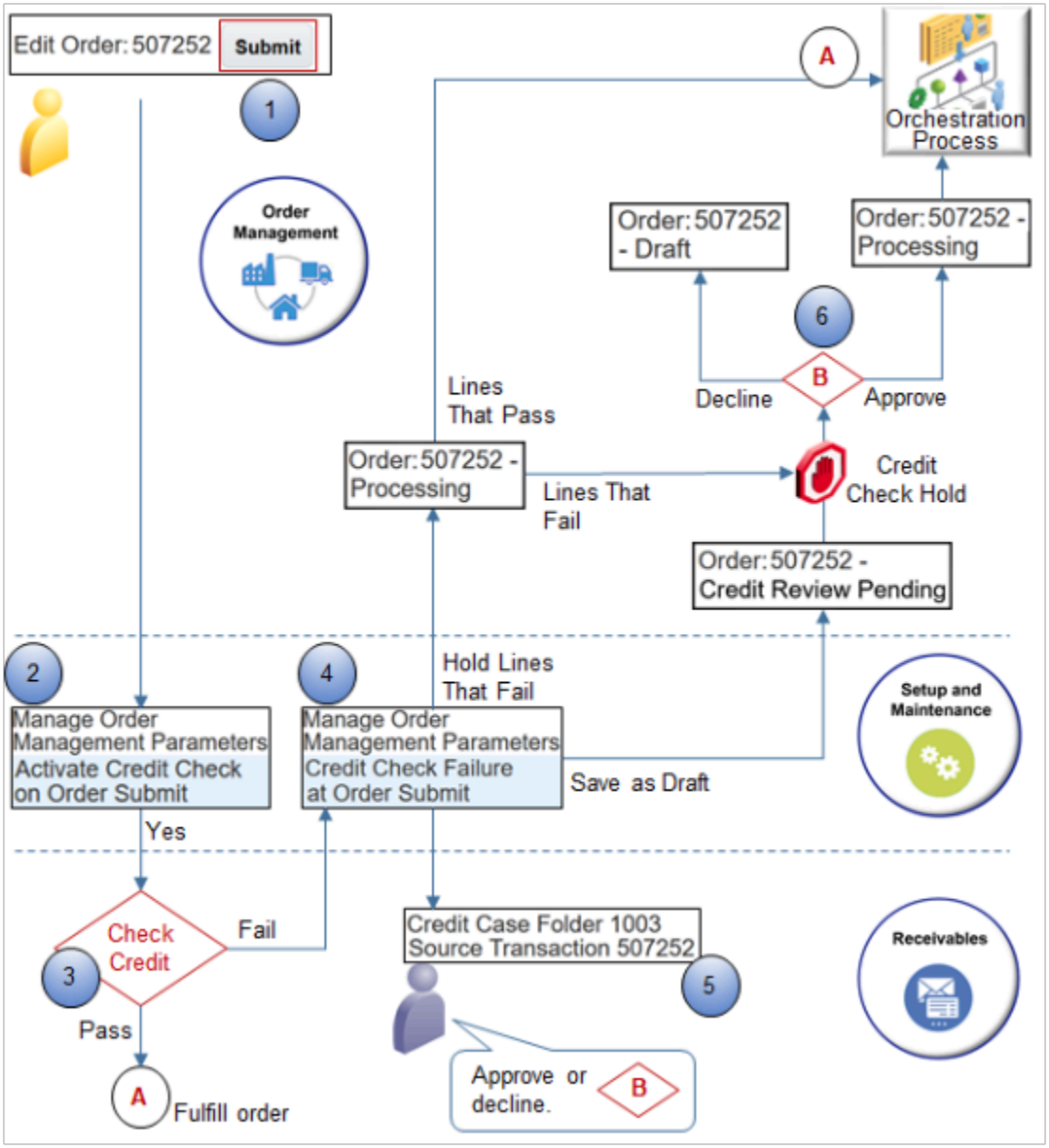
Overview of Setting Up Credit Check

Use credit check in Order Management to minimize the financial risk that your organization assumes during day-to-day operations. Validate each sales order and verify your customer has credit available that's sufficient to cover the cost of purchase.

For example:

- Screen each sales order for customer credit at order submit.
- Do credit check according to order type.
- Automatically release credit hold after the credit check issue resolves and proceed to fulfillment.
- Check credit before shipment during a long fulfillment cycle. Credit status might change during fulfillment.
- Allow only some users to manually release a credit check hold.

Order Management can do a credit check on each sales order or on each order line, and can hold an order line that doesn't pass credit check.



Note

1. The Order Entry Specialist creates a sales order in Order Management, adds order lines, sets payment terms, payment method, and Bill-to Customer, then clicks Submit.
2. If you set the Activate Credit Check on Order Submit parameter to:
    - **Yes.**. Order Management calls the credit check service in Receivables.
    - **No.** Order Management orchestrates fulfillment.

    For details, see *Manage Order Management Parameters*.

3. Do credit check.
    - Receivables does credit check. If credit check.
        - **Passes.** Receivables sends the result to Order Management, and Order Management orchestrates fulfillment.
        - **Fails.** Credit check continues.
4. Order Management processes the sales order depending on how you set the value of the Credit Check Failure at Order Submit parameter.

Value	Description
Save Order in Draft Status	Save the sales order in Draft status and don't proceed to fulfillment. Wait for the credit analyst to approve or decline.
Submit the Order with Hold on Lines That Failed Credit Check	Save the sales order in Processing status, place a hold on each order line that fails credit check, then send order lines that aren't on hold to fulfillment.

   For details, see the Handle Credit Check Failure section later in this topic.
5. Receivables creates a case folder for the credit request and determines credit status.

Status	Description
Open	Receivables authorizes credit and sets Authorization Status to Open.
Failed	Receivables sends the credit check result to Order Management, including credit authorization number, credit authorization amount, and credit expiration date.

   The credit analyst uses the Credit Reviews page in the Receivables work area to examine the case folder, then approves or declines the credit request. Receivables sends the result of the credit review to Order Management.
6. If the credit analyst.
    - **Approves**. Receivables sends the authorization ID, date, Bill To, and release reason to Order Management. Order Management sets the status to Processing and orchestrates fulfillment for the entire sales order.
    - **Declines**. Order Management sets the status to Draft and doesn't send any part of the sales order to fulfillment. The Order Entry Specialist must revise the sales order.

Note

- Credit check establishes a credit line only for a single currency, and only for the customer account.
- Order Management runs credit check when the user submits the sales order, and immediately before running approval. If credit check succeeds, and if approval fails, then Order Management removes authorization for the authorization number, then gets a new authorization number after the user resubmits the sales order.

- If the Order Entry Specialist modifies an attribute that affects payment sometime after Order Management runs credit check, then Order Management runs credit check again. Bill-To Customer is an example of an attribute that affects payment.
- If the Order Entry Specialist cancels a sales order before fulfillment finishes, then Order Management communicates with Credit Management in Receivables to reverse the credit amount of the sales order. It reverses this amount in the customer account in Oracle Financials.

Calculate Available Credit

Here's the equation that Receivables uses when it calculates available credit.

- Credit limit, minus the open balance, minus total outstanding authorization, equals available credit.

Receivables also does this.

- Uses invoice details to determine the open balance
- Sums the amounts of an authorization to determine the total outstanding authorization

For example, assume a customer account contains these values.

- Credit limit is $2,000
- Open balance is $0
- Outstanding authorization is $500
- Customer orders an item that costs $1,000

Receivables does these calculations.

- $2,000 credit limit, minus $0 open balance, minus $500 outstanding authorization, equals $1,500 available credit.
- $1,500 available credit exceeds the $1,000 item cost, so Receivables authorizes the purchase, and then sends an authorization number, expiration date, authorization amount of $1,000, and available credit to Order Management. Receivables uses the same currency that the sales order uses.

You can use the Receivables work area to define a different formula, such as modifying credit limit, including shipping charges and taxes, and so on. For details, see *Using Receivables Credit to Cash*.

Handle Credit Check Failure

Order Management processes credit check differently depending on the value you set for the Credit Check Failure at Order Submit parameter.

Save Order in Draft Status

This section assumes you enable the Credit Management feature in Receivables.

If any order line fails credit check, then the entire sales order fails during order submit.

Order Management processes the sales order.

1. Saves the sales order in status Credit Review Pending.
2. Doesn't submit the sales order to order fulfillment.
3. Doesn't apply a credit check hold on any order line.
4. Receivables opens a case folder for the sales order in Credit Reviews.
5. The credit analyst sets the recommendation and closes the case folder in Credit Management.

6. Order Management does these steps.
    - Updates the Credit Authorization Number attribute and the Expiration Date attribute on the sales order.
    - Displays a message for the fulfillment lines. The message contains case folder details, such as Closure Date, Status, and Closed By. For details, see *Release Holds for Order Lines That Fail Credit Check*.
    - Changes order status and continues processing depending on how the credit analyst sets the recommendation.

Credit Analyst's Recommendation	Result in Order Management
Approve Source Transaction Credit Request	Depends on whether you implement trade compliance check or order approval. - **Don't implement.** Changes order status to Processing, automatically removes credit hold, and proceeds to order fulfillment. - **Do implement.** Sets status according to the state of compliance check or order approval, then proceeds to compliance check or order approval.
Decline Source Transaction Credit Request	Change order status to Draft. The Order Entry Specialist must revise and resubmit the sales order.

Submit Order with Hold on Lines that Failed Credit Check

This section assumes you enable the Credit Management feature in the Receivables work area.

Order Management does these steps.

1. Submits the sales order to order fulfillment even if credit check fails.
2. Applies a credit check hold on each order line that fails credit check.
3. Holds the lines that fail credit check on the first step of the orchestration process.
4. Opens a case folder for the sales order in Credit Management.
5. Sends fulfillment lines that pass credit check to the next step in the orchestration process.
6. The credit analyst sets the recommendation and closes the case folder in Credit Management.
7. Order Management does these steps.
    - Updates attributes and displays a message. This behavior is similar to the behavior that occurs when you use Save Order in Draft Status.
    - Changes order status depending on how the credit analyst sets the recommendation when closing the case folder.

Recommendation	Result in Order Management
Approve Source Transaction Credit Request	Release credit check hold on fulfillment lines and send them to order fulfillment.
Decline Source Transaction Credit Request	Doesn't release credit check hold on fulfillment lines. These lines remain on hold, and Order Management doesn't send them to order fulfillment. The Order Entry Specialist can query or cancel failed lines, but can't revise them.

Recommendation	Result in Order Management

Revising Sales Orders

Note

- If a revision fails credit check, then Order Management sets the status to Credit Review Pending regardless of how you set the parameter.
- If the sales order or order revision is in the Credit Review Pending status, then the Order Entry Specialist can't make any changes. The Order Entry Specialist must use the Revert to Draft action, and then make changes.
- If you enable or don't enable the Credit Management feature.
    - **Enable.** Order Management waits for the credit analyst to close the case folder before submitting failed lines to order fulfillment.
    - **Don't enable.** The Order Entry Specialist must manually revise the revision to a draft, modify the sales order, such as delete lines that failed credit check, reduce quantity so the sales order total doesn't exceed credit limit, and so on, and then resubmit.
- The revision must successfully pass credit check so Order Management can merge the Draft sales order into the Processing instance of the same sales order that's currently running in order fulfillment.

Behavior When You Don't Enable Credit Management

This section assumes you don't enable the Credit Management feature in Receivables, and credit check fails during order submit.

1. Order Management takes action depending on the value of the Credit Check Failure at Order Submit parameter.

Value	Result
Save Order in Draft Status	Set order status to Credit Review Pending. Wait for the credit analyst to resolve the issue and close the case folder. Don't submit any order lines to order fulfillment.
Submit Order with Hold on Lines that Failed Credit Check	Set order status to Processing. Place each order line that fails credit check on credit check hold. Submit all order lines, including order lines that fail credit check, to order fulfillment.

2. Credit analyst resolves the issue and closes the case folder.
3. Order Entry Specialist must revise the sales order, then resubmit it.

Related Topics

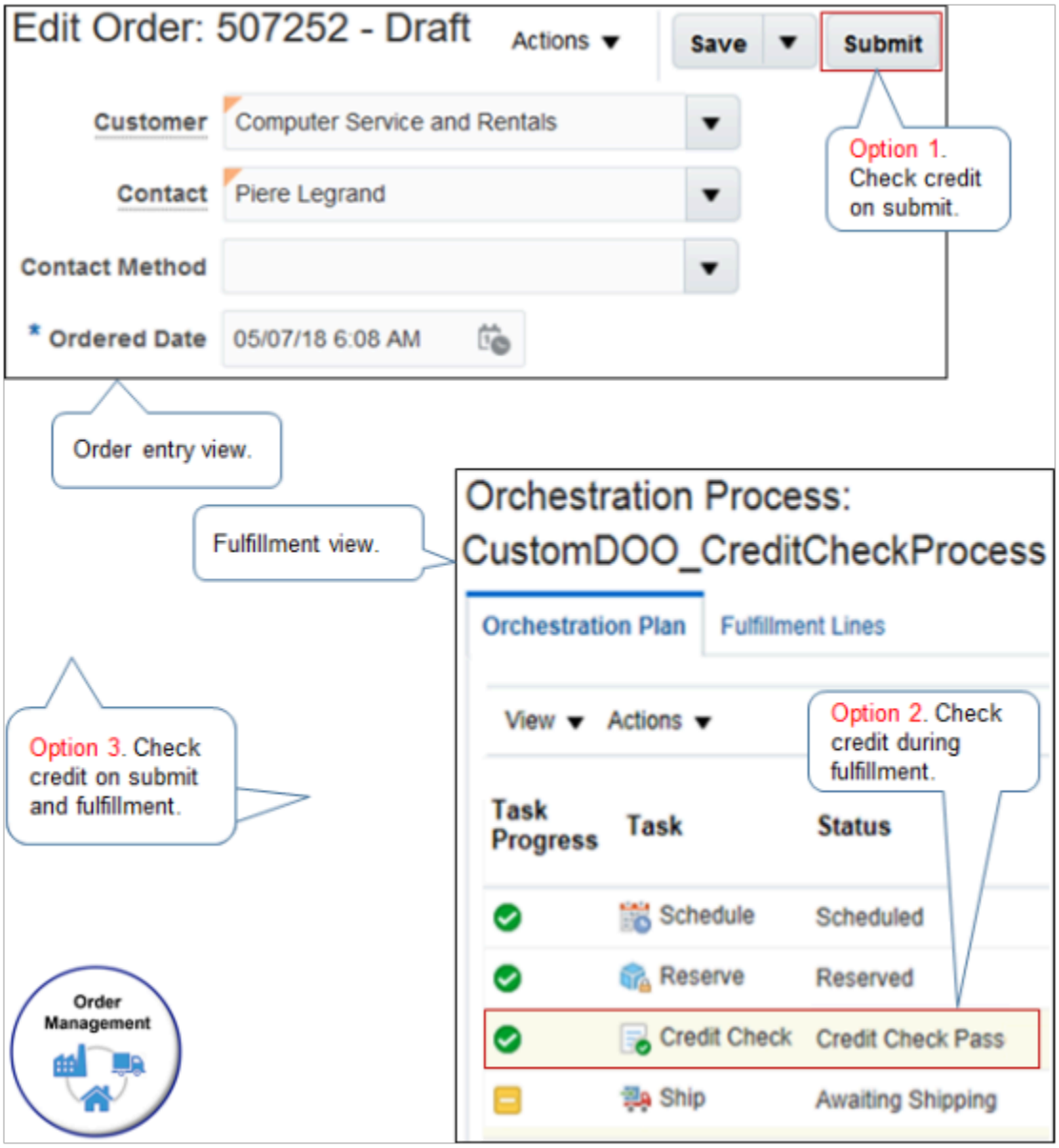
- Import Source Orders That Include Credit Check
- Manage Order Management Parameters
- Set Up Credit Check
- Release Holds for Order Lines That Fail Credit Check
- Reauthorize Payment

Guidelines for Setting Up Credit Check

Use these guidelines to help you set up credit check in Order Management.

Determine When to Check Credit

Credit check during order entry or order fulfillment.



Note

- **Option 1.** Check credit when the Order Entry Specialist clicks Submit when creating or revising a sales order.
    - Submit the sales order to order fulfillment but hold order lines that fail credit check.
    - Don't submit the sales order to order fulfillment. Instead, set the status to Credit Review Pending, and route the sales order to a credit analyst to do credit check.
- **Option 2.** Check credit during order fulfillment. For example, add a credit check step to the orchestration process after scheduling and reserving the item, but before shipping it.
- **Option 3.** Check credit during order entry at order submit, then check credit again during order fulfillment.

Order Management processes credit check differently during order submit or order fulfillment.

Question	Check Credit On Submit	Check Credit During Order Fulfillment
How does Order Management send order lines to Credit Management?	Send order lines as a group according to bill-to customer.	Send one fulfillment line at a time.
What happens when an order line fails credit check?	Order Management does one of these. • Place order line on credit check hold. • Lock order line, set status to Credit Review Pending, and wait for credit analyst to review case folder.	Set fulfillment line to error.
How can an order line that fails credit check proceed?	Order Entry Specialist does one of these. • Manually releases hold. • Revises the sales order to status Draft, modifies it, and submits it again.	Do one of these. • Order Entry Specialist revises the sales order. • Credit analyst adjusts credit limit, then Order Management runs a scheduled process that you set up that retries the orchestration process step. The Order Entry Specialist can also manually retry.
Limitations	Credit check happens only for all order lines in the entire sales order. You can't check credit for a single line or only for some lines. Order Entry Specialist can't modify attributes that affect credit while the fulfillment line is on credit check hold. Order Entry Specialist can only cancel the line.	Order Entry Specialist can't manually proceed past the credit check. Can only retry the step.

Order Management processes credit check differently depending on the state of the order lines.

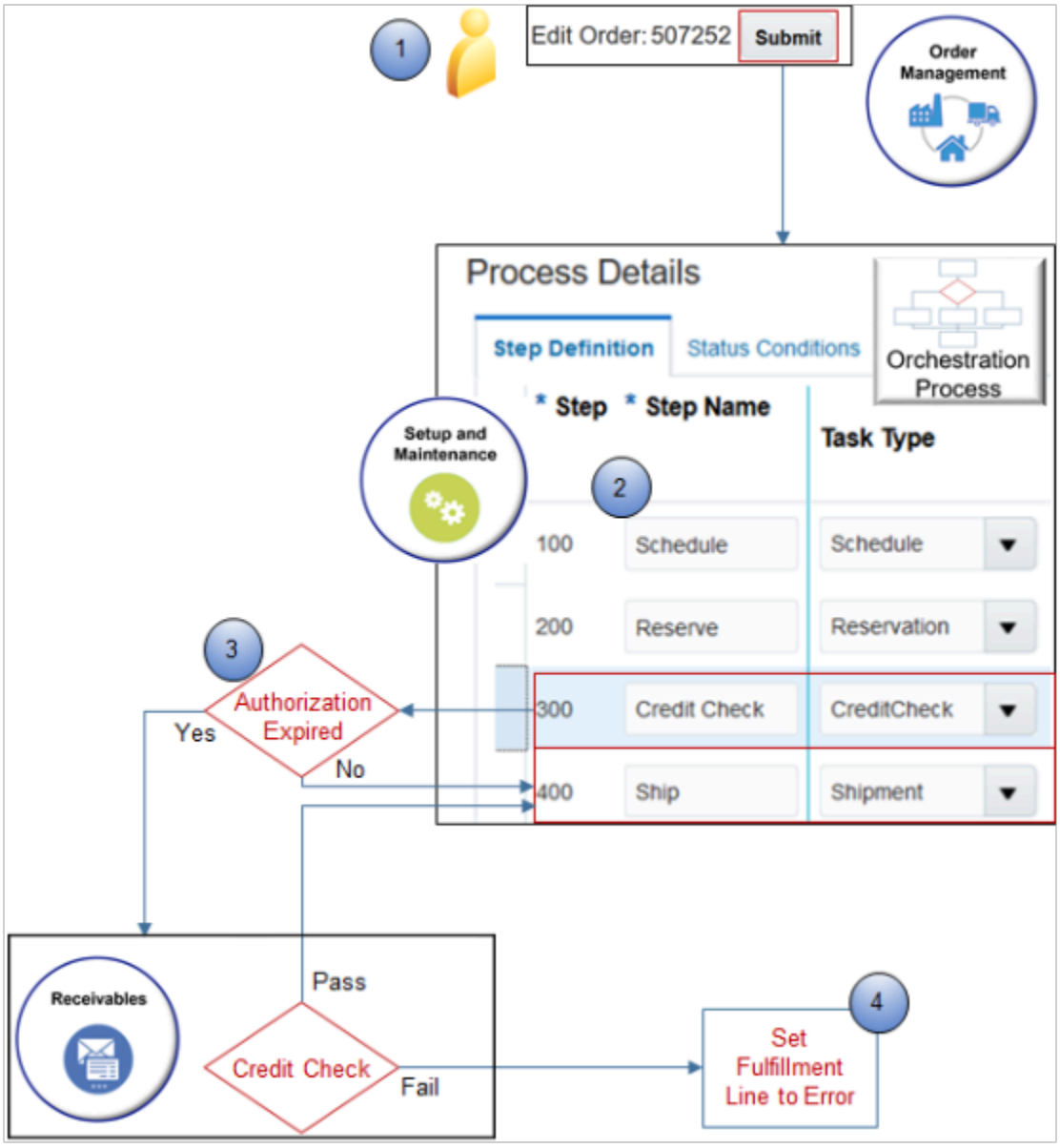
State of Order Lines	Check Credit On Submit	Check Credit During Order Fulfillment
In draft.	Allow all changes. Make changes, then recheck credit on all lines.	Fulfillment hasn't started, so allow all changes.
Lines are in order fulfillment, and. • Before credit check step. • In error because credit check failed during fulfillment. • Credit check finished.	Order Management rechecks each line that it checked earlier during submit. It does the recheck regardless of the change. It creates a new authorization.	Allow all changes. Apply normal change processing. If Order management finished credit check during order fulfillment, and if credit check fails when processing the change, then Order Management doesn't allow the Order Entry Specialist to revise the sales order. You must use error recovery to fix the problem.
In order fulfillment, on credit hold.	If changed attributes. • **Affect credit.** Reject revision. • **Don't affect credit.** Recheck credit.	Allow all changes. Apply normal change processing.

Check Credit During Order Fulfillment

Here's when you can check credit during order fulfillment.

- Your fulfillment cycles run for a long time. For example, run credit check immediately before shipping to make sure credit authorization hasn't expired while process the sales order through fulfillment.
- You import sales orders from a source system, the source system already checked credit, so you don't want to check credit again when submitting the order to fulfillment. But you also don't want to send the order to shipping if credit authorization expired. A long delay might exist between the time the source system checks credit and when the item is ready to ship.

Here's how it works.



Note

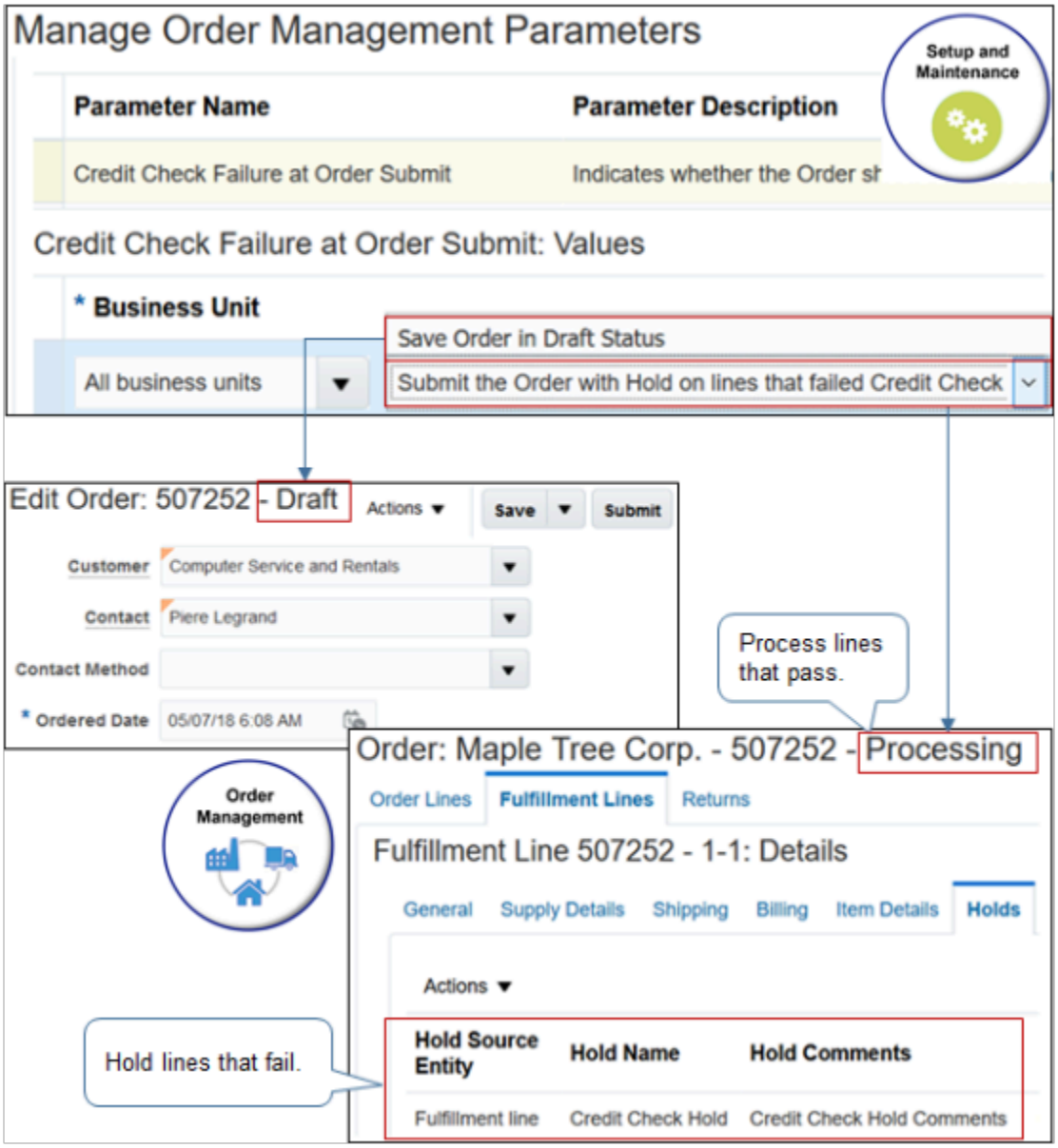
1. Order Entry Specialist clicks Submit.
2. The orchestration process starts fulfillment, such as schedule inventory, reserve inventory, then reaches credit check step.
3. If the expiration date of the credit authorization is expired, or if Order Management hasn't already done credit check at least one time, then the orchestration process sends a request to Receivables to do credit check on fulfillment lines.
    - If credit check passes the first time Receivables performs credit check, then it creates an authorization number and authorization expiration date.

      For example, assume you do credit check during order submit on January 1, 2019, credit check passes, and Receivables sets the expiration date to January 10, 2019. If the orchestration process runs the credit check step on.

        - January 2 during fulfillment, then it won't run credit check again, but will instead proceed to the next orchestration process step.
        - January 11, then it will run credit check again.
    - The orchestration process sends only the fulfillment lines that require credit check. It doesn't send the entire sales order.
4. If credit check fails, then the fulfillment line goes into error and processing stops.
    - Credit analyst must adjust credit in Credit Management in Receivables.
    - Orchestration process will run credit check step again after credit analyst adjusts credit.

Set the Order Management Parameter

Set the value for the Credit Check Failure at Order Submit parameter.



Note

Value	Description
Save Order in Draft Status	Save the sales order in the Draft status. Don't send any lines to order fulfillment. Use when. • Your sales order includes more than one value for Bill-to Customer on the order lines, and you must move the order to fulfillment only if credit check passes for all bill-to customers. • You don't need to do credit check during fulfillment. • You don't need to release credit check hold so fulfillment can continue. You can wait for the credit analyst to resolve the hold.

Value	Description
Submit Order with Hold on Lines that Failed Credit Check	Save the sales order in Processing status, place a hold on each order line that fails credit check, then send order lines that aren't on hold to fulfillment. Use when. • Your sales orders sometimes include more than one value for Bill-to Customer on the order lines, and you must move the order to fulfillment even if credit check fails for some or all bill-to customers. For example, if credit check for Bill To Customer x succeeds, and if credit check for Bill To Customer y fails, then Order Management can send order lines for customer x to order fulfillment, and place order lines for customer y on hold. If order lines include only one Bill To Customer value, then Order Management uses the order total when it checks credit. • You don't revise attributes that affect credit check, or you might cancel the failed lines. • You must release credit check hold so fulfillment can continue, even if you must manually release the hold.

Frequent Implementation Questions

Question	Description
Why does Order Management place a credit check hold on the fulfillment line instead of on the order header when all fulfillment lines of the sales order fail credit check?	The orchestration process orchestrates and Order Management tracks each fulfillment line separately so they can optimize fulfillment. For example, if supply is available for lines x and y but not z, then it can reserve supply and move to the next orchestration process step for x and y. If Order Management placed the hold on the sales order instead of the fulfillment lines, then it would have to pause orchestration for all lines while it waits for supply to become available for z.
How can we allow fulfillment lines that are on credit check hold to proceed beyond scheduling?	Manually release the credit check hold. To make sure you don't ship a line that fails credit check, add another orchestration process step that does credit check. Add it after the scheduling step.
Assume credit check fails during fulfillment and the credit check step is in error. Can we manually get that line past the credit check step?	You must retry the step and it must pass credit check so the fulfillment line can move forward in the flow. You can't manually remove a hold because Order Management doesn't place a hold when credit check fails during fulfillment. It only places the line in error.
Can I set up my orchestration process so we don't do a credit check according to the type of sales order? For example, I create a task type named Skip Credit Check, and then add a condition to my orchestration process: **`If task type equals Skip Credit Check, the release pause tasks that are on only because of credit check.`**	You can't set up an orchestration process that skips credit check according to the type of sales order.

Related Topics

- Import Source Orders That Include Credit Check
- Manage Order Management Parameters
- Set Up Credit Check
- Release Holds for Order Lines That Fail Credit Check
- Reauthorize Payment

Set Up Credit Check

Do the setup tasks, such as making sure the customer record meets requirements, verifying customer account setup, and so on.

1. Make sure the customer record exists in Oracle Financials, and make sure the billing account for the customer meets these requirements.
    - Is active.
    - Has an established credit line.
    - Make sure the Include in Credit Check option for the account contains a check mark.
    - Set these attributes on the account.
        - Credit Limit
        - Credit Currency
        - Order Amount Limit
2. Verify the customer account setup.
    - Make sure you have the privileges that you need to administer Order Management.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Financials
        - Functional Area: Customers
        - Task: Create Customer
    - On the Create Organization Customer page, in the Organization Information area, search for the customer account you must verify.
    - Click **Profile History**, then verify the attributes that affect credit, such as Credit Line and Credit Currency.

      To get details about how to do these set ups, see Oracle Credit Management User Guide.
        - Automatically do credit check in Order Management. You must enable the Credit Management in Oracle Receivables feature.
        - Use Credit Management in Oracle Financials to enable the customer for credit check, set up the customer credit profile, and so on.
3. Optional. Enable the Credit Management in Receivables feature.

   Enable the AR_CREDIT_MGMT lookup code. Use the Manage Standard Lookups page in the Setup and Maintenance work area.

4. Set up these parameters.
    - Activate Credit Check on Order Submit
    - Credit Check Failure at Order Submit

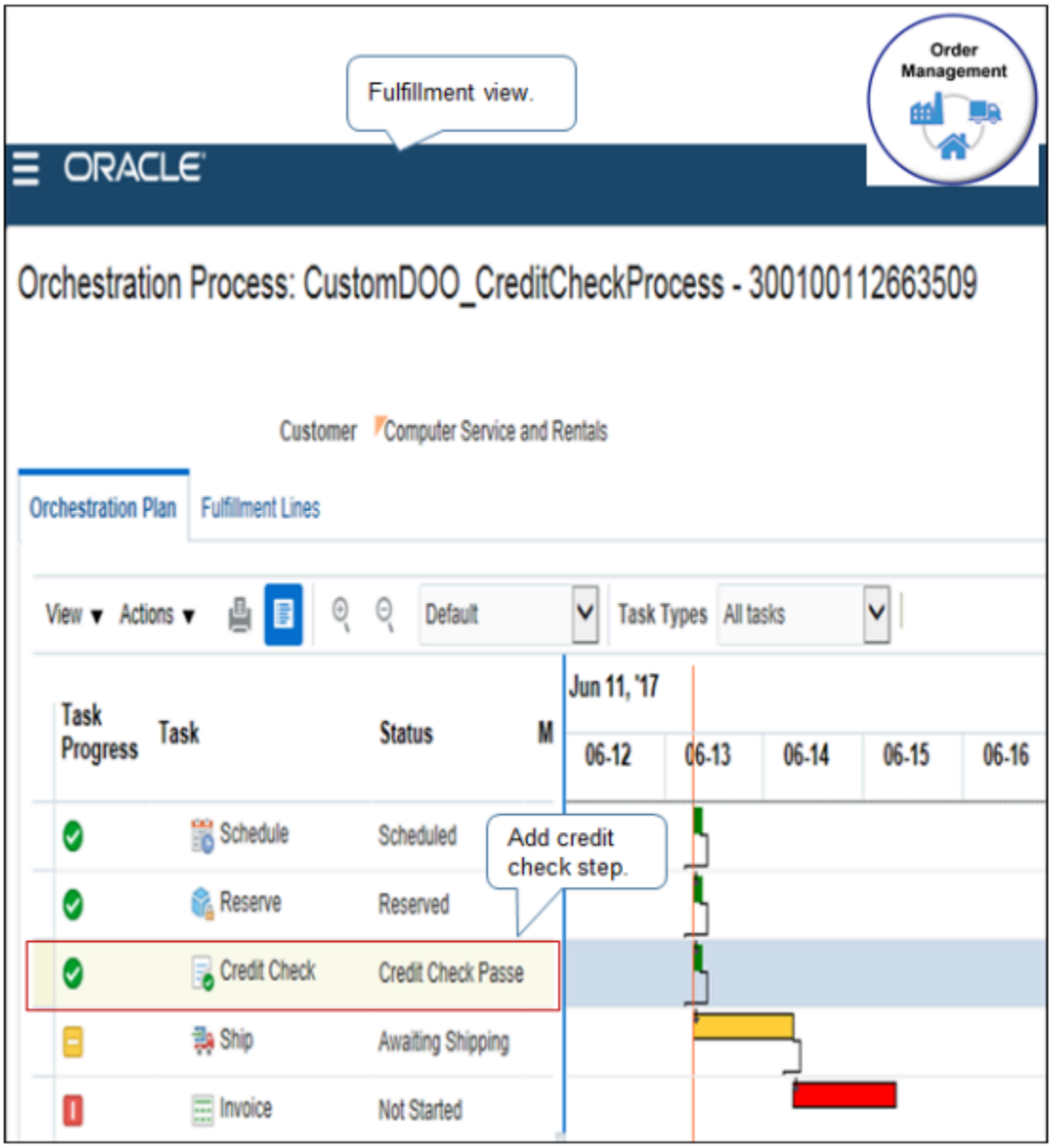
    For details, see *Manage Order Management Parameters*.
5. Optional. Set up credit check to occur during order fulfillment.

Set Up Credit Check to Occur During Order Fulfillment

In some situations, order fulfillment might require days to complete because of a variety of factors. For example:

- Inventory is out of stock for the item and requires several days to replenish.
- The credit that's available for a customer might decrease during this time because the customer continues to make other purchases.

You can add a credit check step at any point in the orchestration process. For example, to occur immediately before shipping. Use this functionality to make sure the credit check is accurate and up to date so you don't ship an item when customer credit no longer covers the purchase amount.



For example, you can add a credit check step to an orchestration process immediately after the Reserve step, and immediately before the Ship step.

Here are the values you set for the credit check step you add.

Attribute	Value
Step Name	Enter text that describes the step, such as Credit Check.
Task Type	DOO_CreditCheck
Task	DOO_CreditCheck

Attribute	Value
Service	Create Credit Check Request

If credit check fails when you run it from the DOO_CreditCheck task, then credit check sets the task to an error state, stops processing order lines, and pauses the orchestration process. To recover, you must set up the Recover Errors scheduled process. Attempting to release a hold won't resume processing.

If you add a credit check step to an orchestration process, and if the orchestration process already checked credit for the sales order before order submit, then Order Management examines the expiration date of the credit check.

- **If the current date occurs after the expiration date.** Order Management does the credit check.
- **If the current date occurs on or before the expiration date.** Order Management doesn't do the credit check.

For details, see *Set Up Orchestration Processes*.

Related Topics

- Manage Order Management Parameters
- Import Source Orders That Include Credit Check
- Overview of Orchestration Processes
- Manage Credit Check

Allow Users to Close Case Folders

Set up Order Management to allow your users to review customer credit and approve the case folder.

In most organizations, the person who books sales orders and views them go on credit check hold doesn't sign in with the privileges that allow them to review customer credit or approve the case folder. However, some users, such as an order manager, might need to review and approve.

For details about case folders and the set ups you must do to use them, see *Implementing Receivables Credit to Cash* and *Using Receivables Credit to Cash*.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

In this example, you will allow user Aaron Holmes to close case folders.

1. Assign a role to your user that allows the user to view and approve credit.

   For details, see Oracle Credit Management User Guide.
2. Make sure you have the privileges that you need to administer users and security.
3. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Users and Security
   - Task: Manage Job Roles

     The Security Console displays the Roles page after you click Manage Job Roles. For details, see *Securing SCM*.

4. Add your user to the Credit Manager job role.
    - On the Roles page, search for, then open the Credit Manager job role.
    - Click **Actions > Copy Role**.
    - In the Copy Options dialog, click **Copy Top Role and Inherited Roles > Copy Role**.
    - On the task-based page that displays, click **Next** four times until Users is the active step on the task line.
    - On the Copy Role Credit Manager Custom Users page, click **Add User**.
    - In the Add User dialog, in the search window, enter the predefined user `ar_mgr_operations`, then click **Search**.

      If you don't use a predefined user, but instead create your own user, then make sure the user includes a role that allows the user to close case folders.
    - In the search results, click the **row** that contains Aaron Holmes, the user you must add, then click **Add User to Role > Cancel**.
    - Confirm that the Copy Role Credit Manager Custom Users page displays the user you just added, then click **Next > Submit and Close**.
5. Import the user and role.
    - Go to the Scheduled Processes work area.
    - On the Scheduled Processes page, click **Schedule New Process**.
    - In the Schedule New Process dialog, click the **down arrow**, search for Import User and Role Application Security Data, then click **OK > OK**. For details, see *Configure the Security Console*.
    - In the Process Details dialog, click **Submit**.
    - On the Scheduled Processes page, click **Actions > Refresh**.

      Repeat until this value displays for your scheduled process.

Attribute	Value
Status	Succeeded

6. Sign out of Order Management.

Related Topics

- Manage Order Management Parameters
- Import Source Orders That Include Credit Check
- Overview of Orchestration Processes
- Manage Credit Check

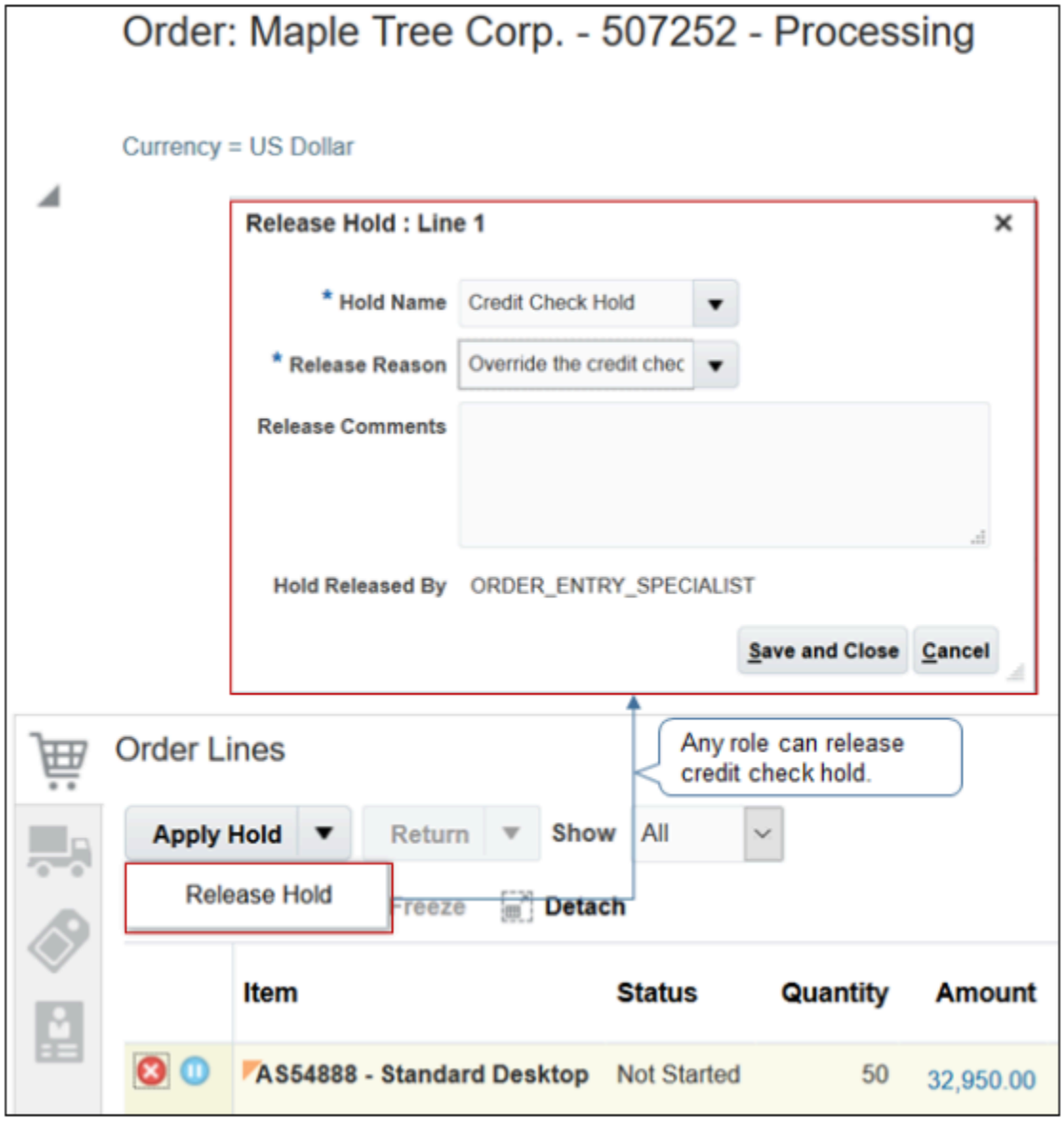
Control Who Can Release Credit Check Hold

Set up a hold code to control who can release a credit check hold in the Order Management work area.

Order Management comes predefined to allow any user to release a credit check hold in the Order Management work area. In this example, assume you must allow only the Order Manager to release credit check hold.

1. Examine the predefined behavior.

- Make sure you have the privileges that you need to create sales orders.
- Open a sales order that's on hold.
- In the Order Lines area, click **Apply Hold > Release Hold**.
- Notice that you can use the Release Hold dialog to set the values and release the hold.



2. Specify the role that can release credit check hold.
    - Make sure you have the privileges that you need to administer Order Management.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Hold Codes
    - On the Manage Hold Definitions page, click **Query by Example**, enter the value, then press `Enter` on your keyboard.

Attribute	Value
Name	Credit Check Hold

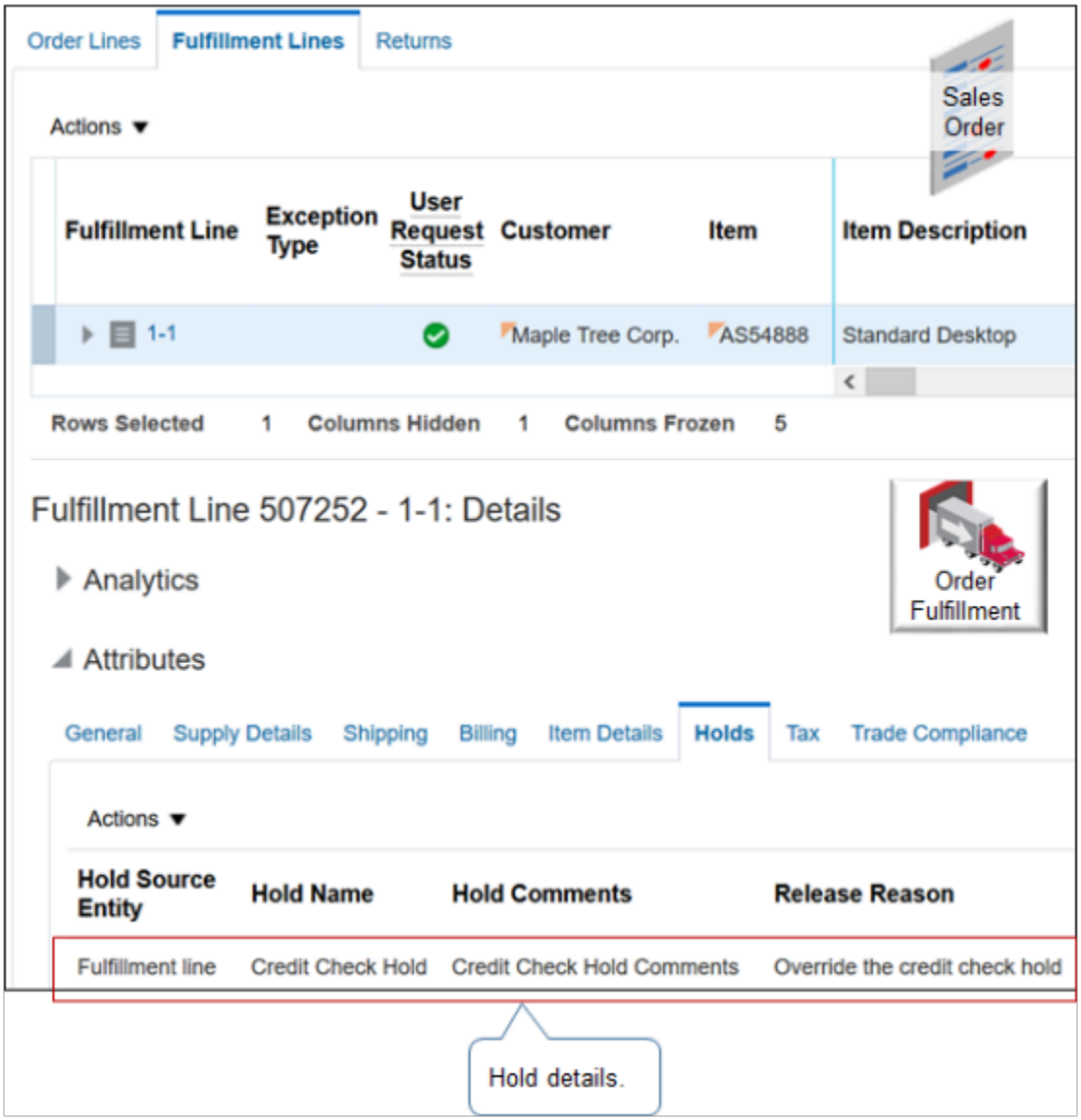
    - In the Details area, click **Applicable Roles > Selected Roles**.
    - Click **Actions > Select and Add**.
    - In the Select and Add dialog, search for this value.

Attribute	Value
Role	Order Manager

    - In the search results, click the **row** that includes Order Manager, then click **Apply > OK**.
    - In the Details area, make sure the Release Hold attribute contains a check mark, then click **Save and Close**.

3. Test your set up.
    - Make sure you have the privileges that you need to create sales orders.
    - Open the sales order, then click **Apply Hold > Release Hold**.
    - In the Release Hold dialog, verify that you can't set the Hold Name attribute, and can't release the hold.
    - Sign out, then sign in with the privileges that you need to manage sales orders.
    - Open the sales order, verify you can set the Hold Name attribute in the Release Hold dialog, then click **Save and Close**.
    - On the Order page, wait a moment, then click Refresh.

      Notice that Order Management changes the Status attribute on the order line from Not Started to Scheduled. If the status doesn't change, wait a moment, then try again.
    - Click **Actions > Switch to Fulfillment View**.
    - Click **Fulfillment Lines**.
    - In the Details area, click **Holds**, then examine the release details.



Related Topics

- Manage Order Management Parameters
- Import Source Orders That Include Credit Check
- Overview of Orchestration Processes
- Manage Credit Check

Import Source Orders That Include Credit Check

Control how Order Management handles credit check for each source order that you import from a source system.

Use a web service or file-based data import. This example does credit check in the source system.

Set attributes in the payload you use to import your source order.

Attribute	Description
PreCreditCheckedFlag	Set the value. • **True.** Don't check credit. Order Management won't do credit check on the source order even if you enable credit check for the business unit. • **False.** Do check credit.
CreditCheckAuthorizationCode	Set to any numeric value, such as 9090. Order Management uses the value that you set to populate the Credit Reference attribute on the order line.

Attribute	Description
	If you import a source order, and if a sales order already exists in Order Management that matches the source order, then Order Management examines the value of Credit Reference in the sales order. If Credit Reference isn't equal to the value of CreditCheckAuthorizationCode, then Order Management checks credit for the source order.
CreditCheckAuthorizationExpiryDate	Set to a date value, such as 2016-10-05T10:00:00Z. Order Management uses the value that you set to populate the Expiration Date attribute on the order line. If the current date. • **Occurs after CreditCheckAuthorizationExpiryDate.** Order Management checks credit. • **Occurs on or before CreditCheckAuthorizationExpiryDate.** Order Management doesn't check credit.

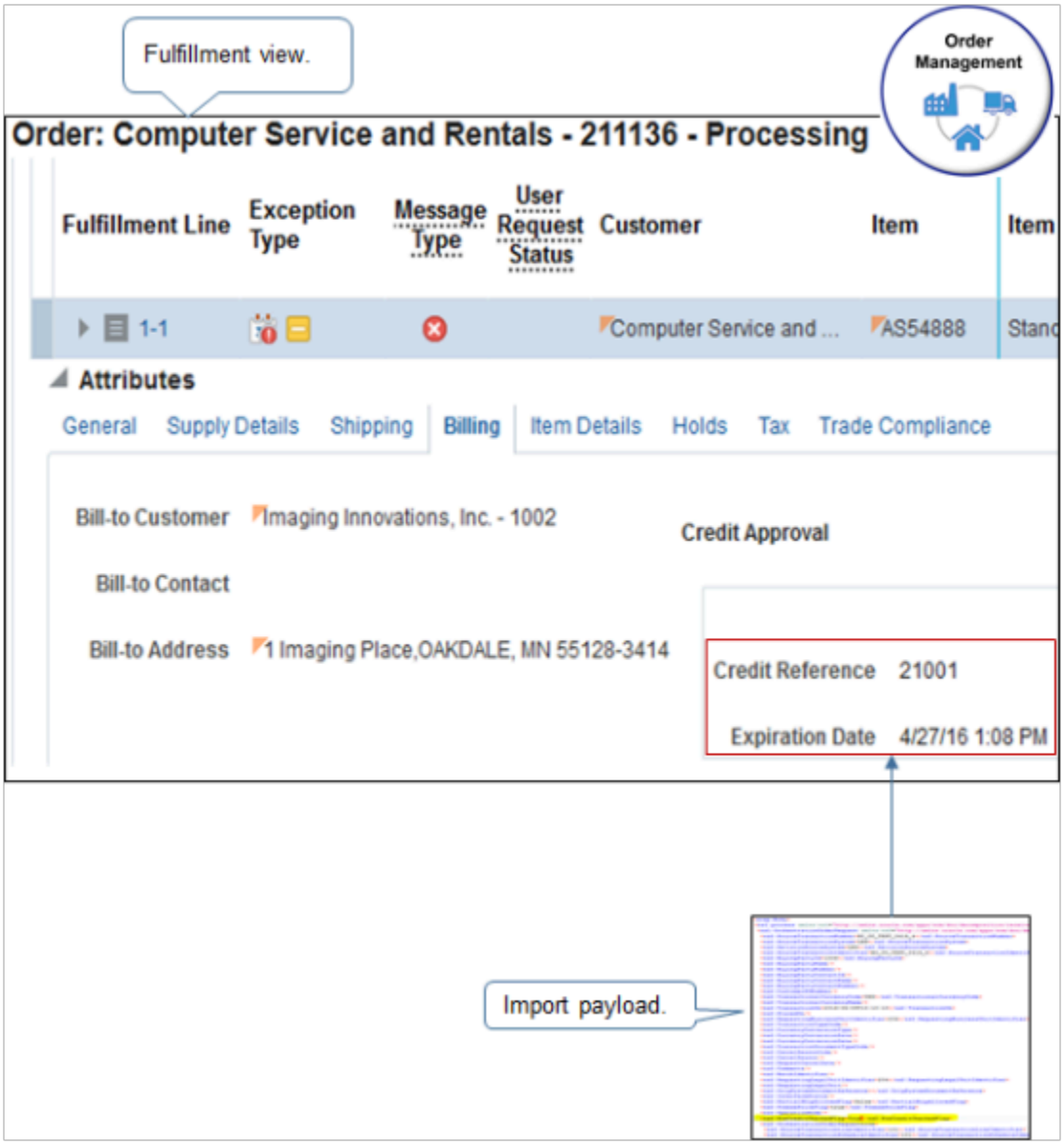
Note

- You can reference PreCreditCheckedFlag from a transformation rule or an order management extension.
- You can send PreCreditCheckedFlag only through order import.
- PreCreditCheckedFlag doesn't display in the Order Management work area.

To view the code and date, in the Order Management work area, navigate to a fulfillment view, then open a sales order. On the Order page, in the Fulfillment Line Details area, in the Attributes area, on the Billing tab, in the Credit Approval area, note.

- CreditCheckAuthorizationCode displays in the Credit Reference attribute.
- CreditCheckAuthorizationExpiryDate displays in the Expiration Date attribute.

For example:



Here's an example payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Header><your security details></
soap:Header>
 <soap:Body>
 <ns1:process xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/DooDecompReceiveOrderComposite">
 <ns1:OrchestrationOrderRequest xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/model/">
 <ns2:SourceTransactionNumber>RD_CC_TEST_0907_1</ns2:SourceTransactionNumber>
 <ns2:SourceTransactionSystem>LEG</ns2:SourceTransactionSystem>
 <ns2:RevisionSourceSystem>LEG</ns2:RevisionSourceSystem>
 <ns2:SourceTransactionIdentifier>RD_CC_TEST_0907_1</ns2:SourceTransactionIdentifier>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:BuyingPartyName></ns2:BuyingPartyName>
 <ns2:BuyingPartyNumber></ns2:BuyingPartyNumber>
 <ns2:BuyingPartyContactId></ns2:BuyingPartyContactId>
 <ns2:BuyingPartyContactName></ns2:BuyingPartyContactName>
 <ns2:BuyingPartyContactNumber></ns2:BuyingPartyContactNumber>
```

```
<ns2:CustomerPONumber></ns2:CustomerPONumber>
<ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
<ns2:TransactionalCurrencyName></ns2:TransactionalCurrencyName>
<ns2:TransactionOn>2017-06-07T10:10:10</ns2:TransactionOn>
<ns2:PlacedOn></ns2:PlacedOn>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:TransactionTypeCode></ns2:TransactionTypeCode>
<ns2:CurrencyConversionType></ns2:CurrencyConversionType>
<ns2:CurrencyConversionRate></ns2:CurrencyConversionRate>
<ns2:CurrencyConversionDate></ns2:CurrencyConversionDate>
<ns2:TransactionDocumentTypeCode></ns2:TransactionDocumentTypeCode>
<ns2:CancelReasonCode></ns2:CancelReasonCode>
<ns2:CancelReason></ns2:CancelReason>
<ns2:RequestCancelDate></ns2:RequestCancelDate>
<ns2:Comments></ns2:Comments>
<ns2:BatchIdentifier></ns2:BatchIdentifier>
<ns2:RequestingLegalUnitIdentifier>204</ns2:RequestingLegalUnitIdentifier>
<ns2:RequestingLegalUnit></ns2:RequestingLegalUnit>
<ns2:OrigSystemDocumentReference>ManualShipmentProcess</ns2:OrigSystemDocumentReference>
<ns2:InterfaceStatus></ns2:InterfaceStatus>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:FreezePriceFlag>true</ns2:FreezePriceFlag>
<ns2:OperationMode></ns2:OperationMode>
<ns2:PreCreditCheckedFlag>true</ns2:PreCreditCheckedFlag>
<ns2:OrchestrationOrderRequestLine>
<ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
<ns2:ProductIdentifier>149</ns2:ProductIdentifier>
<ns2:ProductNumber></ns2:ProductNumber>
<ns2:ProductName></ns2:ProductName>
<ns2:OrderedQuantity>10</ns2:OrderedQuantity>
<ns2:CanceledQuantity></ns2:CanceledQuantity>
<ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
<ns2:RequestedFulfillmentOrganizationIdentifier>204</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestedFulfillmentOrganizationCode></ns2:RequestedFulfillmentOrganizationCode>
<ns2:BusinessUnitIdentifier></ns2:BusinessUnitIdentifier>
<ns2:BusinessUnitName></ns2:BusinessUnitName>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:RequestingBusinessUnitName></ns2:RequestingBusinessUnitName>
<ns2:CancelReasonCode></ns2:CancelReasonCode>
<ns2:CancelReason></ns2:CancelReason>
<ns2:SubstitutionAllowedFlag></ns2:SubstitutionAllowedFlag>
<ns2:SubstitutionReasonCode></ns2:SubstitutionReasonCode>
<ns2:CustomerPONumber></ns2:CustomerPONumber>
<ns2:CustomerPOLineNumber></ns2:CustomerPOLineNumber>
<ns2:CustomerPOScheduleNumber></ns2:CustomerPOScheduleNumber>
<ns2:CustomerProductIdentifier></ns2:CustomerProductIdentifier>
<ns2:TransactionLineTypeCode></ns2:TransactionLineTypeCode>
<ns2:ParentLineReference></ns2:ParentLineReference>
<ns2:RootParentLineReference></ns2:RootParentLineReference>
<ns2:ShippingInstructions>XYZ</ns2:ShippingInstructions>
<ns2:PackingInstructions>XYZ</ns2:PackingInstructions>
<ns2:InvoicingRuleCode></ns2:InvoicingRuleCode>
<ns2:InvoicingRule></ns2:InvoicingRule>
<ns2:AccountingRuleCode></ns2:AccountingRuleCode>
<ns2:AccountingRule></ns2:AccountingRule>
<ns2:RequestedShipDate></ns2:RequestedShipDate>
<ns2:RequestedArrivalDate>2015-11-15T10:10:10</ns2:RequestedArrivalDate>
<ns2:ScheduleShipDate>2015-11-15T10:10:10</ns2:ScheduleShipDate>
<ns2:ScheduleArrivalDate>2015-11-15T10:10:10</ns2:ScheduleArrivalDate>
<ns2:DemandClassCode></ns2:DemandClassCode>
<ns2:DemandClass></ns2:DemandClass>
<ns2:IncotermCode></ns2:IncotermCode>
<ns2:Incoterm></ns2:Incoterm>
```

```
<ns2:ShippingCarrierCode></ns2:ShippingCarrierCode>
<ns2:ShippingCarrier></ns2:ShippingCarrier>
<ns2:PaymentTerms></ns2:PaymentTerms>
<ns2:PaymentTermsCode>1</ns2:PaymentTermsCode>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:ReturnReasonCode></ns2:ReturnReasonCode>
<ns2:ReturnReason></ns2:ReturnReason>
<ns2:ShippingServiceLevelCode></ns2:ShippingServiceLevelCode>
<ns2:ShippingServiceLevel></ns2:ShippingServiceLevel>
<ns2:ShippingModeCode></ns2:ShippingModeCode>
<ns2:ShippingMode></ns2:ShippingMode>
<ns2:ShipmentPriorityCode></ns2:ShipmentPriorityCode>
<ns2:ShipmentPriority></ns2:ShipmentPriority>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:InventoryOrganization></ns2:InventoryOrganization>
<ns2:FreightTermsCode></ns2:FreightTermsCode>
<ns2:FreightTerms></ns2:FreightTerms>
<ns2:RequestCancelDate></ns2:RequestCancelDate>
<ns2:OriginalProductIdentifier></ns2:OriginalProductIdentifier>
<ns2:OriginalProductName></ns2:OriginalProductName>
<ns2:OriginalProductNumber></ns2:OriginalProductNumber>
<ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
<ns2:ShipToPartyName></ns2:ShipToPartyName>
<ns2:ShipToPartyNumber></ns2:ShipToPartyNumber>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:ShipToAddress1></ns2:ShipToAddress1>
<ns2:ShipToAddress2></ns2:ShipToAddress2>
<ns2:ShipToAddress3></ns2:ShipToAddress3>
<ns2:ShipToAddress4></ns2:ShipToAddress4>
<ns2:ShipToCity></ns2:ShipToCity>
<ns2:ShipToPostalCode></ns2:ShipToPostalCode>
<ns2:ShipToState></ns2:ShipToState>
<ns2:ShipToProvince></ns2:ShipToProvince>
<ns2:ShipToCountry></ns2:ShipToCountry>
<ns2:ShipToContactPartyIdentifier></ns2:ShipToContactPartyIdentifier>
<ns2:ShipToContactPartyNumber></ns2:ShipToContactPartyNumber>
<ns2:ShipToContactPartyName></ns2:ShipToContactPartyName>
<ns2:BillToCustomerIdentifier>1002</ns2:BillToCustomerIdentifier>
<ns2:BillToCustomerName></ns2:BillToCustomerName>
<ns2:BillToCustomerNumber></ns2:BillToCustomerNumber>
<ns2:BillToAccountSiteUseIdentifier>1009</ns2:BillToAccountSiteUseIdentifier>
<ns2:BillToAddress1></ns2:BillToAddress1>
<ns2:BillToAddress2></ns2:BillToAddress2>
<ns2:BillToAddress3></ns2:BillToAddress3>
<ns2:BillToAddress4></ns2:BillToAddress4>
<ns2:BillToCity></ns2:BillToCity>
<ns2:BillToPostalCode></ns2:BillToPostalCode>
<ns2:BillToState></ns2:BillToState>
<ns2:BillToProvince></ns2:BillToProvince>
<ns2:BillToCountry></ns2:BillToCountry>
<ns2:BillToAccountContactIdentifier></ns2:BillToAccountContactIdentifier>
<ns2:BillToAccountContactName></ns2:BillToAccountContactName>
<ns2:BillToAccountContactNumber></ns2:BillToAccountContactNumber>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:FulfillmentLineIdentifier></ns2:FulfillmentLineIdentifier>
<ns2:ShipToRequestRegion></ns2:ShipToRequestRegion>
<ns2:RequestedSupplierCode></ns2:RequestedSupplierCode>
<ns2:RequestedSupplierName></ns2:RequestedSupplierName>
<ns2:RequestedSupplierNumber></ns2:RequestedSupplierNumber>
<ns2:RequestedSupplierSiteCode></ns2:RequestedSupplierSiteCode>
<ns2:SupplierAddressLine1></ns2:SupplierAddressLine1>
<ns2:SupplierAddressLine2></ns2:SupplierAddressLine2>
<ns2:SupplierAddressLine3></ns2:SupplierAddressLine3>
<ns2:SupplierAddressLine4></ns2:SupplierAddressLine4>
<ns2:SupplierAddressCity></ns2:SupplierAddressCity>
<ns2:SupplierAddressState></ns2:SupplierAddressState>
```

```
<ns2:SupplierAddressZipCode></ns2:SupplierAddressZipCode>
<ns2:SupplierAddressProvince></ns2:SupplierAddressProvince>
<ns2:SupplierAddressCountry></ns2:SupplierAddressCountry>
<ns2:FulfillmentMethodCode></ns2:FulfillmentMethodCode>
<ns2:Comments></ns2:Comments>
<ns2:ReferenceTransactionLineId></ns2:ReferenceTransactionLineId>
<ns2:InterfaceStatus></ns2:InterfaceStatus>
<ns2:UnitListPrice>5</ns2:UnitListPrice>
<ns2:UnitSellingPrice>5</ns2:UnitSellingPrice>
<ns2:ExtendedAmount>100</ns2:ExtendedAmount>
<ns2:BatchIdentifier></ns2:BatchIdentifier>
<ns2:DestinationShippingOrganizationIdentifier></ns2:DestinationShippingOrganizationIdentifier>
<ns2:DestinationShippingLocationIdentifier></ns2:DestinationShippingLocationIdentifier>
<ns2:EarliestAcceptableShipDate></ns2:EarliestAcceptableShipDate>
<ns2:LatestAcceptableShipDate></ns2:LatestAcceptableShipDate>
<ns2:EarliestAcceptableArrivalDate></ns2:EarliestAcceptableArrivalDate>
<ns2:LatestAcceptableArrivalDate></ns2:LatestAcceptableArrivalDate>
<ns2:PromiseShipDate></ns2:PromiseShipDate>
<ns2:PromiseArrivalDate></ns2:PromiseArrivalDate>
<ns2:SubInventoryCode></ns2:SubInventoryCode>
<ns2:SubInventory></ns2:SubInventory>
<ns2:ShipSetName></ns2:ShipSetName>
<ns2:TaxExemptFlag>S</ns2:TaxExemptFlag>
<ns2:TaxClassificationCode></ns2:TaxClassificationCode>
<ns2:TaxExemptionCertificateNumber></ns2:TaxExemptionCertificateNumber>
<ns2:TaxExemptReasonCode></ns2:TaxExemptReasonCode>
<ns2:DefaultTaxationCountry></ns2:DefaultTaxationCountry>
<ns2:FirstPartyTaxRegistration></ns2:FirstPartyTaxRegistration>
<ns2:ThirdPartyTaxRegistration></ns2:ThirdPartyTaxRegistration>
<ns2:DocumentSubType></ns2:DocumentSubType>
<ns2:FinalDischargeLocationIdentifier></ns2:FinalDischargeLocationIdentifier>
<ns2:ProductFiscalCategoryIdentifier></ns2:ProductFiscalCategoryIdentifier>
<ns2:ProductType></ns2:ProductType>
<ns2:ProductCategory></ns2:ProductCategory>
<ns2:TransactionBusinessCategory></ns2:TransactionBusinessCategory>
<ns2:AssessableValue></ns2:AssessableValue>
<ns2:UserDefinedFiscClass></ns2:UserDefinedFiscClass>
<ns2:IntendedUseClassificationIdentifier></ns2:IntendedUseClassificationIdentifier>
<ns2:FOBPointCode></ns2:FOBPointCode>
<ns2:FOBPoint></ns2:FOBPoint>
<ns2:OrigSystemDocumentReference>ORIGSYS</ns2:OrigSystemDocumentReference>
<ns2:OrigSystemDocumentLineReference>ORIGSYSLINE</ns2:OrigSystemDocumentLineReference>
<ns2:CreditCheckAuthorizationCode>1235</ns2:CreditCheckAuthorizationCode>
<ns2:CreditCheckAuthorizationExpiryDate>2016-11-15T10:10:10</ns2:CreditCheckAuthorizationExpiryDate>
<ns2:ContractStartDate></ns2:ContractStartDate>
<ns2:LineCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>1</ns2:PricedQuantity>
<ns2:PricedQuantityUOMCode>Ea</ns2:PricedQuantityUOMCode>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode></ns2:PricePeriodicityCode>
<ns2:GsaUnitPrice></ns2:GsaUnitPrice>
<ns2:ChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>97</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
```

```
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>97</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>97</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>97</ns2:ChargeCurrencyExtendedAmount>
</ns2:ChargeComponent>
<ns2:ChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>97</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>97</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>97</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>97</ns2:ChargeCurrencyExtendedAmount>
</ns2:ChargeComponent>
</ns2:LineCharge>
</ns2:OrchestrationOrderRequestLine>


</ns1:OrchestrationOrderRequest>
</ns1:process>
</soap:Body>
</soap:Envelope>
```

Related Topics

- Set Up Credit Check
- Overview of Importing Orders Into Order Management
- Reauthorize Payment
- Manage Credit Check

Extend Credit Check

Use an order management extension to modify credit check.

Skip Credit Check for Order Revisions

Use the PreCreditCheckedFlag attribute in an order management extension to skip credit check when the Order Entry Specialist revises the sales order. For example:

- If the sales order is a change order, and if Credit Management isn't enabled, then set PreCreditCheckedFlag to true.

Setting PreCreditCheckedFlag to true instructs Order Management to skip credit check.

Here's your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
Boolean isCCMgmtEnabled = false;
BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
if(refHeaderId != null)
{
```

```
def lookupVO =context.getViewObject("oracle.apps.financials.assets.shared.publicView.LookupPVO");
def vc = lookupVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("LookupType", 'AR_FEATURES');//ACCOUNT_TYPE AR_FEATURES
vcrow.setAttribute("EnabledFlag", 'Y');
vcrow.setAttribute("LookupCode", 'AR_CREDIT_MGMT');
def rowset = lookupVO.findByViewCriteria(vc, -1);
rowset.reset();
if(rowset.hasNext()){
isCCMgmtEnabled = true;
}
if(!isCCMgmtEnabled)
header.setAttribute("PreCreditCheckedFlag", "Y");
}
```

Skip Credit Check When Lines Are On Hold

Skip credit check when an order line in a change order is on credit check hold. For example:

- If the sales order is a change order, and if no order lines in the original order are on credit check hold, and if Credit Management isn't enabled, then set PreCreditCheckedFlag to true.

Here's your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
Boolean isCCMgmtEnabled = false;
BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
Boolean ccHoldFoundOnOriginalLines = false;
if(refHeaderId != null)
{
def holdInstanceVO =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HoldInstancePVO");
def hodlInstanceVc = holdInstanceVO.createViewCriteria();
def holdInstanceVrow = hodlInstanceVc.createViewCriteriaRow();
holdInstanceVrow.setAttribute("HeaderHeaderId", refHeaderId);
holdInstanceVrow.setAttribute("FulfillLineOnHold", 'Y');
def holdInstanceRowset = holdInstanceVO.findByViewCriteria(hodlInstanceVc, -1);
holdInstanceRowset.reset();
while(holdInstanceRowset.hasNext())
{
def holdInstance = holdInstanceRowset.next();
def holdDefs = holdInstance.getAttribute("HoldDefinition");
String holdCode = holdDefs.getAttribute("HoldHoldCode");
if("DOO_CREDIT_CHECK".equalsIgnoreCase(holdCode))
{
ccHoldFoundOnOriginalLines = true;
break;
}
}
if(!ccHoldFoundOnOriginalLines)
{
def lookupVO =context.getViewObject("oracle.apps.financials.assets.shared.publicView.LookupPVO");
def vc = lookupVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("LookupType", 'AR_FEATURES');//ACCOUNT_TYPE AR_FEATURES
vcrow.setAttribute("EnabledFlag", 'Y');
vcrow.setAttribute("LookupCode", 'AR_CREDIT_MGMT');
def rowset = lookupVO.findByViewCriteria(vc, -1);
rowset.reset();
if(rowset.hasNext())
{
isCCMgmtEnabled = true;
}
if(!isCCMgmtEnabled)
header.setAttribute("PreCreditCheckedFlag", "Y");
}
```

```
}
```

Manage the Authorization Expiration Date

If you set the PreCreditCheckedFlag to Y in an order management extension to skip credit check, then you must also set the CreditCheckAuthorizationExpiryDate attribute on the fulfillment line to `null`. A null value means the authorization never expires. If you don't do this, and if you revise the sales order, then Order Management might run credit check again because the authorization expired.

Here's how:

```
if ("CC".equals(header.getAttribute("CustomerPONumber"))) {
 if ("Y".equals(header.getAttribute("PreCreditCheckedFlag"))) {
 def lines = header.getAttribute("Lines");
 while (lines.hasNext()) {
 def line = lines.next();
 def creditAuthExpDate = line.getAttribute("CreditCheckAuthorizationExpiryDate");

 if (creditAuthExpDate != null) {
 java.sql.Date currentDate = new java.sql.Date((new Date()).getTime());
 java.sql.Date creditAuthExpSqlDate = new java.sql.Date((creditAuthExpDate).getTime());

 if (creditAuthExpSqlDate < currentDate) {
 //line.setAttribute("CreditCheckAuthorizationExpiryDate", null);
 line.setAttribute("CreditCheckAuthorizationExpiryDate", currentDate);
 }
 }
 }
 }
}
```

As an alternative, you can set the Expiration Offset Days attribute on the customer account to a large value so the authorization doesn't expire while you're revising the sales order. For details, see *How You Set Up a Customer Profile for Credit Checks*.

Related Topics

- Overview of Creating Order Management Extensions

Skip Credit Check

If you implement credit check in Order Management but your fulfillment line fails credit check when you revise a sales order, you might notice that the status on the fulfillment line gets stuck at Credit Review Pending and the line doesn't move along in fulfillment with the other lines in the sales order. You can fix this problem.

You can do one of:

1. Pay or revise.
    - Send payment to the invoice for the Bill To Customer that's on the sales order.
    - Revise the credit limit.
2. Change the revision to Draft status, then resubmit it.

3. Skip credit check altogether on the sales order. There are two different ways to do this.
    - Use order import to set the Credit Authorized in Source attribute to Y for the revised sales order. For details, see *Import Source Orders That Include Credit Check*.
    - Use an order management extension to set the PreCreditCheckedFlag attribute to Y for the revised sales order. The extension makes sure Order Management skips calls to credit check for the revision according to a condition that you specify in the extension.
      Note that if authorization has expired, and if the order line isn't closed, then Order Management will run the credit check even if you set PreCreditCheckedFlag to Y.

This topic describes different ways to use an order management extension to skip credit check for a single sales order. For details about how to create an order management extension, see *Overview of Creating Order Management Extensions*.

Skip Credit Check on the Sales Order Revision

1. Create an extension on the On Save event and name it Skip Credit Check for Order Revisions.
2. Add this code to your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

Boolean isCCMgmtEnabled = false;
BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
if(refHeaderId != null)
{
 def lookupVO =context.getViewObject("oracle.apps.financials.assets.shared.publicView.LookupPVO");
 def vc = lookupVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("LookupType", 'AR_FEATURES');//ACCOUNT_TYPE AR_FEATURES
 vcrow.setAttribute("EnabledFlag", 'Y');
 vcrow.setAttribute("LookupCode", 'AR_CREDIT_MGMT');
 def rowset = lookupVO.findByViewCriteria(vc, -1);
 rowset.reset();
 if(rowset.hasNext()){
 isCCMgmtEnabled = true;
 }
 if(!isCCMgmtEnabled)
 header.setAttribute("PreCreditCheckedFlag", "Y");
}
```

Here's what it does.

- `if(refHeaderId != null)` determines whether the sales order is an original order or a revision.
- Determine whether credit check is enabled in Credit Management. If it isn't, then `header.setAttribute("PreCreditCheckedFlag", "Y"` enables it.

This example skips credit check for all revisions, including the first revision and any subsequent revision. If you need to skip credit check for only the latest revision, then you might need to modify this example or create a new extension.

Skip Credit Check on an Original Order

1. Create an extension on the On Save event and name it Skip Credit Check for Sales Orders.
2. Add this code to your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

Boolean isCCMgmtEnabled = false;
BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
Boolean ccHoldFoundOnOriginalLines = false;
if(refHeaderId != null)
```

```
{
 def holdInstanceVO =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HoldInstancePVO");
 def hodlInstanceVc = holdInstanceVO.createViewCriteria();
 def holdInstanceVrow = hodlInstanceVc.createViewCriteriaRow();
 holdInstanceVrow.setAttribute("HeaderHeaderId", refHeaderId);
 holdInstanceVrow.setAttribute("FulfillLineOnHold", 'Y');
 def holdInstanceRowset = holdInstanceVO.findByViewCriteria(hodlInstanceVc, -1);
 holdInstanceRowset.reset();
 while(holdInstanceRowset.hasNext())
 {
 def holdInstance = holdInstanceRowset.next();
 def holdDefs = holdInstance.getAttribute("HoldDefinition");
 String holdCode = holdDefs.getAttribute("HoldHoldCode");
 if("DOO_CREDIT_CHECK".equalsIgnoreCase(holdCode))
 {
 ccHoldFoundOnOriginalLines = true;
 break;
 }
 }
 if(!ccHoldFoundOnOriginalLines)
 {
 def lookupVO =context.getViewObject("oracle.apps.financials.assets.shared.publicView.LookupPVO");
 def vc = lookupVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();

 vcrow.setAttribute("LookupType", 'AR_FEATURES');//ACCOUNT_TYPE AR_FEATURES
 vcrow.setAttribute("EnabledFlag", 'Y');
 vcrow.setAttribute("LookupCode", 'AR_CREDIT_MGMT');
 def rowset = lookupVO.findByViewCriteria(vc, -1);
 rowset.reset();
 if(rowset.hasNext())
 {
 isCCMgmtEnabled = true;
 }
 if(!isCCMgmtEnabled)
 header.setAttribute("PreCreditCheckedFlag", "Y");
 }
}
```

Here's what it does.

- Determines whether there any lines in the original sales order that are on credit check hold. If not, exit the extension.
- Determines whether credit check is enabled in Credit Management. If it isn't, then `header.setAttribute("PreCreditCheckedFlag", "Y"` enables it.

Skip Credit Check When You Use an Extensible Flexfield

1. Set up your flexfield.
    - Edit the predefined Header Information extensible flexfield.
    - On the Edit Extensible Flexfield page, Select **Additional Header Information** in the Category area.
    - In the Associated Contexts area, create a new context.

Attribute	Value
Display Name	Skip Credit Check for Sales Orders That Have Flexfields

Attribute	Value
Associated Category	Additional Header Information
Behavior	Single Row

- Create a segment in the context that you just added.

Attribute	Value
Sequence	10
Name	Skip Credit Check
Table Column	ATTRIBUTE_CHAR1
Code	Skip Credit Check

For details about how to edit this flexfield, see *Overview of Using Extensible Flexfields in Order Management*.

2. Create an extension on the On Save event and name it Skip Credit Check for Sales Orders with Flexfields.
3. Add this code to your extension.

```
Boolean isCCMgmtEnabled = false;
BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
if(refHeaderId != null)
{
 def lookupVO =context.getViewObject("oracle.apps.financials.assets.shared.publicView.LookupPVO");
 def vc = lookupVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("LookupType", 'AR_FEATURES');//ACCOUNT_TYPE AR_FEATURES
 vcrow.setAttribute("EnabledFlag", 'Y');
 vcrow.setAttribute("LookupCode", 'AR_CREDIT_MGMT');
 def rowset = lookupVO.findByViewCriteria(vc, -1);
 rowset.reset();
 if(rowset.hasNext()){
 isCCMgmtEnabled = true;
 }
 if(!isCCMgmtEnabled)
 {

 def headerEFF = header.getContextRow("Override Credit Failure for Revision");
 if(headerEFF != null)
 {
 def segmentCode = headerEFF.getAttribute("skipCreditCheck");
 if(segmentCode != null)
 {
 if(segmentCode.equals("Y"))
 header.setAttribute("PreCreditCheckedFlag", "Y");
 else if(segmentCode.equals("N"))
 header.setAttribute("PreCreditCheckedFlag", "N");
 }
 }
 }
```

Here's what it does.

- Determines whether the sales order is an original order or a revision.

- Determines whether credit check is enabled in Credit Management. If it isn't, then it reads the Override Credit Failure for Revision flexfield context on the order header, and it reads the value in the Skip Credit Check segment. If the segment contains:
  - Y, the code sets PreCreditCheckedFlag to Y to skip credit check.
  - Not Y, the code sets PreCreditCheckedFlag to N, and Order Management will do the credit check.

Skip Credit Check According to the Total on the Order Revision

1. Create an extension on the On Save event and name it Skip Credit Check when Order Total Doesn't Change.
2. Add this code to your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String headerId = header.getAttribute("HeaderId");
String refHeaderId = header.getAttribute("ReferenceHeaderId");

//skip if its new order
if( refHeaderId == null ) return;

String currTotal,refTotal;

currTotal= getTotalValue(headerId);
refTotal= getTotalValue(refHeaderId);

if(currTotal.equals(refTotal)) {
//If the total is the same, then skip credit check.
header.setAttribute("PreCreditCheckedFlag","Y");
} else {
header.setAttribute("PreCreditCheckedFlag","N");
}
Object getTotalValue(String header_id){
String soTotal;
def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.OrderTotalPVO");
def vc = vo.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("HeaderHeaderId", header_id); // it was refHeaderId earlier
vcrow.setAttribute("OrderTotalPrimaryFlag", "Y");
vc.add(vcrow);
def rowset = vo.findByViewCriteria(vc,-1);
def totalSet = rowset.first();
if(totalSet!=null)
soTotal = totalSet.getAttribute("OrderTotalTotalAmount");
return soTotal;
}
```

Here's what it does.

- Determines whether the sales order is an original order or a revision.
- Determines whether the total in the revision matches the total in the original sales order. If yes, then Order Management runs a credit check on the revision. If no, then Order Management skips the check.
- This extension doesn't determine whether credit management is enabled.

Skip Credit Check According to the Payment Term on the Order Revision

1. Create an extension on the On Save event and name it Skip Credit Check According to Payment Term.
2. Add this code to your extension.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
String customerPONumber = header.getAttribute("CustomerPONumber");
if(customerPONumber.equals("RD_TEST"))
```

```
{
 BigDecimal headerId = header.getAttribute("HeaderId");
 BigDecimal refHeaderId = header.getAttribute("ReferenceHeaderId");
 Boolean ccHoldFoundOnOriginalLines = false;
 if(refHeaderId != null)
 {
 BigDecimal currentOrderAmount = new BigDecimal(0);
 BigDecimal refTotalAmount = new BigDecimal(0);

 def orderTotals = header.getAttribute("OrderTotals");
 orderTotals.reset();
 while(orderTotals.hasNext()) {
 def orderTotal = orderTotals.next();
 String totalCode = orderTotal.getAttribute("TotalCode");
 if(totalCode.equals("QP_TOTAL_PAY_NOW"))
 {
 currentOrderAmount = orderTotal.getAttribute("TotalAmount");
 }
 }
 def headereVO =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HeaderPVO");
 def headereVC = headereVO.createViewCriteria();
 def headerVCRow = headereVC.createViewCriteriaRow();

 headerVCRow.setAttribute("HeaderId", refHeaderId);//ACCOUNT_TYPE AR_FEATURES
 def headerRowset = headereVO.findByViewCriteria(headereVC, -1);
 headerRowset.reset();
 if(headerRowset.hasNext())
 {
 def refHeader = headerRowset.next();
 def refOrderTotlasVO =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.OrderTotalPVO");
 def refOrderTotlasVC = refOrderTotlasVO.createViewCriteria();
 def refOrderTotlasVCRow = refOrderTotlasVC.createViewCriteriaRow();
 refOrderTotlasVCRow.setAttribute("OrderTotalHeaderId", refHeader.getAttribute("HeaderId"));//
ACCOUNT_TYPE AR_FEATURES
 def refOrderTotlasRowset = refOrderTotlasVO.findByViewCriteria(refOrderTotlasVC, -1);
 refOrderTotlasRowset.reset();
 while(refOrderTotlasRowset.hasNext()) {
 def refOrderTotal = refOrderTotlasRowset.next();
 String refTotalCode = refOrderTotal.getAttribute("OrderTotalTotalCode");
 if(refTotalCode.equals("QP_TOTAL_PAY_NOW"))
 {
 refTotalAmount = refOrderTotal.getAttribute("OrderTotalTotalAmount");
 }
 }
 }
 if(currentOrderAmount.compareTo(refTotalAmount) == 0)
 {
 def hasPaymentTermChange = false;
 def lines = header.getAttribute("Lines");
 while( lines.hasNext() )
 {
 def line = lines.next();
 def fulfillLinePaymentTermId = line.getAttribute("PaymentTermCode");
 //header.setAttribute("Comments",fulfillLinePaymentTermId);

 def refFlineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");
 // header.setAttribute("Comments","1");
 def refFlineVO =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 def refFlineVC = refFlineVO.createViewCriteria();
 def refFlineVCRow = refFlineVC.createViewCriteriaRow();
 refFlineVCRow.setAttribute("FulfillLineId", refFlineId);
 def refFlineRowset = refFlineVO.findByViewCriteria(refFlineVC, -1);
 refFlineRowset.reset();
 if(refFlineRowset.hasNext())
 {
 def refFline = refFlineRowset.next();
```

```
def refPaymentTerm = refFline.getAttribute("FulfillLinePaymentTermId");
if(fulfillLinePaymentTermId != refPaymentTerm)
{
hasPaymentTermChange = true;
}
}
}
if(!hasPaymentTermChange)
header.setAttribute("PreCreditCheckedFlag", "Y");
}
}
}
```

Here's what it does.

- Determines whether the sales order is an original order or a revision.
- Determines whether the total in the revision matches the total in the original sales order, and whether the payment term on the order header of the revision matches the payment term on the original order.
  - If the total or the payment term doesn't match, then Order Management does the credit check.
  - If the total and the payment term do match, then Order Management skips the credit check.
- This extension doesn't determine whether credit management is enabled.

CustomerPONumber contains RD_TEST for testing purposes. You must remove this value or use your own value.

Disable Credit Check for All Sales Orders

You can disable credit check for all your sales orders.

1. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Financials
   - Functional Area: Receivables
   - Task: Manage Receivables Lookups
2. On the Manage Receivables Lookups page, search for the value.

Attribute	Value
Lookup Type	AR_FEATURES

3. In the search results, in the Financials Generic Lookup Type area, in the row that has AR_CREDIT_MGMT in the Lookup Code column, remove the check mark from the Enabled attribute.

Credit Card

Overview of Setting Up Credit Cards

Integrate Oracle Order Management with Oracle Payments so your users can select an existing credit card or add a new one to a sales order.

Here's a summary of what you can do.

- Allow your users to select an existing credit card or add a new one to a sales order, with or without the three digit card verification value.
- Import a transaction that's already authorized.
- Import a transaction and use the primary card that's already available on the customer account.
- Use a digital verification code when you use a credit card to pay for the sales order.
- The Order Entry Specialist can select a credit card on the sales order in the Order Management work area from a list of cards that are available for the bill-to-customer. Use a single credit card to pay for the entire sales order, or use a different credit card on each order line.
- Use a digital verification code for each credit card.
- If you set the Payment Method attribute on the sales order to Credit Card, then the Payment Details area displays the card that you specify as the primary card for the account in the Accounts Receivable work area. Note that you can use only one payment method for each sales order.
- If you enable the Digital Verification Code attribute in Oracle Payments, then the Payment Details area on the order header displays the Security Code attribute. The Override Order Line dialog also displays the Security Code attribute.
- Use an order management extension to prevent the Order Entry Specialist from editing a sales order that's already authorized.
- Use the integration to make sure your implementation complies with the Payment Card Industry Data Security Standard (PCI DSS).

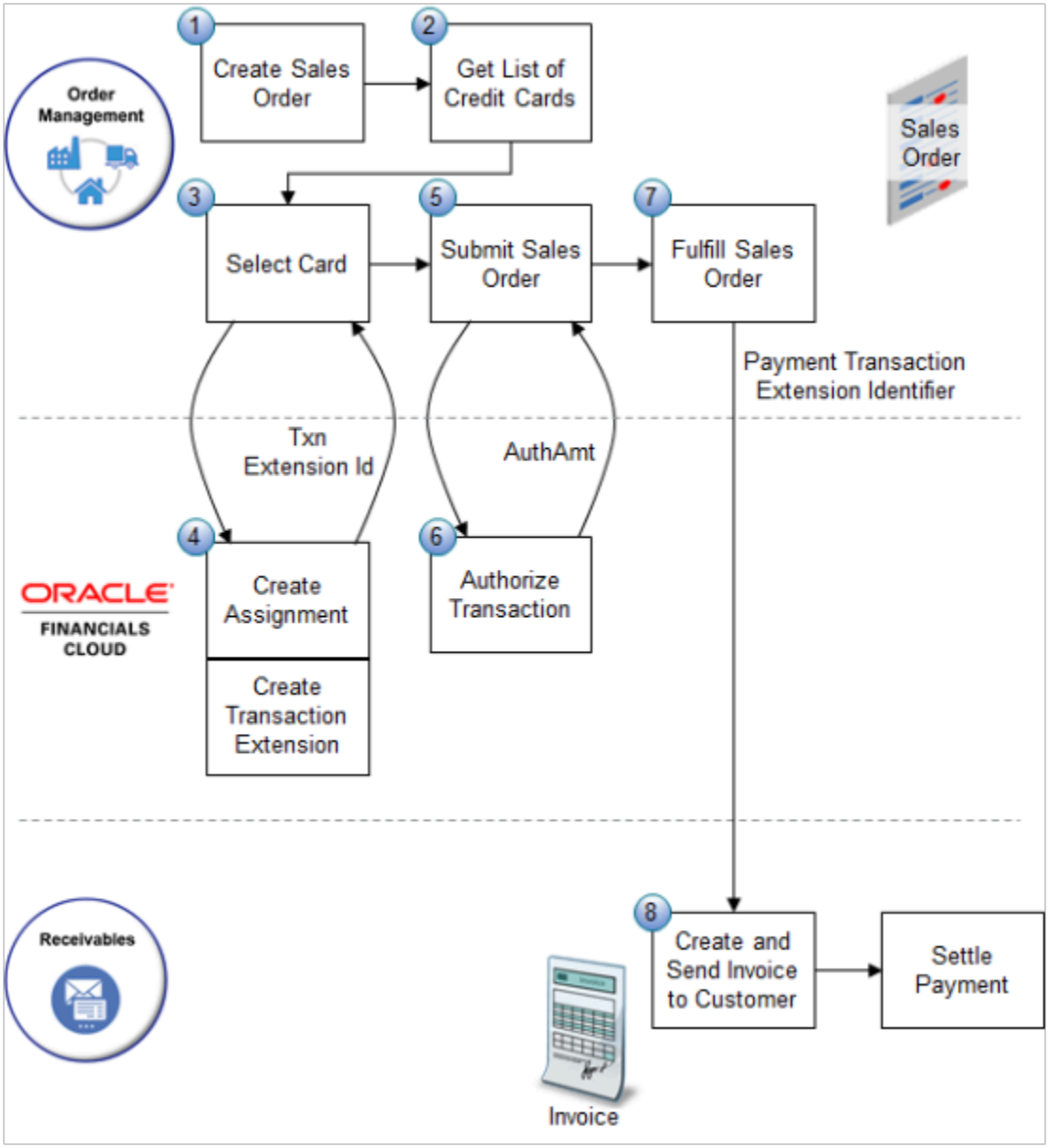
Note

- You must not send credit card numbers that aren't tokenized to Oracle Applications.
- You must consult with your own Qualified Security Assessor (QSA) to make sure your deployment complies with the Payment Card Industry Data Security Standard (PCI DSS) and the compliance requirements set by the PCI Security Standards Council.

How it Works

Order Management uses Oracle Payments to manage and process credit card transactions. Oracle Payments handles the details, such as authorizing the purchase, communicating with the bank that issues the credit card, making sure funds are available for the credit card, that the card hasn't expired, and so on.

Here's how the flow works for credit cards that already exist.



What the Numbers Mean

1. The Order Entry Specialist creates a sales order in the Order Management work area.
2. Order Management gets the list of credit cards that are currently active for the customer account, then makes them available in the drop list for the Credit Card attribute in the Payment Details area of the Billing and Payment details tab of the sales order.

   Order Management gets these cards from the Manage Customer page in Trading Community Architecture. You can use the Bill-to Account attribute or Bill-to Customer Site attribute in Trading Community Architecture to filter the list of cards. For details, see *Overview of Displaying Customer Details on Sales Orders*.
3. The Order Entry Specialist selects the card in the Credit Card attribute.

   If you enable Voice Authorization, and if the Order Entry Specialist adds voice authorization details, then Order Management sends them to Oracle Financials.

4. Oracle Financials creates an assignment for the card and a transaction extension for the transaction, then sends the `Txn Extension Id` attribute to Order Management.
5. The Order Entry Specialist adds an order line to the sales order, then clicks Submit.
6. Oracle Payments attempts to authorize the transaction.

If Authorization	Then Payments Sends
Succeeds	Authorization details, including the AuthAmt
Fails	The reasons why authorization failed

   For details, see *How Authorizations for Credit Cards Are Processed*.
7. Order Management fulfills the sales order, then sends fulfillment details including the Payment Transaction Extension Identifier to Oracle Receivables.
8. Oracle Receivables creates an invoice, sends it to the customer, and settles the payment with the customer.

The flow is similar for other scenarios, with a few variations.

Scenario	Variation
Order Entry Specialist creates a new credit card.	Oracle Financials creates and tokenizes the card in Financials. The card is now ready for the current and future sales orders.
You import a sales order that includes credit authorization details.	Oracle Financials works with your upstream order capture system to authorize credit, then sends the Payment Transaction Extension Identifier attribute to Order Management in the import payload. Order Management doesn't reauthorize credit.
You import a sales order that includes credit card details.	-

Prevent the Order Entry Specialist From Editing a Sales Order That's Already Authorized

You can use an order management extension to prevent the Order Entry Specialist from editing a sales order that's already authorized. Examine the Prevent Users From Editing Payment Attributes subtopic. For details, see *Extend Shipping*.

Related Topics

- Overview of Displaying Customer Details on Sales Orders
- Extend Shipping
- Enable the Credit Card Feature in Order Management
- How Authorizations for Credit Cards Are Processed

Enable the Credit Card Feature in Order Management

Learn how to enable the credit card feature in Order Management.

1. Log a service request with Oracle Support. Include details in the request.
    - Product: Oracle Payments Cloud Service
    - Subject: `Your Environment Name`: Request to Enable Credit Card Processing
2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Profiles
3. On the Manage Order Profiles page, search for the value.

Attribute	Value
Profile Option Code	ORA_FOM_ENABLE_CARD_PAYMENT

4. In the search results, in the Profile Option Levels area, set the values to meet your requirements.

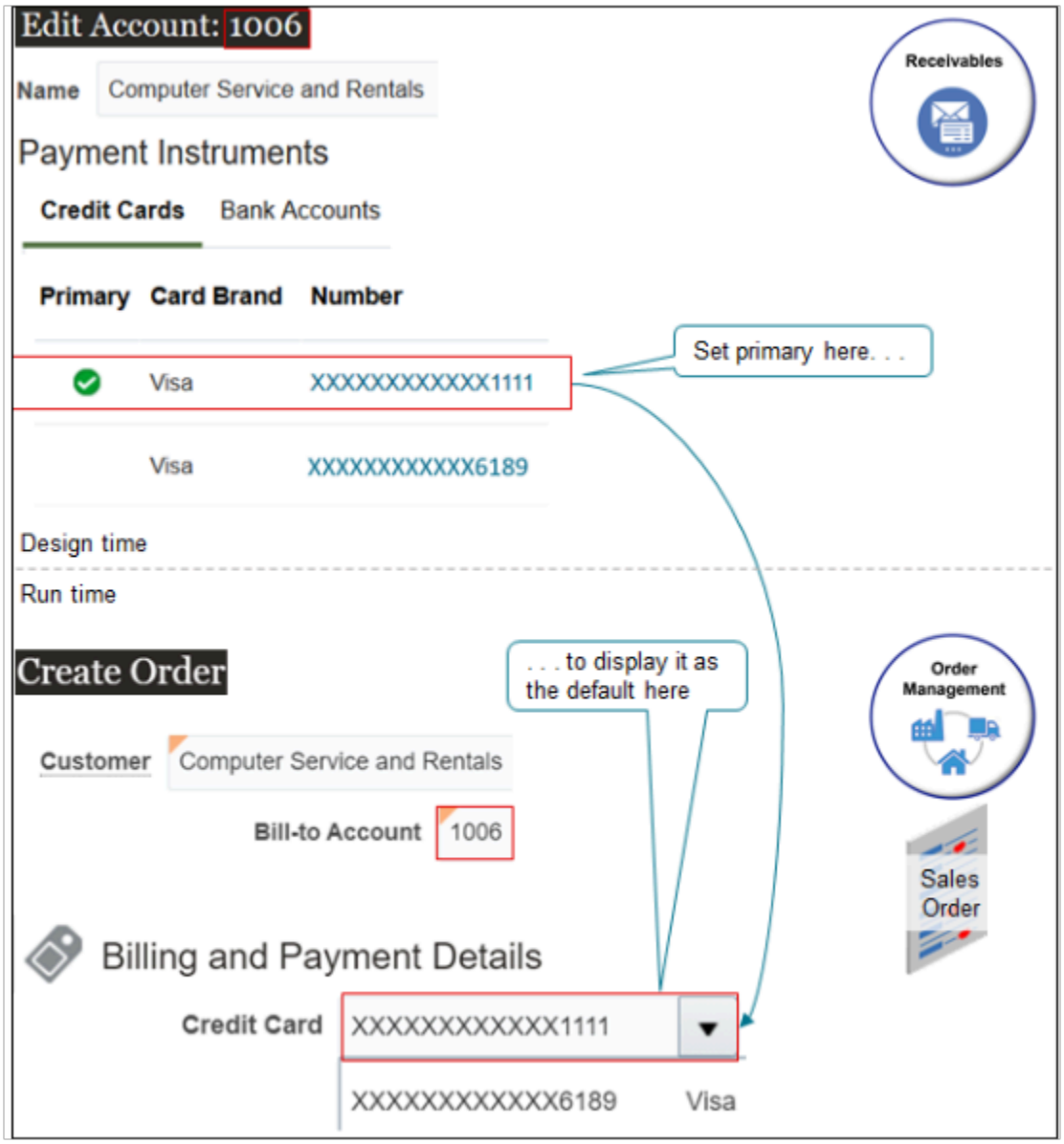
   For example:

Attribute	Value
Level	Site
Enabled	Contains a check mark.
Updatable	Contains a check mark.

That's it! You can now use credit cards in Order Management.

Add New Credit Card and Make it the Primary

Assume you need to add a credit card for your Computer Service and Rentals customer and make it the primary card.



Note

- You add a credit card to the account in the Accounts Receivable work area.
- You can set the card as the primary card when you add it.
- At run time, Order Management selects and displays the primary card, by default. It also makes the other cards that you add in the Accounts Receivable work area available on the sales order.

Summary of the Setup

1. Enable tokenization.
2. Add the credit card.
3. Test your setup.

Enable Tokenization

1. Make sure you have the privileges that you need to manage security. For example, it_security_manager.

2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Payments
    - Task: Manage System Security Options
3. On the Manage System Security Options page, click **Edit Tokenization Payment System**.
4. In the Edit Tokenization Payment System dialog, select the value, then click **Save and Close**.

Attribute	Value
Tokenization Payment System	CyberSource

5. In the Credit Card Data area, click **Tokenize**.

   Make sure all credit cards are tokenized. If they aren't, the Tokenize button isn't active and you can't use it.
6. Notice the process ID in the dialog. For this example, assume its 470870.
7. On the Encryption and Tokenization of Payment Instrument Sensitive Data page, verify that process 470870 succeeds.

   If you don't do this set up, you might encounter an error.

   ```
   You must enable credit card tokenization before creating a new customer credit card.
   ```

   The error will display when you attempt to add a credit card to the account in the Accounts Receivable work area, or when you attempt to add it to a sales order in the Order Management work area.

   For details, see *Use Credit Card Tokens to Improve Security*.

Add the Credit Card

1. Make sure you have the privileges that you need to manage Accounts Receivable.
2. Go to the Accounts Receivable work area.
3. On the Accounts Receivable page, click **Tasks > Manage Customers**.
4. On the Manage Customers page, search for the value.

Attribute	Value
Organization Name	Computer Service and Rentals

5. In the search results, click the link in the Account Number column.

Attribute	Value
Account Number	1006

6. On the Edit Account page, in the Payment Instruments area, in the Credit Cards list, click **Create a Credit Card**. Its the icon that looks like a page.

Test Your Setup

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Bill-to Account	1006 Notice that this is the same value you clicked on the Manage Customers page.

3. Click **Billing and Payment Details**.
4. On the Billing and Payment Details tab, set the value.

Attribute	Value
Payment Method	Credit Card

5. In the Payment Details area, verify that the attribute value defaults to the primary card.

Attribute	Value
Credit Card	XXXXXXXXXXXX1111

Related Topics

- Options for System Security

Import Credit Cards Into Order Management

Use a web service to import credit cards into Order Management.

Example Payload That Includes Authorization Details

Include the Payment entity in your payload, and make sure you set the PaymentMethod attribute to ORA_CREDIT_CARD. For example:

```
<ns2:Payment>
 <ns2:PaymentMethodCode>30</ns2:PaymentMethodCode>
```

```
    <ns2:PaymentTransactionIdentifier>101</ns2:PaymentTransactionIdentifier>
    <ns2:PaymentSetIdentifier>56001</ns2:PaymentSetIdentifier>

    <ns2:SourceTransactionPaymentIdentifier>H101</ns2:SourceTransactionPaymentIdentifier>
    <ns2:PaymentMethod/>
    <ns2:PaymentTypeCode>ORA_CREDIT_CARD</ns2:PaymentTypeCode>

   </ns2:Payment>
```

Here's the entire payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:BatchName/>
 <ns2:Order>
 <ns2:SourceTransactionIdentifier>RS_SO_0901_PreAuthCC03</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionSystem>LEG</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionNumber>RS_SO_0901_PreAuthCC03</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:BuyingPartyContactId>1560</ns2:BuyingPartyContactId>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2021-09-20T06:08:52.0340</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:RequestingLegalUnitIdentifier/>
 <ns2:FreezePriceFlag>false</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>false</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>false</ns2:FreezeTaxFlag>
 <ns2:CustomerPONumber/>
 <ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
 <ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
 <ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
 <ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
 <ns2:BillToCustomerName/>
 <ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
 <ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
 <ns2:OrigSystemDocumentReference>DOO_BillOnlyGenericProcess</ns2:OrigSystemDocumentReference>
 <ns2:Line>
 <ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
 <ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
 <ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
 <ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
 <!--ns2:ProductIdentifier>2157</ns2:ProductIdentifier-->
 <ns2:ProductNumber>AS92888</ns2:ProductNumber>
 <!--ns2:SourceSystemProductReference>2157</ns2:SourceSystemProductReference-->
 <!--ns2:SourceSystemProductReference>2157</ns2:SourceSystemProductReference-->
 <ns2:OrderedQuantity>10</ns2:OrderedQuantity>
 <ns2:OrderedUOM>Each</ns2:OrderedUOM>
 <ns2:OperationMode/>
 <ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:RequestedShipDate>2018-11-18T06:08:52.0340</ns2:RequestedShipDate>
 <ns2:ScheduleShipDate>2018-11-20T06:08:52.0340</ns2:ScheduleShipDate>
 <!--Optional:-->
 <ns2:ScheduleArrivalDate>2018-11-20T06:08:52.0340</ns2:ScheduleArrivalDate>
 <!--Optional:-->
 <ns2:PaymentTermsCode>1</ns2:PaymentTermsCode>
 <ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
 <ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
 <ns2:ShipToPartySiteIdentifier>1220</ns2:ShipToPartySiteIdentifier>
 <ns2:ShipToPartyContactName/>
 <ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
 <ns2:BillToCustomerName>1006</ns2:BillToCustomerName>
 <ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
 <ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
```

```
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:ShippingInstructions/>
<ns2:UnitListPrice currencyCode="USD">105</ns2:UnitListPrice>
<ns2:PartialShipAllowedFlag>FALSE</ns2:PartialShipAllowedFlag>
<ns2:FOBPoint>Destination</ns2:FOBPoint>
<ns2:InvoicingRuleCode/>
<!--Optional:-->
<!--ns2:InvoicingRule>Immediate</ns2:InvoicingRule-->
<!--Optional:-->
<ns2:AccountingRuleCode/>
<!--Optional:-->
<!--ns2:AccountingRule>Advance Invoice</ns2:AccountingRule-->
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>1</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>1000</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>10</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>10</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>100</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>10001</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>90</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>9</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>9</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
</ns2:Line>
<ns2:Payment>
<ns2:PaymentMethodCode>30</ns2:PaymentMethodCode>
<ns2:PaymentTransactionIdentifier>101</ns2:PaymentTransactionIdentifier>
<ns2:PaymentSetIdentifier>56001</ns2:PaymentSetIdentifier>
<ns2:SourceTransactionPaymentIdentifier>H101</ns2:SourceTransactionPaymentIdentifier>
```

```
 <ns2:PaymentMethod/>
 <ns2:PaymentTypeCode>ORA_CREDIT_CARD</ns2:PaymentTypeCode>

 </ns2:Payment>
 <ns2:OrderPreferences>
 <!--Optional:-->
 <ns2:CreateCustomerInformationFlag>false</ns2:CreateCustomerInformationFlag>
 <!--Optional:-->
 <ns2:SubmitFlag>true</ns2:SubmitFlag>
 </ns2:OrderPreferences>
 </ns2:Order>
 </ns1:request>
 </ns1:createOrders>
 </soap:Body>
</soap:Envelope>
```

Example Payload That Includes Credit Card Details

Include the Payment entity in your payload, and make sure you set the PaymentTypeCode attribute to ORA_CREDIT_CARD. For example:

```
<ns2:Payment>

 <ns2:PaymentTypeCode>ORA_CREDIT_CARD</ns2:PaymentTypeCode>
 <ns2:SourceTransactionPaymentIdentifier>H101</ns2:SourceTransactionPaymentIdentifier>

</ns2:Payment>
```

Here's the entire payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:BatchName/>
 <ns2:Order>
 <ns2:SourceTransactionIdentifier>RS_SO_0804_CC04</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionNumber>RS_SO_0804_CC04</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:BuyingPartyContactId>1560</ns2:BuyingPartyContactId>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2019-12-04T06:08:52.0340</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:RequestingLegalUnitIdentifier>204</ns2:RequestingLegalUnitIdentifier>
 <ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
 <ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
 <ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
 <ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
 <ns2:BillToCustomerName>1006</ns2:BillToCustomerName>
 <ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
 <ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
 <ns2:FreezePriceFlag>true</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>true</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>true</ns2:FreezeTaxFlag>
 <ns2:OrigSystemDocumentReference></ns2:OrigSystemDocumentReference>
 <ns2:Salesperson>James Seller</ns2:Salesperson>
 <ns2:Line>
 <ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
 <ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
 <ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
 <ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
 <ns2:ProductNumber>AS92888</ns2:ProductNumber>
 <ns2:OrderedQuantity unitCode="Ea">100</ns2:OrderedQuantity>
 <ns2:ShippingServiceLevelCode/>
```

```
<ns2:ShipmentPriorityCode/>
<ns2:ShipmentPriority/>
<ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
<ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:RequestedShipDate>2019-12-04T06:08:52.0340</ns2:RequestedShipDate>
<ns2:PaymentTermsCode>4</ns2:PaymentTermsCode>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:UnitListPrice currencyCode="USD">105</ns2:UnitListPrice>
<ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
<ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
<ns2:BillToCustomerName>1006</ns2:BillToCustomerName>
<ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
<ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
<ns2:PartialShipAllowedFlag>FALSE</ns2:PartialShipAllowedFlag>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>1</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>10001</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>100</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>100</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>3027</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>10001</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>10001</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>100</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>100</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
```

```
</ns2:OrderChargeComponent>
</ns2:OrderCharge>

</ns2:Line>

<ns2:Payment>

<ns2:PaymentTypeCode>ORA_CREDIT_CARD</ns2:PaymentTypeCode>
<ns2:SourceTransactionPaymentIdentifier>H101</ns2:SourceTransactionPaymentIdentifier>

</ns2:Payment>

</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Use Credit Card Tokens to Improve Security

Use tokens to improve security for the credit cards that you use with Order Management.

If you use credit cards to make payment in your upstream source system, then you can use this feature to include credit card tokens that remove sensitive details when you pay for the sales order transaction.

- Use it to help meet the Payment Card Industry Data Security Standard (PCI DSS) requirements in your order-to-cash process.
- Specify exact, credit card details in a secure way to pay for the transaction. Use the card token without handling any sensitive credit card details.
- Send a value that identifies the authorization request to your upstream source system.
- Order Management gets token details from the Payment Gateway and sends them to Oracle Payments to finish the payment.
- Order Management sends token details to Oracle Payments to validate, authorize and process the payment. You can send details about the credit card token on the order header and the order line. Use this feature to get the value that identifies the authorization request and the authorized amount from the CyberSource payment gateway.

Note

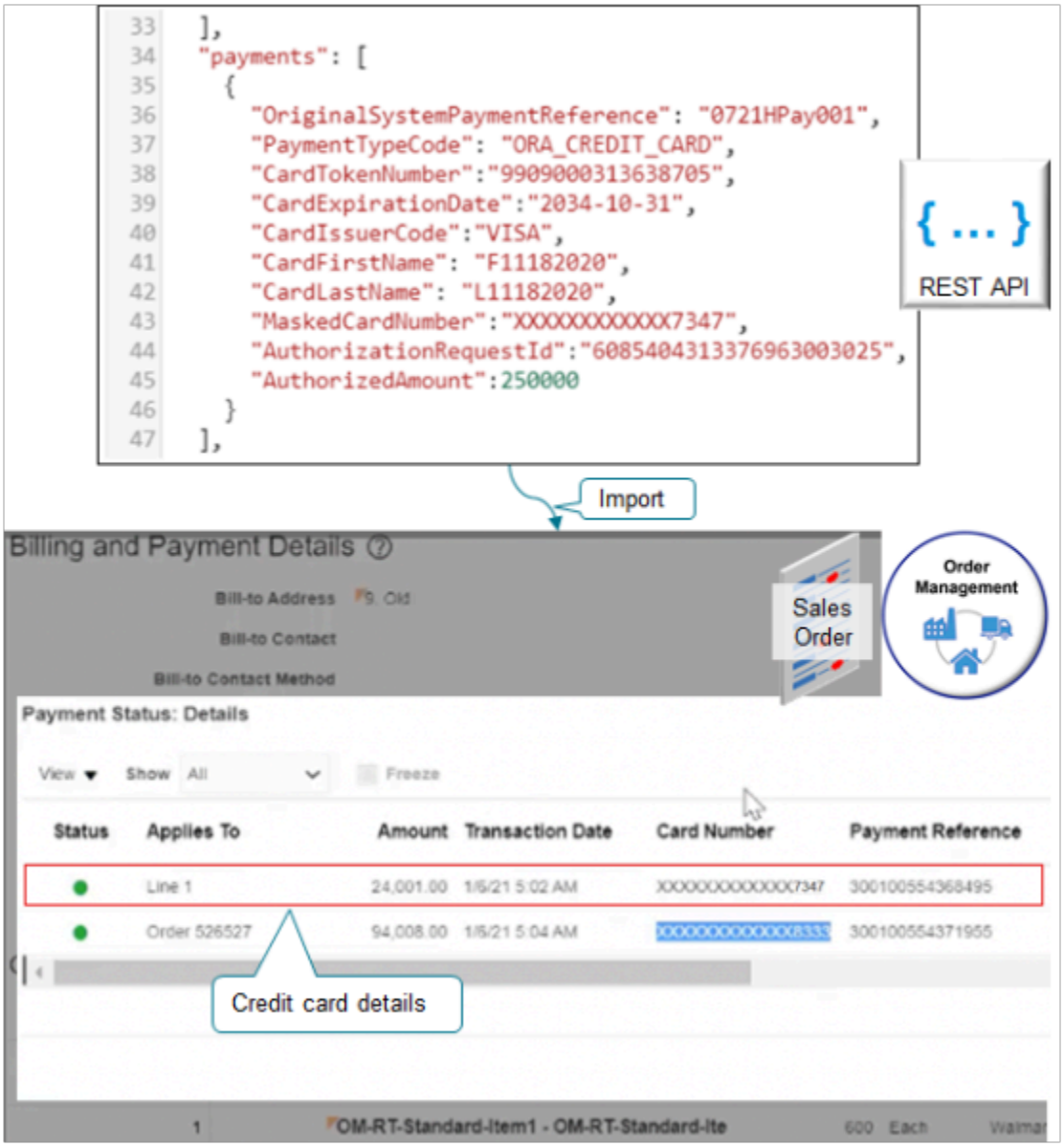
- To use this feature, you must enable the FOM_ENABLE_CARD_PAYMENT order profile. For details, see *Enable the Credit Card Feature in Order Management*.
- This feature applies only for source orders that you import through REST API.
- You can process credit card details in Oracle Applications only under controlled availability.
- Credit card processing is available only for Oracle Applications services that use Oracle Payments.
- Credit card processing is available only in data centers where Oracle Payments is certified for Payment Card Industry Data Security Standard (PCI DSS v3.2.1).
- You can use Oracle Payments only with payment gateways that can process tokens and credit card payments. For details about the certified data centers and payment gateways that you can use, see *Is Credit Card Processing Supported In Oracle Applications? (Doc ID 1949941.1)*.
- Send tokenization attributes for the credit card only when the Payment Method attribute equals Credit Card. If you don't use this method, then Order Management won't create the sales order.
- The Credit Card Token attribute is required only if your import payload includes the token attributes for the credit card.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

CAUTION: You must never send credit card numbers that aren't tokenized to Oracle Cloud Service. If they aren't tokenized, then you must modify them so they don't reveal the actual card number. For example, you can truncate the number so that you send no more than the first six digits or the last four digits of the number. You must never send the credit card data, including credit card tokens, outside the supported business flows through a file, attachment, email, descriptive flexfield, or any other attribute.

How it Works

Here's a summary of how it works.



Note

1. You import credit card details through a REST API payload.
2. Order Management calls Oracle Payments.
3. Payments communicates with CyberSource to validate and store the details that you import.
4. Payments sends the Payment Transaction Extension Identifier to Order Management.
5. To view credit card details, you go to the Order Management work area, open your sales order, go to the Billing and Payment Details tab, then use the Payment Status dialog on the order line.

REST API

Use attributes in your REST API payload.

Scenario	**Variation**	
Resource	Attribute	Description
payments	CardTokenNumber	Token number from the service that provides the token for the card number. If you import a token and an authorization, then you must include a value for CardTokenNumber.
payments	CardFirstName	First name of the card holder.
payments	CardLastName	Last name of the card holder.
payments	CardExpirationDate	Expiration date on the credit card. Provide a value in the format YYYY/MM/DD.
payments	CardIssuerCode	Abbreviation that identifies the organization that issues the card, such as Visa or MasterCard.
payments	MaskedCardNumber	Masked format that displays only the last four digits of a card number, and replaces all other digits with an X, for security purposes. The length of the value for MaskedCardNumber must match the length of the number on the card. For example, for a Visa card with number 4123456789012345, set MaskedCardNumber to XXXXXXXXXXXX2345.
payments	AuthorizationRequestId	Value that uniquely identifies the authorization request that you receive from the token service. If you don't want to use CardTokenNumber to authorize your import, then you must provide a value for AuthorizationRequestId or VoiceAuthorizationCode. If you provide a value for both of these attributes, then the import uses AuthorizationRequestId.
payments	VoiceAuthorizationCode	Abbreviation that identifies the voice authorization. If you don't want to

Scenario	Variation	
		use CardTokenNumber to authorize your import, then you must provide a value for AuthorizationRequestId or VoiceAuthorizationCode. If you provide a value for both of these attributes, then the import uses AuthorizationRequestId.
payments	PaymentServerOrderNumber	Number that identifies the card payment that Oracle Payment Server authorized.
payments	AuthorizedAmount	Amount that the token service authorized for the transaction. If you provide a value in the AuthorizationRequestId in your import payload, then you must also include a value for AuthorizedAmount.

Related Topics

- Overview of Displaying Customer Details on Sales Orders
- Extend Shipping
- Enable the Credit Card Feature in Order Management
- How Authorizations for Credit Cards Are Processed

Projects

Overview of Setting Up Projects in Order Management

Create and fulfill sales orders that include project attributes, such as Project Number, Task Number, and Expenditure Organization.

Store project details on sales orders and order lines to reduce order processing time and improve the accuracy of tracking cost, revenue, and profitability throughout your supply chain.

- Store project details on each sales order.
- Fulfill all sales order for one project from the same inventory.
- Examine the cost of services and items that you shipped for the project.
- Compare budgeted costs to actual costs.
- Create an invoice according to percent complete or milestone.
- Manage more than one project from the same set of warehouses while maintaining visibility to material and cost for each project.
- Ship each sales order for your project from project inventory or common inventory, according to your business rules.
- Create a business rule that reserves project-specific inventory first, then reserves the remaining quantity from common inventory.

- Set up orchestration so it reserves order line quantity in project-specific inventory, a mix of project-specific and common inventory, or only common inventory.
- If you ship from common inventory, then you can send the cost of shipped goods to the project during shipment.
- Use the ProjectRecordIndicator attribute to prevent Order Management from sending project-specific order lines to Receivables. Use the indicator when you set up your orchestration process and set up your assignment rule.

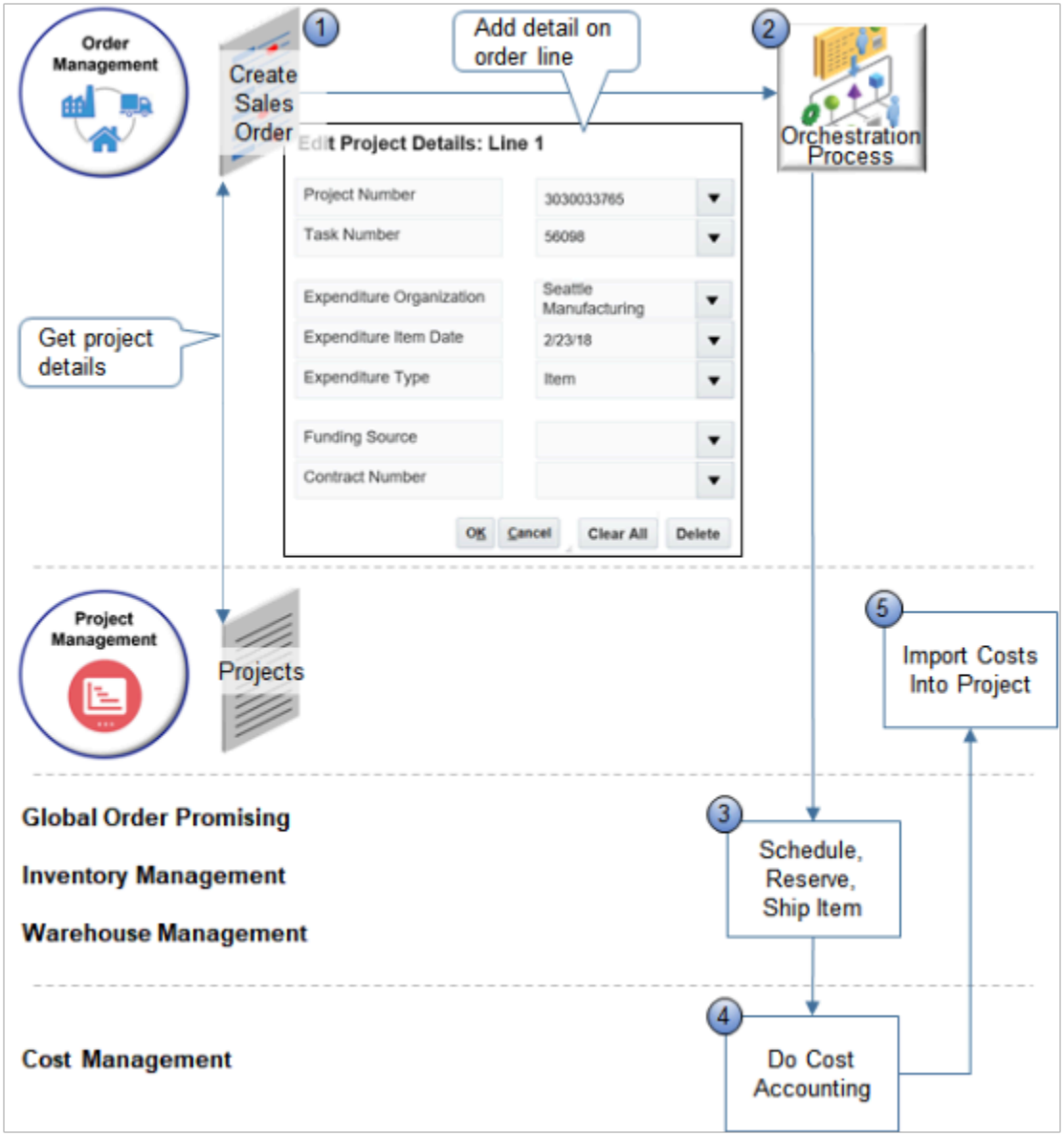
For example, Green Corp bids on a project to design and supply 100 custom batteries for an electric vehicle manufacturer. Green Corp expects design and development will require 100 hours at a cost of $50,000, and it will cost $1,000 to manufacture each unit. Green Corp adds 40% overhead and profit, and bids the job at $210,000, or $2,100 for each battery. Green Corp decides to run invoicing from the sales order when it ships the first 20 batteries, and then again when it ships the remaining 80.

Green Corp creates a project to capture cost and revenue, and then to calculate gross margin for the project.

- Set up a project with a cost budget of $150,000 and revenue budget of $210,000.
- Use the existing Quick Charge Battery item as the base item. Don't modify the original set up for this item because most items on the project are minor variances of Quick Charge Battery.
- Standard cost for Quick Charge Battery is $1,000.
- Charge 90 hours at a total of $45,000 directly to the project for design and development.
- Include $100,000 for material in the project.
- Project creates invoice for $210,000.

The project captures cost for manufacturing, design and development. It also captures revenue. For brevity, this example doesn't include shipping and other costs.

How it Works


Order Management
Create Sales Order
1
Add detail on order line
2
Orchestration Process
Edit Project Details: Line 1
Project Number
3030033765
Task Number
56098
Expenditure Organization
Seattle Manufacturing
Expenditure Item Date
2/23/18
Expenditure Type
Item
Funding Source
Contract Number
OK
Cancel
Clear All
Delete
Get project details
Project Management
Projects
5
Import Costs Into Project
Global Order Promising
Inventory Management
Warehouse Management
3
Schedule, Reserve, Ship Item
4
Cost Management
Do Cost Accounting

Note

1. Create a sales order that includes project details, such as project number, task number, and so on.

   Order Management gets the list of projects defined in Project Management, filters them according to Sales Order Business Unit and Warehouse Business Unit, then displays them as choices in the Edit Project Details dialog on the order line.

   The user clicks Submit, then Order Management validates each order line that includes project details. For example, Order Management makes sure.

   - The project that the user selects is associated with a project contract.
   - The value in the Ship-to Address attribute on the order line matches the value in the Ship-to Address for the contract.
   - The value in the Bill-to Address attribute on the order line matches the value in the Bill-to Address for the contract.

   If validation fails, Order Management displays an error and the user must update the order and submit again.

   Learn how to set up and manage projects. For details, see *Using Project Costing*.
2. An orchestration process orchestrates fulfillment for the item, including schedule, reserve, and ship.

   The ProjectRecordIndicator attribute comes predefined as set to No. If the Order Entry Specialist adds project details, then Order Management sets it to Yes, then sends it in a request to Warehouse Management. Shipping ships the order line, then uses the indicator to determine whether to add project details to the inventory transaction.
3. Fulfillment.
   - Global Order Promising promises and schedules without the context of the project, project task, or other project details, and updates order fulfillment details.

     Global Order Promising determines the warehouse for the order line, then Order Management validates the line again immediately before the orchestration process sends the line to shipping. Order Management also sets the Scheduled Ship Date to the Expenditure Item Date so it reflects the actual transaction date.
   - Inventory Management reserves the item in inventory.
   - Warehouse Management creates the shipment, picks the item from inventory, ships the item, then sends shipment confirmation to Cost Management.
4. Cost Management uses the sales order as input when it does cost accounting.
5. Project Management imports costs into the project.

If you enable Grants Management, then the Order Management work area displays the Contract Number and Funding Source attributes.

You can't.

- Include project details in a drop-ship or back-to-back flow. For details, see *Overview of Drop Ship in Order Management* and *Overview of Back-to-Back Fulfillment*.
- Send a transfer order that includes project details through Order Management.
- Use project details with sales orders that you capture through a business-to-business flow (B2B) or electronic data interchange (EDI).
- Use project details to bill an item that provides a service, such as a coverage or subscription.

- Do order promising according to project or project task.

Extend

Include project attributes in an Order Management Extension.

Order Management calls Project Management when the On Start of Submission event happens. The call validates the changes that the Order Entry Specialist makes.

For example:

```
if("PROJECT EXTN".equals(header.getAttribute("CustomerPONumber"))){
 def lines = header.getAttribute("Lines"){

 while(lines.hasNext()){
 def line = lines.next();
 if("N".equals(line.getAttribute)("ProjectRecordIndicator"))){
 def rowIter = line.getAttribute("Projects");
 def row = rowIter.createRow();
 row.setAttribute("ProjectNumber","BAT-PJCBAT-Proj-02");
 row.setAttribute("TaskId",new BigDecimal"1000000019394094");
 row.setAttribute("ExpenditureType","Cartridges");
```

```
row.setAttribute("ExpenditureOrganization","Vision Operations");
row.setAttribute("ExpenditureItemDate",java.sql.Date.valueOf("2018-07-30"));
rowIter.insertRow(row);
}
}
}
```

Note

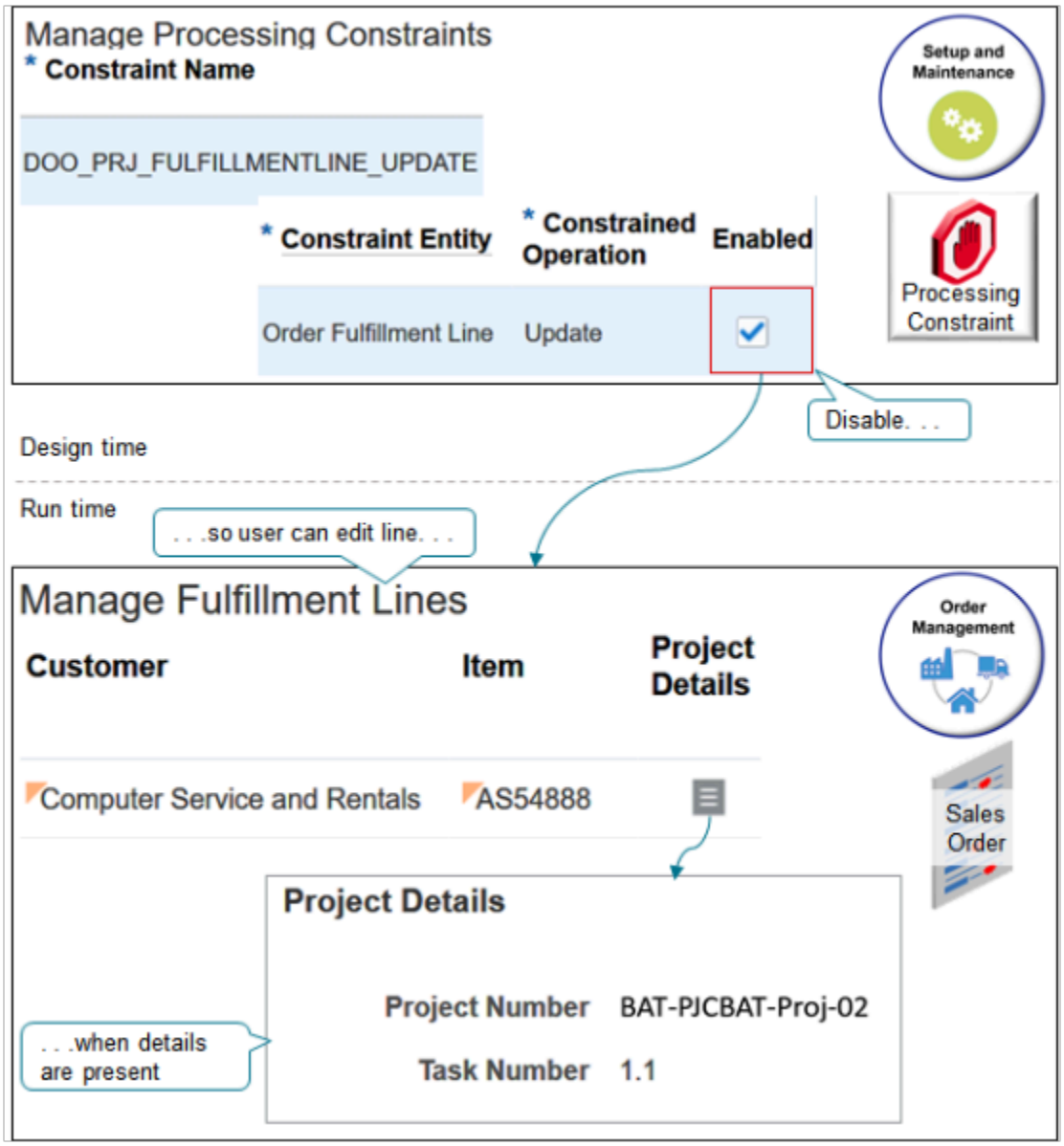
- Examine the ProjectRecordIndicator attribute to determine whether the order line contains project details.
- Set a value for each of the required project attributes.
    - ProjectNumber
    - TaskId
    - ExpenditureType
    - ExpenditureOrganization
    - ExpenditureItemDate
- Use extension event On Save, On Start of Submission, or On End of Submission to read project attributes on the order line and ProjectRecordIndicator.
- Use the On Save event or the On Start of Submission event to update project attributes on the order line.
- Use an extension to set the default value of a project attribute when the value is empty.
- Don't use.
    - On End of Submission to update project attributes.
    - Any extension event to update ProjectRecordIndicator.
    - An extension to override an existing value of a project attribute.

Constrain

The DOO_PRJ_FULFILLMENTLINE_UPDATE processing constraint comes predefined to prevent the Order Entry Specialist from updating a fulfillment line that includes project details. If you disable it, then you can update attributes that don't contain project details on the fulfillment line, such as the quantity, warehouse, and so on.

You can't update attributes that contain project details after you submit the sales order even if you disable the constraint.

Order Management applies the disable behavior only on sales orders that you submit after you disable the constraint. For details, see *Processing Constraints*.



Related Topics

- Overview of Creating Order Management Extensions
- Import Different Kinds of Data
- Different Ways of Managing Project Lifecycle
- Overview of Back-to-Back Fulfillment

Import Your Project Details

Use a web service, file, or REST API to import project details into Order Management.

For example:

- Import the project number, task number, expenditure organization, and other attributes for each project.
- Set the default value for expenditure type, user defined attributes, contract number, and funding source when you do your import.
- Include pricing in the source order that you import, or leave pricing empty in the source order and do pricing in Order Management.

Import Through Web Services

Use web service orderImportService to import a sales order that includes project details.

Here's the part of the payload where you include project details. All attributes are required.

```
<ns2:Project xmlns:pjc="http://xmlns.oracle.com/apps/flex/scm/doo/processOrder/pjcDff/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
 <ns2:PJCDFFVA xsi:type="pjc:OmSalesOrder">
 <pjc:projectId_Display>PDSC-CT-PDSCM1X-004</pjc:projectId_Display>
 <pjc:taskId_Display>1.1.1</pjc:taskId_Display>
 <pjc:expenditureItemDate>2019-03-20</pjc:expenditureItemDate>
 <pjc:expenditureTypeId_Display>Material</pjc:expenditureTypeId_Display>
 <pjc:organizationId_Display>Vision Operations</pjc:organizationId_Display>
 </ns2:PJCDFFVA>
</ns2:Project>
```

where

Attribute or Code	Value
pjcDff	Indicates that you're using a descriptive flexfield to store project details. You must use this value.
xsi:type	Identifies the source type as OmSalesOrder, which means Order Management sales order. You must use this value.
projectId_Display	Value that uniquely identifies the project. This example uses 1.1.1 as the value.
expenditureItemDate	Date to expense the item. You must use the Year-Month-Day format. For example, 2019-03-20 indicates March 20, 2019.
expenditureTypeId_Display	Identifies the type of expenditure, such as Material.
organizationId_Display	Identifies the business unit, such as Vision Operations.
Reservation ID	The example payload doesn't include this attribute. The web service uses the attribute only for earlier updates. For the current update, the value is empty, and the web service doesn't use it.

`_Display` indicates to use the value when we display the attribute in an Oracle Application.

Here's an example payload that contains the same project details.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:Order>
 <ns2:SourceTransactionIdentifier>RRFPDSC_TEST_112</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionSystem>LEG</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionNumber>RRFPDSC_TEST_112</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyName>FOM-Customer-001</ns2:BuyingPartyName>
 <ns2:BuyingPartyContactName>James Pattison</ns2:BuyingPartyContactName>
 <ns2:CustomerPONumber>PDSCSPBU</ns2:CustomerPONumber>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2019-08-22T00:00:00.000</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitName>Vision Operations</ns2:RequestingBusinessUnitName>
 <ns2:OrigSystemDocumentReference/>
 <ns2:TransactionTypeCode>STD</ns2:TransactionTypeCode>
 <ns2:FreezePriceFlag>false</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>false</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>false</ns2:FreezeTaxFlag>
 <ns2:ShipToPartyName>FOM-Customer-001</ns2:ShipToPartyName>
 <ns2:ShipToAddress1>1045, 5th Avenue</ns2:ShipToAddress1>
 <ns2:ShipToCity>San Diego Country Estate</ns2:ShipToCity>
 <ns2:ShipToPostalCode>92065</ns2:ShipToPostalCode>
 <ns2:ShipToState>CA</ns2:ShipToState>
 <ns2:ShipToCountry>US</ns2:ShipToCountry>
 <ns2:ShipToPartyContactName>James Pattison</ns2:ShipToPartyContactName>
 <ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
 <ns2:BillToCustomerName>FOM-Customer-001</ns2:BillToCustomerName>
 <ns2:BillToCustomerIdentifier>300100046859202</ns2:BillToCustomerIdentifier>
 <ns2:BillToAddress1>1045, 5th Avenue</ns2:BillToAddress1>
 <ns2:BillToCity>San Diego Country Estate</ns2:BillToCity>
 <ns2:BillToPostalCode>92065</ns2:BillToPostalCode>
 <ns2:BillToState>CA</ns2:BillToState>
 <ns2:BillToCountry>US</ns2:BillToCountry>
 <ns2:BillToAccountContactName>James Pattison</ns2:BillToAccountContactName>
 <ns2:SalesChannel>Direct</ns2:SalesChannel>
 <ns2:Salesperson>Paul Robert Scholes</ns2:Salesperson>
 <ns2:SalesCredit>
 <ns2:SourceTransactionSalesCreditIdentifier>OSC-001</ns2:SourceTransactionSalesCreditIdentifier>
 <ns2:SalesPerson>Paul Robert Scholes</ns2:SalesPerson>
 <ns2:Percent>100</ns2:Percent>
 <ns2:SalesCreditTypeCode>1</ns2:SalesCreditTypeCode>
 <ns2:SalesCreditTypeReference>Quota Sales Credit</ns2:SalesCreditTypeReference>
 </ns2:SalesCredit>
 <ns2:Line>
 <ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
 <ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
 <ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
 <ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
 <ns2:ParentLineReference/>
 <ns2:RootParentLineReference/>
 <ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
 <ns2:ProductNumber>OM-PDSC-STD-03-C</ns2:ProductNumber>
 <ns2:OrderedQuantity>100</ns2:OrderedQuantity>
 <ns2:OrderedUOM>Each</ns2:OrderedUOM>
 <ns2:RequestedFulfillmentOrganizationCode>PDSCM1</ns2:RequestedFulfillmentOrganizationCode>
 <ns2:CustomerPONumber>NKPO2017/10/005/01</ns2:CustomerPONumber>
 <ns2:CustomerPOLineNumber>1</ns2:CustomerPOLineNumber>
 <ns2:RequestedShipDate>2019-06-30T23:08:52Z</ns2:RequestedShipDate>
 <ns2:PaymentTerms>30 Net</ns2:PaymentTerms>
 <ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
```

```
 <ns2:Comments/>
 <ns2:TaxExempt>S</ns2:TaxExempt>
 <ns2:ShipToPartyName>FOM-Customer-001</ns2:ShipToPartyName>
 <ns2:ShipToAddress1>1045, 5th Avenue</ns2:ShipToAddress1>
 <ns2:ShipToCity>San Diego Country Estate</ns2:ShipToCity>
 <ns2:ShipToPostalCode>92065</ns2:ShipToPostalCode>
 <ns2:ShipToState>CA</ns2:ShipToState>
 <ns2:ShipToCountry>US</ns2:ShipToCountry>
 <ns2:ShipToPartyContactName>James Pattison</ns2:ShipToPartyContactName>
 <ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
 <ns2:BillToCustomerName>FOM-Customer-001</ns2:BillToCustomerName>
 <ns2:BillToCustomerIdentifier>300100046859202</ns2:BillToCustomerIdentifier>
 <ns2:BillToAddress1>1045, 5th Avenue</ns2:BillToAddress1>
 <ns2:BillToCity>San Diego Country Estate</ns2:BillToCity>
 <ns2:BillToPostalCode>92065</ns2:BillToPostalCode>
 <ns2:BillToState>CA</ns2:BillToState>
 <ns2:BillToCountry>US</ns2:BillToCountry>
 <ns2:BillToAccountContactName>James Pattison</ns2:BillToAccountContactName>
 <ns2:Project xmlns:pjc="http://xmlns.oracle.com/apps/flex/scm/doo/processOrder/pjcDff/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
 <ns2:PJCDFFVA xsi:type="pjc:OmSalesOrder">
 <pjc:projectId_Display>PDSC-CT-PDSCM1X-004</pjc:projectId_Display>
 <pjc:taskId_Display>1.1.1</pjc:taskId_Display>
 <pjc:expenditureItemDate>2019-03-20</pjc:expenditureItemDate>
 <pjc:expenditureTypeId_Display>Material</pjc:expenditureTypeId_Display>
 <pjc:organizationId_Display>Vision Operations</pjc:organizationId_Display>
 </ns2:PJCDFFVA>
 </ns2:Project>
 <ns2:AdditionalFulfillmentLineInformationCategories
 xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate" xmlns:ns12="http://xmlns.oracle.com/
apps/scm/doo/processOrder/flex/fulfillLineCategories/" xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/
processOrder/flex/fulfillLineContextsB/" xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/
model/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <ns12:FulfillLineEffBShipment__InstructionsprivateVO>
 <ns8:ContextCode>Shipment Instructions</ns8:ContextCode>
 <ns22:packInstructions>OM-PI-Test</ns22:packInstructions>
 <ns22:shipInstructions>OM-SI-Test</ns22:shipInstructions>
 <ns22:handlingCost>12.5</ns22:handlingCost>
 <ns22:needByDate>2018-10-15</ns22:needByDate>
 <ns22:expectedPickDate>2018-10-15T12:12:12</ns22:expectedPickDate>
 </ns12:FulfillLineEffBShipment__InstructionsprivateVO>
 </ns2:AdditionalFulfillmentLineInformationCategories>
 </ns2:Line>
 <ns2:Line>
 </ns2:Order>
 </ns1:request>
 </ns1:createOrders>
 </soap:Body>
</soap:Envelope>
```

For brevity, this payload contains only one order line. To see a more complete payload, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Payloads and Files attachment.

Validate Project Details in Fulfillment Lines

Use web service orderImportService to validate the combination of project attributes on a fulfillment line. Do it right after the user submits a new order or revises an existing one, or during a shipment task.

The web service validates the business unit for the selling profit center and the warehouse during the On Submission event.

For example, the web service makes sure.

- The user provides values for each required attribute in the Edit Project Details dialog.
  - Project Number
  - Task Number
  - Expenditure Organization
  - Expenditure Item Date
  - Expenditure Type
- The flow has assigned a contract to the project.
- The expenditure item date that the user set happens within the project start date and finish date.
- The bill-to address on the sales order matches the bill-to address of the contract.
- The ship-to address on the sales order matches the ship-to address of the contract.

If validation fails, then Order Management displays a warning or error and a suggestion of how to fix it.

Order Management does these validations to prevent problems from occurring during shipping or when it invoices the contract. For example, if the ship-to location on the sales order doesn't match the ship-to location on at least one of the contract lines, then the sales order might use an incorrect tax when Order Management invoices the contract for the project. Shipping might also ship the item to an incorrect location. The user would need to change the fulfillment line and manually recover the order so Order Management can send the corrected lines to shipping.

Here's an example response that the web service sends when the expenditure item date happens outside the project dates.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders
<ns0: createOrdersResponse xmlns:ns0="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/
types/">
 <ns1:result xsi:type="ns0:OrderImportResponse xmlns:ns1=
"http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns0:Order>
 <ns0:SourceTransactionNumber>RRFPDSC_TEST_112</ns0:SourceTransactionNumber>
 <ns0:SourceTransactionIdentifier>RRFPDSC_TEST_112</ns0:SourceTransactionIdentifier>
 <ns0:SourceTransactionSystem>LEG</ns0:SourceTransactionSystem>
 <ns0:OrderNumber>514626</ns0:OrderNumber>
 <ns0:HeaderId>xsi:nil="true"</ns0:HeaderId>
 <ns0:OrderStatus>DOO_DRAFT</ns0:OrderStatus>
 <ns0:ReturnStatus>ERROR</ns0:ReturnStatus>
 <ns0:MessageName>env:Server</ns0:MessageName>
 <ns0:MessageText>The submit failed for sales order RRFPDSC_TEST_112 on order line1, in schedule 1. The
 expenditure item date happens outside the project dates. Change the project start and finish dates, or
 change the expenditure item date.</ns0:MessageText>
 <ns0:flowType>xsi_nil="true"</ns0:flowType>
```

Import Through Files

Use the DOO_PROJECTS_INT worksheet in the Order Import Template to import your project details.

- Import a batch. Import sales order lines that include project details and lines that don't in the same batch.
- Import project details for shippable items.
- Import configured items.
- Validate the combination of project attributes as part of the order submission process.

- Use the DOO_PROJECTS_INT worksheet in the template to capture project attributes on the sales order line.

Here are the required attributes.

Attribute	Value
Source Transaction Identifier	Enter any alphanumeric text, such as FBDI_SRK_0318_08.
Source Transaction System	Enter any alphanumeric text, such as GPR.
Source Transaction Line Identifier	Identify the order line in the source transaction, such as 1.
Source Transaction Schedule Identifier	Identify the schedule in the source transaction, such as 1.
Project Number, or Project Name, or Project ID	Identify the project. Include a value for at least one of these attributes, such as 300100113461421 for Project ID.
Task Number, Task Name, Task ID	Identify the task. Include a value for at least one of these attributes, such as 1.1 for Task Number.
Expenditure Item Date	Date to expense the item. You must use the Year-Month-Day format. For example, 2019-03-20 indicates March 20, 2019.
Expenditure Type or Expenditure Type ID	Identify the type of expenditure. Include a value for at least one of these attributes, such as Material for Expenditure Type.
Expenditure Organization or Expenditure Organization ID	Identify the business unit, such as Vision Operations for Expenditure Organization.
Contract Number or Contract ID	If you enable grants for the project, then identify the contract. Include a value for at least one of these attributes.
Funding Source Number or Funding Source Id	If you enable grants for the project, then identify the funding source. Include a value for at least one of these attributes.

Include values for optional attributes, as necessary.

- BillableFlag
- CapitalizableFlag
- ContractLineId
- WorkType

- WorkTypeId
- FundingAllocationId
- UserDefinedAttribute 1 through UserDefinedAttribute n
- ReservedAttribute 2 through ReservedAttribute n

For some templates that include example values, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Payloads and Files attachment.

For details, see *Overview of Importing Orders Into Order Management*. If you need help using the template to import project details, contact Oracle Support.

Integrate Through REST API

Use project resources in the Sales Orders for Order Hub REST API.

Resource	Description
Project	Includes the DooOrderPrjId attribute. It identifies the project, such as 300100010341182.
Project Detail.	Includes other project attributes.

Here are attributes from an example payload that uses the project detail resource.

```
"DooOrderPrjId": 300100181049087,
"__FLEX_Context": "OM_Sales_Order",
"__FLEX_Context_DisplayValue": "SCM: General",
"projectId": 300100010341182,
"projectId_Display": "Projects-TL-Int-01",
"taskId": 300100010341193,
"taskId_Display": "1.1",
"expenditureItemDate": null,
"expenditureTypeId": 10028,
"expenditureTypeId_Display": "Material",
"organizationId": 204,
"organizationId_Display": "Vision Operations",
"contractId": null,
"contractId_Display": null,
"reservedAttribute1": null,
"reservedAttribute1_Display": null,
"billableFlag": null,
"billableFlag_Display": null,
"capitalizableFlag": null,
"capitalizableFlag_Display": null,
"workTypeId": null,
"workTypeId_Display": null,
```

Here's an example payload that creates a sales order for item AS92888 in a project named Projects-TL-Int-01. It includes display attributes that use values to identify the project, such as projectId_Display.

```
URL: https://fuscdrmsmc347-fa-ext.myCompany.com:443/fscmRestApi/resources/11.13.20.01/salesOrdersForOrderHub
METHOD: POST
{
 "SourceTransactionNumber": "R13_project_valueattrs_01",
 "SourceTransactionSystem": "GPR",
 "SourceTransactionId": "R13_project_valueattrs_01",
 "BusinessUnitId": 204,
 "BuyingPartyId": 1006,
```

```
 "BuyingPartyContactId": 2663,
 "TransactionalCurrencyName": "US Dollar",
 "RequestedShipDate": "2019-01-20T20:51:21+00:00",
 "PartialShipAllowedFlag": false,
 "RequestingBusinessUnitId": 204,
 "RequestingLegalEntityId": 204,
 "FreezePriceFlag": "N",
 "FreezeTaxFlag": "N",
 "RequestedFulfillmentOrganizationId":207,
 "PaymentTerms": "30 Net",
 "SubmittedFlag": true,
 "billToCustomer": [
 {
 "CustomerAccountId": 1006,
 "SiteUseId": 1025,
 "ContactFirstName": "Sarah",
 "ContactLastName": "Takesh"
 }
 ],
 "shipToCustomer": [
 {
 "PartyId": 1006,
 "SiteId": 1036
 }
 ],
 "lines": [
 {
 "SourceTransactionLineId": "1",
 "SourceTransactionLineNumber": "1",
 "SourceScheduleNumber": "1",
 "SourceTransactionScheduleId": "1",
 "ProductNumber": "AS92888",
 "OrderedUOMCode": "Ea",
 "PurchasingUOMCode": "Ea",
 "OrderedQuantity": 1,
 "project": [
 {
 "projectDetail": [
 {
 "projectId_Display": "Projects-TL-Int-01",
 "taskId_Display": 1.1,
 "expenditureTypeId_Display": "Material",
 "organizationId_Display": "Vision Operations",
 "contractId": null,
 "reservedAttribute1": null,
 "billableFlag": null,
 "capitalizableFlag": null,
 "workTypeId": null
 }
 ]
 }
 ]
 }
 ]
}
```

Here's another example that creates a sales order for item AS92888 in project 300100010341182. It includes attributes that use an identifier to identify the project, such as projectId.

```
URL: https://fuscdrmsmc347-fa-ext.myCompany.com:443/fscmRestApi/resources/11.13.20.01/salesOrdersForOrderHub
METHOD: POST

{
 "SourceTransactionNumber": "R13_project_Idattrs_01",
 "SourceTransactionSystem": "GPR",
 "SourceTransactionId": "R13_project_Idattrs_01",
```

```
"BusinessUnitId": 204,
"BuyingPartyId": 1006,
"BuyingPartyContactId": 2663,
"TransactionalCurrencyName": "US Dollar",
"RequestedShipDate": "2019-01-20T20:51:21+00:00",
"PartialShipAllowedFlag": false,
"RequestingBusinessUnitId": 204,
"RequestingLegalEntityId": 204,
"FreezePriceFlag": "N",
"FreezeTaxFlag": "N",
"RequestedFulfillmentOrganizationId":207,
"PaymentTerms": "30 Net",
"SubmittedFlag": true,
"billToCustomer": [
{
"CustomerAccountId": 1006,
"SiteUseId": 1025,
"ContactFirstName": "Sarah",
"ContactLastName": "Takesh"
}
],
"shipToCustomer": [
{
"PartyId": 1006,
"SiteId": 1036
}
],
"lines": [
{
"SourceTransactionLineId": "1",
"SourceTransactionLineNumber": "1",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"ProductNumber": "AS92888",
"OrderedUOMCode": "Ea",
"PurchasingUOMCode": "Ea",
"OrderedQuantity": 1,
"project": [
{
"projectDetail": [
{
"projectId": 300100010341182,
"taskId": 300100010341193,
"expenditureTypeId": 10028,
"organizationId": 204,
"expenditureItemDate": "2019-09-24",
"contractId": null,
"reservedAttribute1": null,
"billableFlag": null,
"capitalizableFlag": null,
"workTypeId": null
}
]
}
]
}
]
}
```

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Sales Orders for Order Hub**.

Related Topics

- Import Different Kinds of Data

Set Up Projects in Order Management

Set up Order Management so the Order Entry Specialist can include project details on a sales order.

This topic describes how to set up a few project features that are specific to Order Management. Learn how to set up projects so they work across your entire supply chain. For details, see *Overview of Project-Driven Supply Chain Management*.

Modify Invoicing Behavior

Project Contract Billing typically invoices sales order lines that include project details, so you must prevent Order Management from sending order lines that include project details to Receivables. This section describes how to remove steps that do invoice tasks from your orchestration process. As an alternative, you can also do one of these set ups.

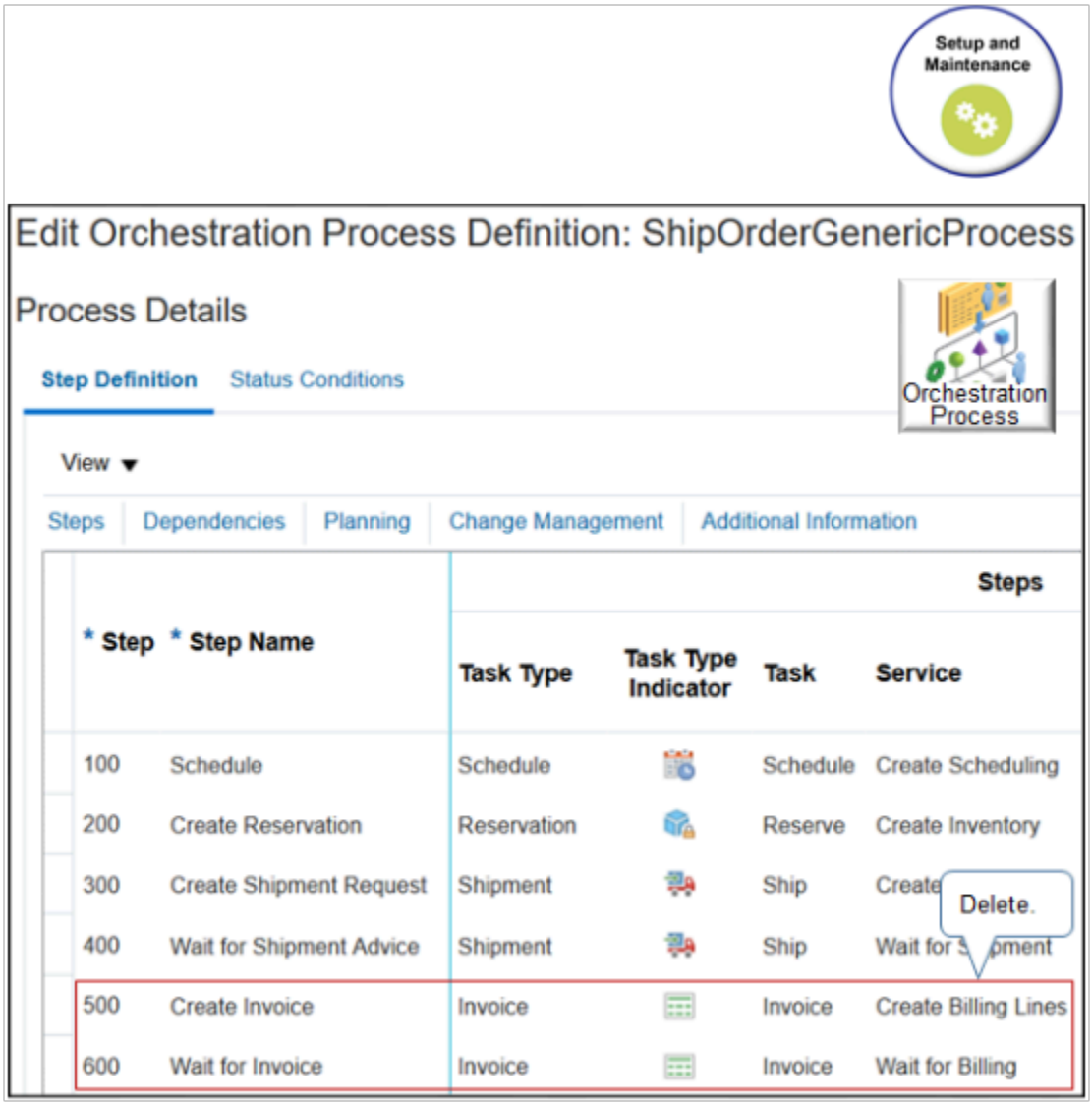
- Create a new line type, such as Project.

  Create an orchestration process that doesn't include an invoice task, then assign the process to the Project line type.
- Create an orchestration process that does include an invoice task and that uses line selection criteria to skip the lines that include the Project line type.

In this example, assume you must modify the ShipOrderGenericProcess orchestration process, and that you already released this process into production.

1. Enable the Process Sales Orders for Projects opt-in feature. For details, see *Opt Into Features in Order Management*.

2. Modify the orchestration process.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Orchestration Process Definitions
    - On the Manage Orchestration Process Definitions page, search for ShipOrderGenericProcess, then click **Actions > Edit**.
    - On the Edit Orchestration Process Definitions page, click **Actions > Revise Process**.
    - In the Process Details area, delete the Create Invoice step and the Wait for Invoice step.



    - Deploy your orchestration process, then click **Save and Close**.
    - On the Manage Orchestration Process Definitions page, click **Cancel**.

3. Create a lookup.
    - Go to the Setup and Maintenance work area, then go to task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Order Lookups
    - On the Manage Order Lookups page, create a new lookup.

Attribute	Value
Name	CUSTOM_ORA_DOO_LINE_TYPES
Meaning	Line Type for Project Details
Description	Allow Order Entry Specialist to specify whether order line contains project details.
Module	Orchestration

    - In the Lookup Codes area, create a code.

Attribute	Value
Lookup Code	PRJ
Meaning	Contains Project Details
Sequence	1

    - Create another one, then click **Save and Close**.

Attribute	Value
Lookup Code	NO_PRJ
Meaning	Doesn't Contain Project Details
Sequence	2

Related Topics

- Opt Into Features in Order Management
- Manage Lookups in Order Management

Dual Units of Measure

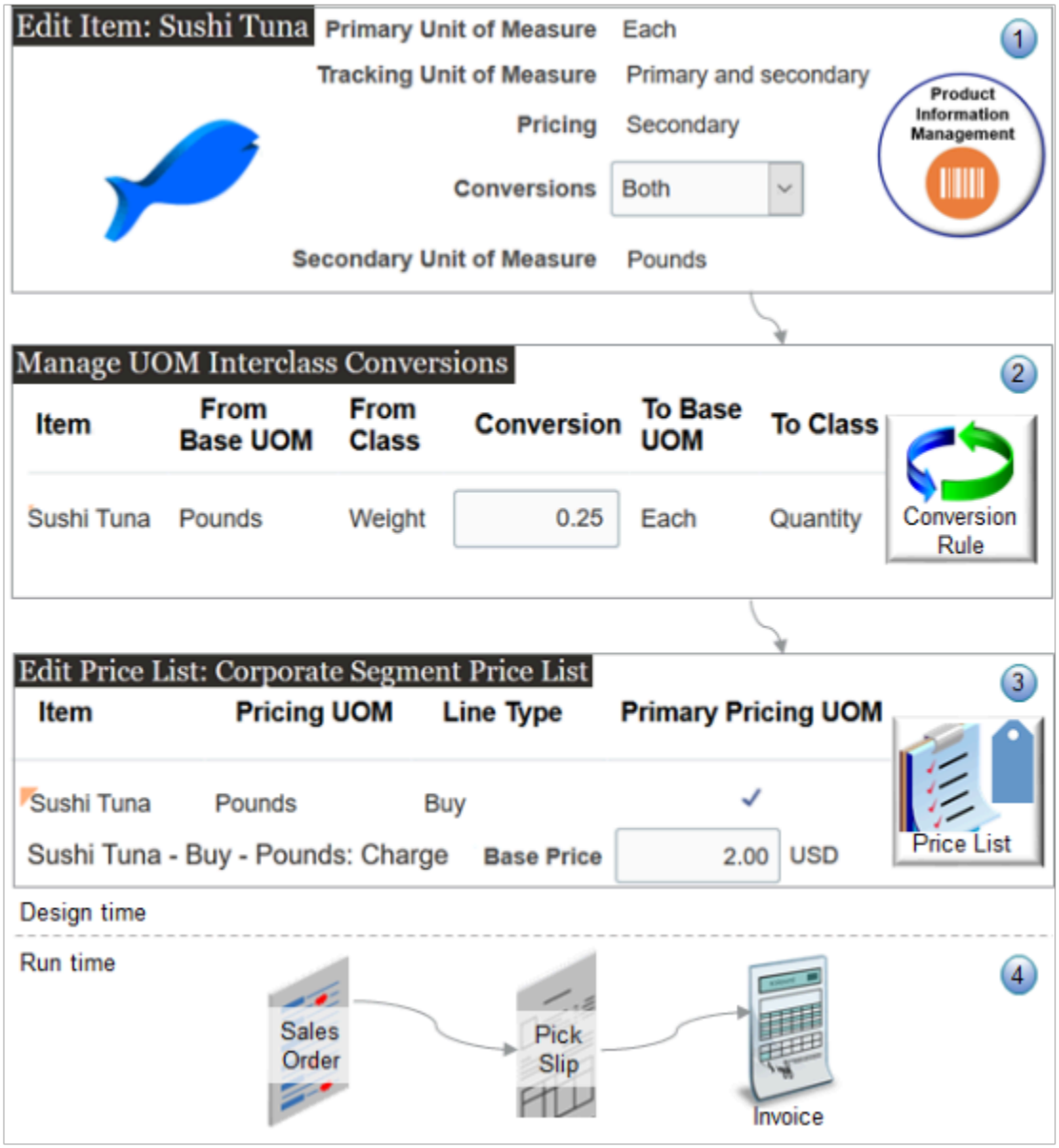
Overview of Setting Up Dual Units of Measure

Get an overview of how to set up dual units of measure so you can order, price, manufacture, receive, pick, pack, and ship an item in the primary UOM but price it in the secondary UOM.

For details, see *Track Items in More Than One Unit of Measure*.

How You Set it Up

Assume you sell an item named Sushi Tuna. You price it by pound, not by each fish, so you need to price it in the secondary unit of measure. Sushi Tuna is a good candidate for dual units of measure because a tuna naturally varies in size, you typically sell each tuna as one fish each, but you price it by weight. You sell each tuna fish for a different amount because each one has a unique weight. You need to track the item in the primary and secondary UOM throughout the order fulfillment lifecycle.



What the Numbers Mean

1. You specify details for the unit of measure when you set up the item in the Product Information Management work area. In this example, you set these attributes.

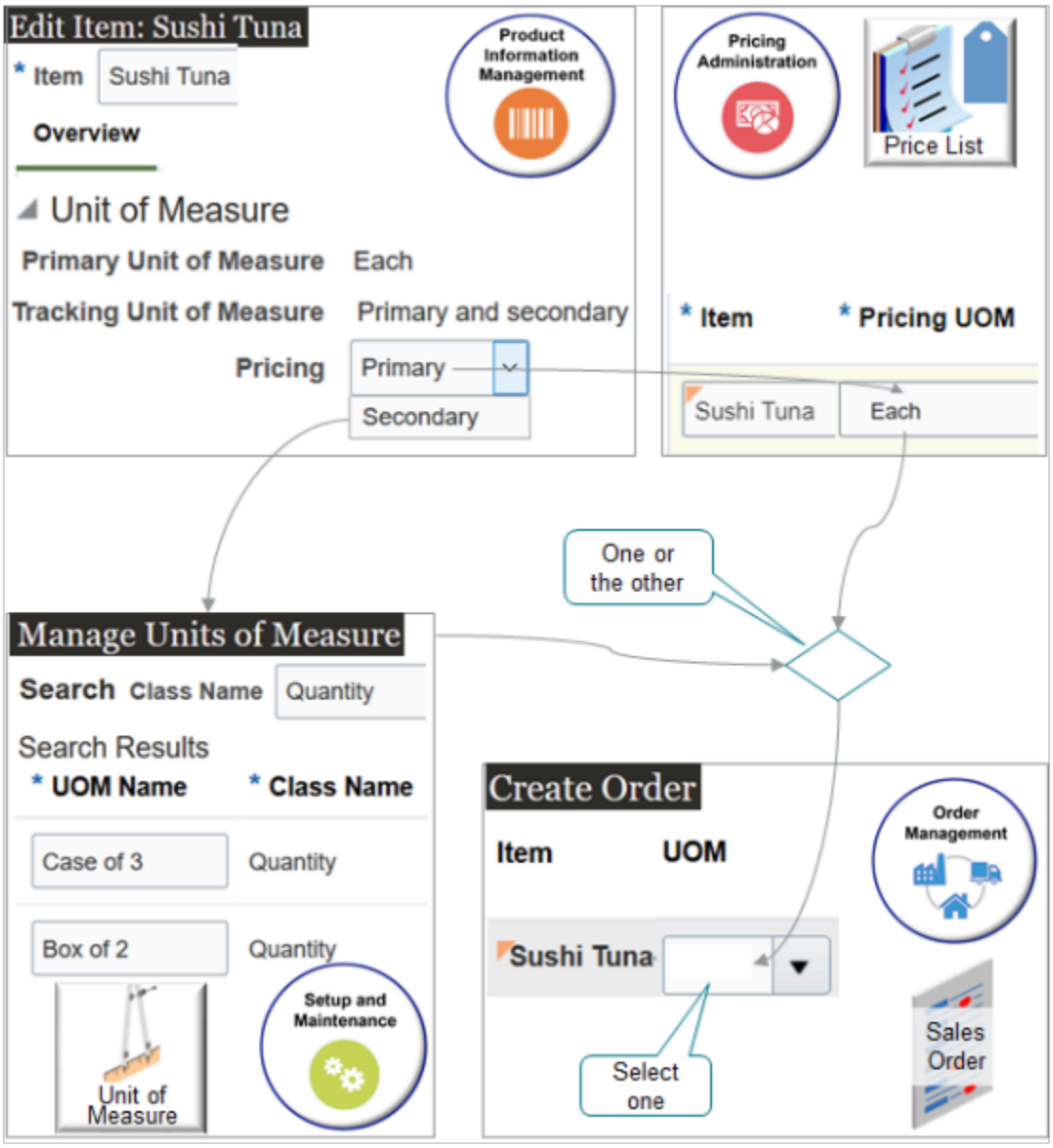
Attribute	Value
Primary Unit of Measure	Each Specify the unit of measure that you use to stock and sell and the item.
Tracking Unit of Measure	Primary and Secondary

Attribute	Value
	Specify how to track your on-hand balance. If you set it to Primary and Secondary, then you're telling Order Management that this item is a dual UOM item.
Pricing	Secondary Specify whether to use the Primary or the Secondary unit of measure to price the item. If you use Secondary, then pricing reprices the order line when you confirm the shipment.
Secondary Unit of Measure	Pound Specifies the unit of measure that you use to price the item.

2. You use the Setup and Maintenance work area to create a conversion rule between the primary UOM and the secondary UOM. For example, convert 0.25 of the secondary UOM Pounds to 1 of the primary UOM Each. This tells Order Management how to convert from the primary to the secondary when it prices your tuna.
3. You use the Pricing Administration work area to add the Sushi Tuna item to a price list, such as the Corporate Segment Price List. In pricing, you specify the Pounds UOM as the primary. This way, pricing will price it by the pound, not by each. You also specify the base price, such as 2.00 USD. So, you're charging $2 for each pound of tuna.
4. Here's what happens at runtime.
    - You use the Order Management work area to create a sales order, add an item that uses dual units of measure, then submit the order.
    - Order Management creates and sends a shipment request to Oracle Shipping.
    - Oracle Shipping creates a shipment and populates the requested quantity and the secondary requested quantity according to your conversion rule.
    - You use the Inventory Management work area to create a pick wave, confirm the pick slip, confirm the shipment, then Inventory Management sends the confirmation to Order Management.
    - Order Management sends the order line to Oracle Receivables.
    - You can use a fulfillment view in the Order Management work area to see that the fulfillment line status is Shipped, and also to examine invoice details.

Product Information Management

The value that you can set for the ordered UOM depends on how you set the Pricing attribute in Product Information Management.



If You Set the Pricing attribute in Product Information Management To	Then You Can Set the Ordered UOM To
Primary	The unit of measure that you specify when you set up pricing for the item in the Pricing Administration work area. For example, if you set the Pricing UOM attribute in the Pricing Administration to Each, then you must set the ordered UOM to Each on the order line.
Secondary	Any unit of measure that's in the same class of the UOM that you specify in the Primary Unit of Measure attribute in Product Information Management. For example, if you set the Primary Unit of Measure attribute in Product Information Management to Each, then you can set the ordered UOM to any UOM that's in the Quantity class because Each is in the

If You Set the Pricing attribute in Product Information Management To	Then You Can Set the Ordered UOM To
	Quantity class. Example measures in this class might include Case of 3, Box 2, Dozen, Each, Gross, and so on. To determine what units of measure are in a class, go to the Setup and Maintenance work area, open the Manage Units of Measure task, then search according to class, such as the Quantity class.

Pricing

Use the Pricing Administration work area to specify pricing for the item.

Note

- You can specify pricing for a dual UOM item in a price list, discount list, or shipping charge list.

- Use the Pricing UOM attribute to tell Pricing which UOM to use to price the item. Use this attribute when you add the item to a list, such as the Corporate Segment Price List.
- Set up pricing in any unit of measure that's in the primary's or the secondary's unit of measure class.
- Use the Manage Units of Measure page in the Setup and Maintenance work area to add or remove units of measure from a class.
- Manage pricing data in the Pricing Administration work area.
- Import pricing data through REST API, file-based data import, or a web service.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Pricing Example

See what the attribute values are as your item moves through fulfillment.

1. Starting on the order line in Order Management.
2. As it moves to the shipment line in Oracle Shipping.
3. On the fulfillment line back in Order Management after Order Management receives the shipment confirmation from Shipping.
4. To the invoice line in Oracle Accounts Receivable.

Assume you sell an item named Sushi Tuna. You sell the whole fish, and you sell it for the same price regardless of how big the fish is, so you use the Each primary UOM to price it. Here's what pricing looks like when you sell a quantity of 2 fish and each one weighs 4 pounds.

Line	Ordered Quantity	Ordered UOM	Shipped Quantity	Secondary Quantity	Secondary UOM	Secondary Shipped Quantity	Secondary Fulfilled Quantity	Unit Price	Extended Amount
Order Line	2	Each	0	8	Pounds	-	-	$4	$8
Shipping Line	2	Each	2	8	Pounds	-	-	-	-
Fulfillment Line	2	Each	2	8	Pounds	8	-	-	-
Invoice Line	2	Each	-	8	Pounds	8	8	$4	$8

Each fish can vary a great deal in weight, so using Each results in pricing that's all over the map. Your customers prefer to buy the whole fish, so you decide to continue selling each whole fish instead of cutting it up, but you use the Pounds secondary UOM to price it. Here's what pricing looks like when you sell a quantity of 2 fish, each one weighs 4 pounds, and you charge $1.50 a pound.

Line	Ordered Quantity	Ordered UOM	Shipped Quantity	Secondary Quantity	Secondary UOM	Secondary Shipped Quantity	Secondary Fulfilled Quantity	Unit Price	Extended Amount
Order Line	2	Each	0	8	Pounds	-	-	$1.5	$6

Line	Ordered Quantity	Ordered UOM	Shipped Quantity	Secondary Quantity	Secondary UOM	Secondary Shipped Quantity	Secondary Fulfilled Quantity	Unit Price	Extended Amount
Shipping Line	2	Each	2	8	Pounds	-	-	-	-
Fulfillment Line	2	Each	2	8	Pounds	7	-	-	-
Invoice Line	2	Each	-	8	Pounds	7	7	$1.5	$10.5

Flows That This Feature Supports

Use more than one unit of measure in your business process flows.

Order to Cash

- Standard order.
- Back-to-back flow. Use this feature to help you create the purchase order, transfer order, or work order when you use dual units of measure to fulfill demand in your back-to-back flow.
- Drop shipment. Create a sales order in the item's primary unit of measure (UOM), then use the item's secondary quantity during fulfillment in your drop ship flow. If you want to use drop ship, then we recommend that you opt into the Purchase Items That are Priced in Their Secondary UOM feature in the Procurement offering.
- Transfer order. Create a transfer order in the primary unit of measure and automatically calculate the expected quantity in the secondary UOM according to a UOM conversion. Specify transfer pricing in the secondary UOM according to the actual secondary quantity that you receive.
- Return order.

If a change of ownership occurs between one business unit to another business unit, then Oracle Supply Chain Financial Orchestration orchestrates financial details and processes the transaction.

Procure to Pay

- Purchasing. Create a purchase order in the primary unit of measure (UOM) and automatically calculate the expected quantity in the secondary UOM according to a UOM conversion. Price the order line in the secondary UOM and invoice it according to the actual secondary quantity that you receive.
- Self Service Procurement.
- Direct Procurement.
- Invoicing in Accounts Payable.
- Advance Shipment Notice.
- Pay on Receipt.
- Debit Memo.

Plan to Produce

- Discrete Manufacturing.
- Process Manufacturing, including work definitions and work orders.

- Contract Manufacturing. Include dual units of measure in the production reports that you get from your contract manufacturers. Include a description of the materials used and what was finished. Include details in the primary and in the secondary unit of measure.
- Outside Processing.

Drop Shipments and Returns

Drop Shipments

Set up your Order Management implementation so your supplier can drop ship each of your sales orders to your customer and invoice it according to the primary quantity or the secondary quantity that your supplier ships on the order line.

As an option, create a blanket purchase agreement (BPA) to procure the item from your supplier at a predefined price, then price it according to the secondary unit of measure.

- Set the Supplier attribute on the Create Agreement dialog when you create the agreement.
- If the blanket purchase agreement exists, then Oracle Purchasing automatically creates a purchase order for the sales order, and it sets the quantity and unit of measure on the Blanket Purchase Agreement page according to the primary unit of measure of the item regardless of the unit of measure that you set on the order line.

You can capture the actual quantity that your fulfillment system ships in the secondary unit of measure, and Order Management will automatically adjust the extended amount on the order line. For example, if you drop ship the item, then you can use the Advance Shipment Notice (ASN) from your supplier to determine the quantity that your supplier shipped in the secondary unit of measure, then automatically adjust the extended amount on the order line.

Note

- You can't use AP Invoice with dual units of measure. You must use an Advance Shipment Notice as the event that starts the transfer of ownership.
- If you use AP Invoice, and if the flow creates the invoice, then you can't revise or cancel the order line.

Referenced Returns

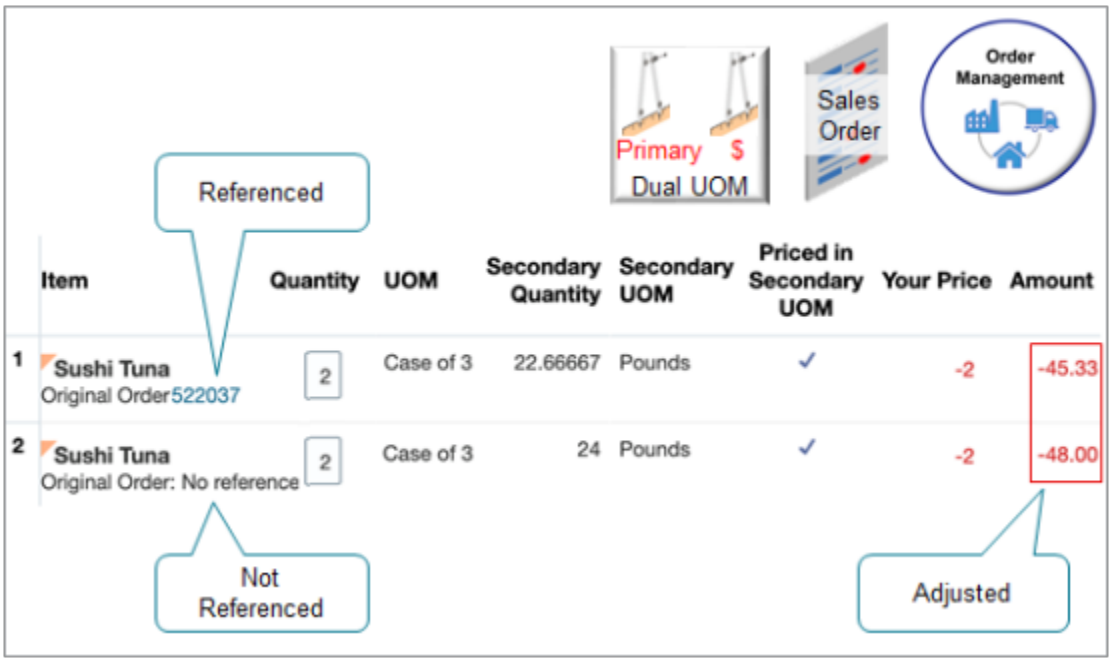
- The Return Quantity defaults to the actual quantity that you shipped.
- Order Management calculates the price on a return order in the same way that it calculated price on the original order. For example, if you price the original order according to the secondary quantity and the secondary UOM, then Order Management prices the return according to the secondary quantity and the secondary UOM.
- Returning the item in the same quantity and UOM that you used in the original order helps to make sure you provide a credit that correctly matches the original charge.
- If you price the item in the secondary unit of measure, then Order Management calculates credit for the line according to the secondary quantity that it fulfilled on the referenced line.
- You can return up to the quantity that Order Management fulfilled on the referenced order line.

Unreferenced Returns

- You can add an unreferenced return line to a sales order.
- Order Management sets the secondary unit of measure on the return line according to how you set up the item in Product Information Management and the item validation organization.
- To calculate the secondary return quantity, Order Management uses the conversion rule that you set up between the return unit of measure and the secondary unit of measure.

- If you track and price the item in the secondary unit of measure, then Order Management calculates credit for the return line according to the secondary return quantity.

Here's an example where one line references the original order and another line doesn't.



Note

- You return a quantity of 2 on each line.
- Order Management prices each line in the secondary unit of measure.
- Order Management prices each line differently depending on whether it references the original order.

If the Return Quantity	Then
On the line that references the original order is 22.66667	○ The credit doesn't depend on the quantity that you receive from Oracle Receiving. ○ You ordered 3 cases of tuna, and the secondary quantity that you fulfill and invoice for the 3 cases is 34 pounds, where each case weighs 11.33333 pounds. ○ Your customer returns 2 of the 3 cases. Order Management will calculate the Secondary Return Quantity as 22.66667 pounds [(2 * 34) / 3], or 2 returned cases multiplied by 34 pounds, then divided by the 3 original cases.
On the line that doesn't reference the original order is 24	○ There's no original return on this line, you never charged the customer for the full price, so you don't need to worry about crediting only a portion of the case of 3. ○ You credit [(2 * 12) -2] = -48, or a quantity of 2 multiplied by 12 pounds equals 24 pounds, multiplied by your price of -2 equals -48.00.

Import

You can import an item that uses dual units of measure in the same way that you import an item that doesn't.

Note

- You don't import attributes for dual units of measure, such as secondary quantity, secondary UOM, secondary ordered quantity, and so on. Instead, Order Management calculates them at runtime according to your set ups.

  Here are the values that this example uses.

Attribute	Value
ProductNumber	Sushi Tuna
OrderedQuantity	3

Attribute	Value
OrderedUOM	Case of 3

- You can import through REST API, a web service, File-Based Data Import (FBDI), or Electronic Data Interchange (EDI).

Import Pricing

You can import a sales order that you already priced.

If the Pricing attribute in the Units of Measure area in Product Information Management contains:

- **Secondary**. Your import must include the price details for the item's secondary unit of measure. If it doesn't, you'll encounter an error during import.
- **Primary**. Your import doesn't have to include price details for the item's secondary unit of measure.

Order Management doesn't check to make sure that the Priced Quantity in the imported order matches the conversion between the quantity ordered in the unit of measure on the order line and the secondary unit of measure for the item. You must do this manually.

Business Intelligence Reports

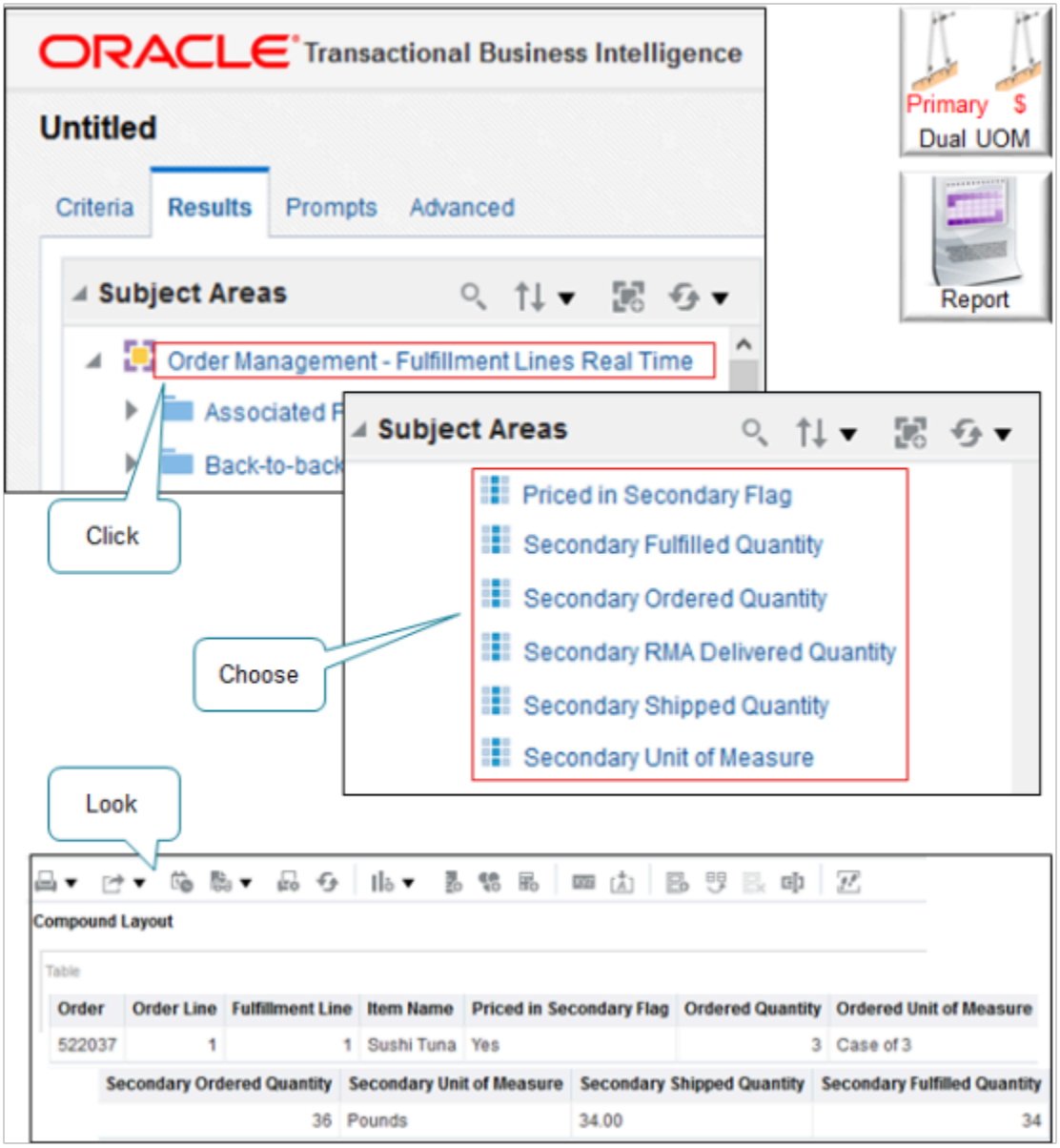
Use attributes that contain dual UOM data in the Order Management Fulfillment Lines Real Time subject area.

- Secondary UOM
- Secondary Quantity
- Priced in Secondary UOM
- Secondary Shipped Quantity
- Secondary Fulfilled Quantity
- Secondary RMA Delivered Quantity

Use them in these dimensions.

- Fulfillment Lines General Details
- Lot and Serial Details

For example:



Other Support

- If you add an item that uses dual units of measure to an order line, then submit that line to fulfillment, then you can't opt out of using dual units of measure. At this point, you're committed to using the feature.
- If you set up an item to use dual units of measure, and if you price it in the primary unit of measure, then you can apply a coverage to the item. If you price it in the secondary, then you can't apply a coverage.
- If you need an orchestration process that does tasks that are specific for your dual UOM item, then we recommend that you use an assignment rule to assign that process instead of using line selection criteria. Also, you can use a dual UOM attribute in your rule, such as Priced in Secondary UOM. For details, see *Guidelines for Assigning Orchestration Processes*.
- Order Management automatically calculates and sets the secondary quantity and the secondary unit of measure on the order line according to your conversion rules. You can't manually modify these values on the order line.

- You can't use dual units of measure with the Reduce Inventory When a Sales Order Doesn't Require Picking or Shipping feature.
- Make sure the setup for the secondary unit of measure for the item is identical in the item validation organization and in the organization that you use to fulfill the order line. If these measures aren't the same, the Order Management work area will display an error message that requests you to change the warehouse or cancel the order line.

Primary and Secondary Quantity

- You can substitute item x for another item y, but only if items x and y use the same primary UOM and the same secondary UOM. Order Management doesn't call pricing after the substitution so it expects the secondary shipped quantities to be in the same unit of measure that you use for the original item.
- If you apply a shipment tolerance on an order line, and if you price the item in the secondary UOM, then the price on the invoice uses the secondary quantity that Order Management fulfilled, regardless of how you set the Quantity to Invoice for Overshipment parameter. For details, see *Manage Order Management Parameters*.
- If you create an order revision, then you can't modify the secondary UOM or the secondary ordered quantity in the revision.

Secondary UOM

You can't use the secondary UOM in some scenarios. You must use the primary UOM for these scenarios.

- Model, such as an assemble-to-order item, pick-to-order item, or kit
- Item that you can't ship, such as a subscription or coverage
- Cost list or pricing guideline
- Promising through Global Order Promising
- Reserve inventory
- Consign inventory
- Outside processing
- Contract manufacturing
- Work order in discrete manufacturing
- Global Trade Management
- Field service
- Service logistics
- You can't use the secondary UOM as the ordered UOM. If you price the item in the secondary UOM, then you can use the primary UOM or any UOM that's in the primary's UOM class as the ordered UOM.

Order Management Extensions

You can use these attributes in an extension.

- Secondary UOM
- Secondary Quantity
- Priced in Secondary UOM
- Secondary Shipped Quantity
- Secondary Fulfilled Quantity
- Secondary RMA Delivered Quantity

Here's an example extension pseudocode.

```
If Priced in Secondary UOM is Yes, then set the value of an extensible flexfield on the order line.
```

You can only read these attributes. You can't update them.

Guidelines

If you use more than 1 UOM to track the item.

- You must set up a conversion between the primary UOM and the secondary UOM.
- And if your flow includes a purchase order or Advance Shipment Notice, then you must set up a conversion between the UOM on the order line and the UOM on the purchase order or advance shipment notice.

Pricing

If you set the Pricing attribute to Secondary for the item in Product Information Management, then Order Management prices the item according to the secondary unit of measure regardless of whether you set up pricing in the Pricing Administration work area according to the primary or to the secondary unit of measure. Pricing won't prevent you from using the primary, but if you use dual units of measure for an item, then you must set up pricing according to the secondary unit of measure. If you don't, you will encounter a runtime error.

For example, assume you have an item where the primary is Kilograms and the secondary is pounds for your item. If you set up the price in Kilograms, then Pricing won't automatically calculate the price in Pounds. You must manually set it up.

However, if you set the Pricing attribute to Primary in Product Information Management, then you can order and price the item according to the primary UOM or to any UOM that's in the primary's UOM class of the item that you specify in the price list in Pricing Administration.

For details, see *Overview of Setting Up Dual Units of Measure*.

Related Topics

- Residual Quantity Transactions with Dual UOMs
- Track Items in More Than One Unit of Measure

Set Up Dual Units of Measure

Set up dual units of measure so you can order, price, manufacture, receive, pick, pack, and ship an item in the primary UOM but price it in the secondary UOM.

Assume you're in the Fresh Fish 4 U organization, and you're a fishmonger who sells an item named Sushi Tuna. You capture tuna in the open ocean and bring them to port every day, where you sell the whole fish to your favorite restaurateur, Fancy Fish. You price it by pound, not by each fish, so you need to price it in the secondary unit of measure. Sushi Tuna is a good candidate for dual units of measure because a tuna naturally varies in size, you typically sell each tuna as one fish each, but you price it by weight. You sell each tuna fish for a different amount because each one has a unique weight. You need to track the item in the primary and secondary UOM throughout the order fulfillment lifecycle.

Summary of the Setup

1. Create the item.
2. Manage the conversion.

3. Set up pricing.
4. Do other setups.
5. Test your setup.
6. Manage inventory.

Look at an illustration of this set up. For details, see *Track Items in More Than One Unit of Measure*.

Each work area that you use in this procedure requires you to have a different set of privileges. For details, see *Privileges That You Need to Implement Order Management*.

Create the Item

1. Go to the Product Information Management work area, then click **Tasks > Create Item**.

   You must create a new item. You can't add dual units of measure to an item that already exists.
2. In the Create Item dialog, set the values.

Attribute	Value
Organization	Fresh Fish 4 U
Create New	Enabled
Number of Items	1
Item Class	Root Item Class

3. Make sure the Selected List window contains only Finished Goods, then click **OK**.
4. On the Create Item page, set the values.

Attribute	Value
Item	Sushi Tuna
Description	The world's tastiest tuna. Remember, you can tune a piano, but you can't tune a fish.
Primary Unit of Measure	Each
Tracking Unit of Measure	Primary and Secondary
Pricing	You have two choices.

Attribute	Value
	○ **Primary**. Pricing will use the primary unit of measure to price the item on the order line. The Order Entry Specialist can change the UOM attribute on the order line to any value that's in the primary UOM's class, and Pricing will recalculate the price according to the secondary quantity in the secondary unit of measure. ○ **Secondary**. Pricing will use the secondary unit of measure to price the item on the order line. If the Pricing attribute for the item validation organization contains Secondary, then you must price the item in the secondary unit of measure. The Order Entry Specialist can change the UOM attribute on the order line to any value that's in the primary UOM's class, and Pricing will recalculate the price. The Order Entry Specialist can't change the Secondary UOM attribute on the order line to another value. For this example, choose Secondary. Consider an example. ○ Assume you set the Item Validation Organization parameter to Fresh Fish 4 U. For details, see *Manage Order Management Parameters*. ○ In Product Information Management, you create the Sushi Tuna item in the Fresh Fish 4 U organization, and you set Sushi Tuna's primary UOM to Each. ○ Each is in the Quantity class, so you can price the item in any unit of measure that's in the Quantity class, such as Case, Sheet, Ream, Vial, Percent, Dozen, Bag, Sack, and so on.
Conversions	Both
Secondary Unit of Measure	Pounds
Defaulting Control	Choose a value. ○ **Default**. Use this value when the secondary unit of measure for your item might change at runtime. ○ **Fixed**. Use this value when the secondary unit of measure for your item won't change at runtime. For details about how to set this attribute, see the How to Set the Defaulting Control section, below.

You can't change some of the values that you set in the Unit of Measure area after you save, so make sure you set them correctly.

5. Click **Specifications > Manufacturing**, then set the value.

Attribute	Value
Structure Item Type	Standard You must use Standard.

6. Click **Sales and Order Management**, then set the values.

Attribute	Value
Sales Product Type	Goods You must use Goods or leave it empty.
Shippable	Yes Can be Yes or No.
Invoiced	Yes
Invoice Enabled	Yes

7. Click **Save > Save and Close**.

How to Set the Defaulting Control

If you set the Defaulting Control attribute to Default or No Default, then you can specify a deviation factor.

Variations sometime happen during fulfillment, particularly with product that varies in weight, such as fish, or your product contains water, the water evaporates over time, which results in less weight. Your user might need to change the quantity on the pick slip to reflect the actual quantity picked. If you set Defaulting Control to Default, then you can use the deviation factor attributes to specify the percent that you will allow your user to set for the Secondary Picked Quantity on the pick slip.

Assume you set up a conversion between the primary Each unit of measure and the secondary Pounds unit of measure as 1 Each equals 10 Pounds, and you then set these values.

Attribute	Value
Positive Deviation Factor	10 If you set the quantity on the order line to 1 Each, then Inventory Management will limit the secondary unit of measure to a maximum of 11 pounds. Here's the math for that. • 10 pounds multiplied by 10% equals a positive deviation of +1. • 10 pounds on the order line plus 1 equals 11. You must use a decimal value.
Negative Deviation Factor	10 If you set the quantity on the order line to 1 Each, then Inventory Management will limit the secondary unit of measure to a maximum of 9 pounds. Here's the math for that. • 10 pounds multiplied by 10% equals a negative deviation of -1. • 10 pounds on the order line minus 1 equals 9.

Attribute	Value
	If you set Positive Deviation Factor to 10 and Negative Deviation Factor to 10, the range of values that the user can set is 9, 10, or 11. You must use a decimal value.

If a change happens between the time you pick the item and ship it, then you can also override the picked quantity and secondary picked quantity, but the secondary shipped quantity must not exceed the secondary picked quantity.

For details, see *Residual Quantity Transactions with Dual UOMs*.

Manage the Conversion

You must set up a conversion between the item's primary unit of measure and the secondary unit of measure. You can use the standard conversion for many items or you can set up a conversion only for the item. In this example you set up a conversion for the item.

- 1 Pound equals 0.25 Each. If you order 1 Each of Sushi Tuna, then the value in the Pounds secondary UOM for the item is 4 pounds.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Items
    - Task: Manage Units of Measure for Interclass Conversion

    In this example you're converting between units of measure that are in two different classes, the Weight class and the Quantity class, so you will create an interclass conversion. If your units of measure are in the same class, then use the Manage Units of Measure for Intraclass Conversion task.
2. On the Manage UOM Interclass Conversions page, click **Change Organization**, set the value, then click **OK**.

Attribute	Value
Organization	Fresh Fish 4 U

3. Click **Actions > Add**, set the values, then click **Save and Close**.

Attribute	Value
Item	Sushi Tuna
From Base UOM	Pounds
From Class	Weight

Attribute	Value
Conversion	0.25 You sell the Bullet Tuna species. Assume each of these tuna averages about 4 pounds each.
To Base UOM	Each
To Class	Quantity

4. Set up a standard conversion.

   Assume you package fish in a case, one case contains 3 tuna, so you also need to set up a rule that converts case to each.

   - In the Setup and Maintenance work area, click **Tasks > Search**, search for and open Manage Units of Measure.
   - On the Manage Units of Measure page, click **Actions > Add**, set the values, then click **Save**.

Attribute	Value
UOM Code	CS3
UOM Name	Case of 3
Description	Case of 3
Class Name	Quantity
Base UOM Name	Each

   - Click **Manage UOM Standard Conversions**.
   - On the Manage UOM Standard Conversions page, click **Actions > Add**, set values, then click **Save and Close**.

Attribute	Value
UOM Name	Case of 3
Conversion	3

Attribute	Value
Base UOM Name	Each
Class Name	Quantity

Set Up Pricing

For this example, assume you use the Corporate Segment Price List, and the sale price for tuna on the commercial market is about $2 a pound.

1. Go to the Pricing Administration work area, then click **Tasks > Manage Price Lists**.
2. Search for and open the Corporate Segment Price List.
3. On the Edit Price List page, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Item	Sushi Tuna
Pricing UOM	Pounds
Line Type	Buy
Primary Pricing UOM	Selected. Enabling this option tells Pricing to use the value that you set in the Pricing UOM to price the item. Don't confuse Primary Pricing UOM with the Primary Unit of Measure that you set for the item in Product Information Management. The Primary Pricing UOM attribute is specifically for pricing, not for the item.

4. Click **Create Charge**, set the values, then click **Save and Close**.

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	2

Attribute	Value
Allow Manual Adjustment	Selected. Market conditions probably vary every day, so let your users adjust the price manually.

Do Other Setups

Set up rules for your item in the same way that you would set them up when you don't use dual units of measure. These rules might include available-to-promise rules, sourcing rules, assignment sets, and so on in Global Ordering Promising.

Test Your Setup

Create an order, search for your item on the catalog line, then add it to an order line.

Note

- You use the Catalog Line on the sales order to search for the item. It displays the quantity and price for the primary and the secondary UOM.
- You can't change the secondary quantity or secondary UOM on the order line, but you can change the primary quantity and Order Management will automatically calculate the secondary quantity and the amount.

Try it.

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Fancy Fish

Attribute	Value
Business Unit	Fresh Fish 4 U

2. Search for the Sushi Tuna item on the catalog line.

   The catalog line sets the unit of measure to the primary, by default. It also displays primary and secondary attributes which makes it easy to see the conversion.

   For this example, assume you change the Quantity to 3 and the UOM to Case of 3.

Attribute	Value
Quantity	3 This is the primary quantity.
UOM	Case of 3 The primary UOM of the item is Each. You can change it to any UOM that's in the primary UOM's class, such as Case of 3.
Secondary Quantity	36 The catalog line uses your conversion rules to automatically calculate the secondary quantity. o 1 case contains 3 tuna. o 1 tuna equals 4 pounds. o 3 tuna multiplied by 4 pounds means each case weighs 12 pounds. o 12 pounds for each case multiplied by the primary quantity of 3 equals a secondary quantity of 36.
Secondary UOM	Pounds
Sale Price	2
Line Price	72.00 Its the secondary quantity of 36 pounds multiplied by the unit price of 2 equals 72.00.
Information Icon	Let's you know that you're using dual UOM tracking on this item.

3. Click **Add**, then notice that the sales order adds the values that you set on the catalog line to the order line. Here are a few more details you should take note of.

Attribute	Value
Item	Contains your dual UOM item.
Priced in Secondary UOM	Contains a value. - **Selected**. Pricing priced the order line according to the value in the Secondary UOM attribute on the order line. In this example, that would be Pounds. - **Not Selected**. Pricing priced the order line according to the value in the UOM attribute on the order line. In this example, that would be Each. Order Management gets this value from the Pricing attribute that you set in Product Information Management.

4. Click **Submit**.

   The orchestration process schedules and reserves the item just like it does any other sales order.

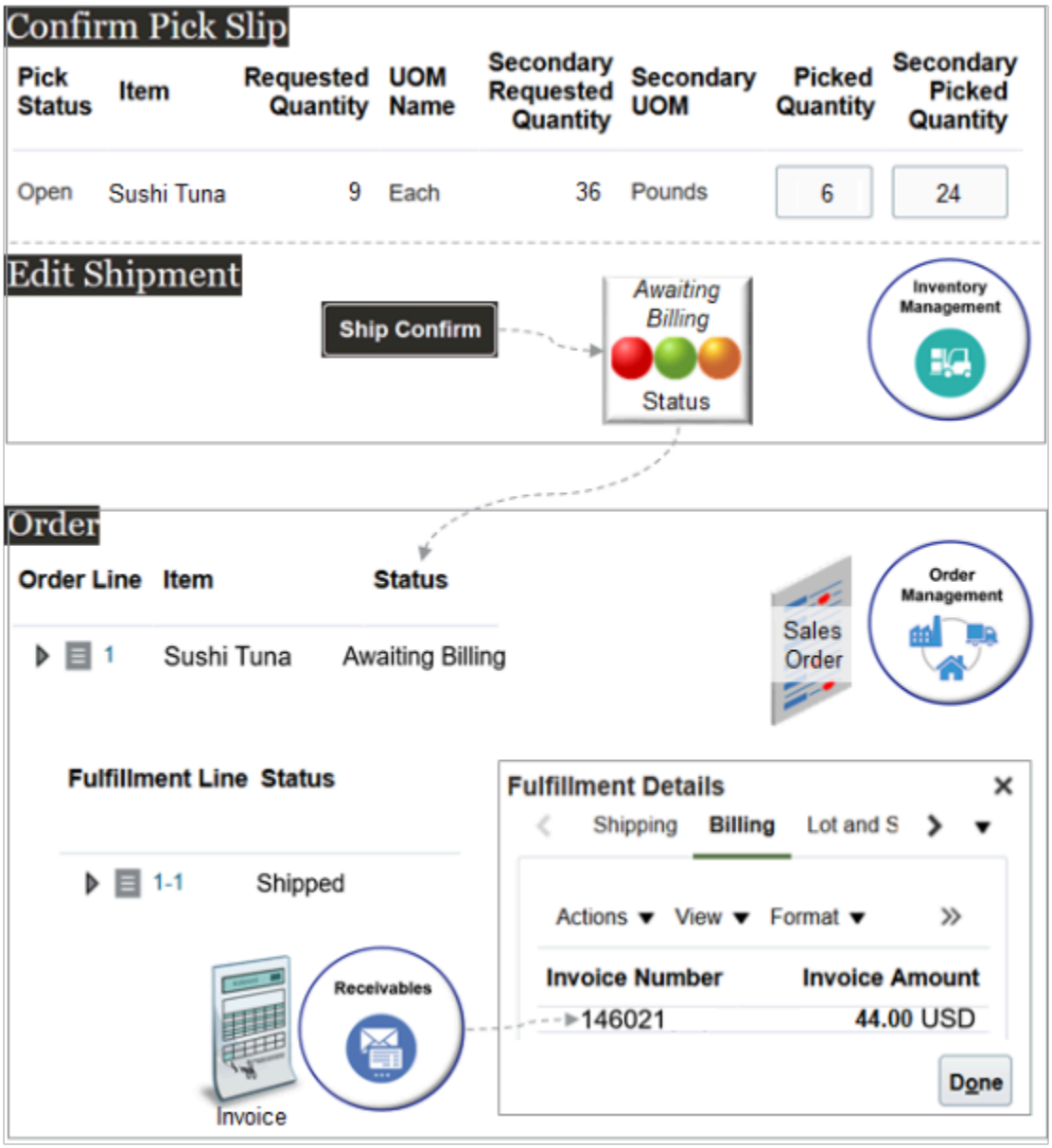
   - The orchestration process schedules the order according to the values in the Quantity attribute and the UOM attribute on the order line.
   - The secondary unit of measure and the secondary quantity doesn't affect scheduling.

Assume the order number is 525940. Here's a summary of the order line details.

Line	Item	Quantity	UOM	Status	Secondar Quantity	Secondar UOM	Price	Extended Amount	Assessab Value	Shipped Quantity	Secondar Shipped Quantity	Secondary Fulfilled Quantity
1	Sushi Tuna	3	Case of 3	Awaiting Shipping	36	Pounds	$2	$72	$72	empty	empty	empty

Manage Inventory

Use Inventory Management and Order Management together to track your UOMs during fulfillment.



Here's what's happening.

- You click **Submit** on the sales order, then Order Management creates and sends a shipment request to Oracle Shipping.
- Oracle Shipping creates a shipment request and populates the requested quantity and the secondary requested quantity according to the standard conversion that you set up for the item.
- You can also use the Inventory Management work area to manually create a pick wave, confirm the pick slip, and confirm the shipment.
- This feature maintains values for various attributes, such as Secondary Requested Quantity and Secondary UOM. It maintains and displays them throughout your dual UOM flow across applications.
- Inventory Management lets Order Management know that it confirmed the shipment, then you use the Order Management work area to see that it updated the status on the order line to Awaiting Billing.

- You can also use the Order Management work area to examine the status on the fulfillment line, such as Shipped, and to get the same invoice details that Oracle Receivables has on the invoice for the fulfillment line.

Notes.

- The Requested Quantity is in the Primary UOM Each. It contains 9 because you ordered 3 cases, and each case has 3 tuna.
- In this example, assume one of the cases sits too long on the shelf and spoils, you pick only 2 cases instead of the requested 3. The Picked Quantity is in the primary UOM Each, so you set Picked Quantity to 6 because each case has 3 tuna. Each case has 3 tuna that weigh 4 pounds each, so Inventory Management automatically sets the Secondary Picked Quantity to 24 because each case weighs 12 pounds and you picked 2 cases.

In this example, you set the picked quantity to a value that's different from the requested quantity. Let's see how you do that.

1. Create a pick wave.
    - Go to the Inventory Management work area.
    - Click **Tasks > Show Tasks > Shipments > Create Pick Wave**.
    - On the Create Pick Wave page, set the values.

Attribute	Value
Release Rule	Standard
Ship From Organization	M1
Order	525940
Customer	Fancy Fish

    - Click **Show More**, click **Options**, then set the value.

Attribute	Value
Autoconfirm Picks	Not selected. You can normally leave this enabled, but disable here so you can examine the confirm flow.

    - Click **Release Now**.
    - In the Confirmation Dialog, notice the text `Number of shipment lines released to warehouse: 1.`

      Assume Inventory Management creates pick wave 1307257.

2. Confirm the pick slip.
    - On the Inventory Management page, click **Tasks > Confirm Pick Slips**.
    - On the Confirm Pick Slips page, search for the value.

Attribute	Value
Pick Wave	1307257

    - In the search results, click the **link** in the Pick Slip column.
    - In the Picks area, note the values.

Attribute	Value
Requested Quantity Maximum Picked Quantity	9 Note that its the quantity in the primary UOM, Each. You ordered 3 cases of tuna, and each case has a quantity of 3, so the total is 9.
Secondary UOM	Pounds
Secondary Requested Quantity	36 The conversion rule that you created calculates this value, which is a conversion between the item's primary unit of measure Each and the item's secondary unit of measure Pounds.
Secondary Picked Quantity	Empty

    - Set the value.

Attribute	Value
Picked Quantity	9 Its the quantity in the primary UOM Each.

    - Notice that the work area automatically updates the Secondary Picked Quantity according to the value that you set in the Picked Quantity.

Attribute	Value
Secondary Picked Quantity	36

Attribute	Value
	Its the quantity in the secondary UOM Pounds.

- You set the Deviation Factor earlier in this procedure to +/- 10%. 10% of 36 is 3.6, so you can set the Secondary Picked Quantity to a:
    - Low of 32.4 (36 minus 3.6).
    - High of 39.6 (36 plus 3.6).

  There's some variability of the weight for tuna. You measured the weight of the two cases you picked and find that they lost weight through evaporation. They actually weigh 34 pounds, not 36 pounds. So you manually decrease the value from 36 to 34, which is above the low range that the deviation allows.

Attribute	Value
Secondary Picked Quantity	34 The Deviation Factor attributes that you set for the item in Product Information Management controls the value that you can set. If you find that the cases you picked weigh less than 32.4 or more than 39.6, then you can't use them. You must pick some other cases.

- Set the value, then click **Confirm > Confirm and Go to Ship Confirm**.

Attribute	Value
Ready to Confirm	Selected.

- Confirm the shipment.
    - On the Edit Shipment page, confirm the values, then click **Ship Confirm**.

Attribute	Value
Line Status	Staged
Requested Quantity Staged Quantity	9 Its the quantity in the primary UOM, Each.

Attribute	Value
Secondary Picked Quantity	34
Secondary Requested Quantity	36

Inventory Management sends an update to Order Management that it confirmed the shipment, and Order Management updates the status for the order to Awaiting Billing.

Here's a summary of the values on the shipment line.

Item	Order Line	Line Status	Quantity UOM	Requested Quantity	Staged Quantity	Secondary Picked Quantity	Secondary Quantity UOM
Sushi Tuna	1	Staged	Each	9	9	34	Pounds

Examine the Sales Order

1. Go to the Order Management work area and open sales order 525940.
2. Notice that the sales order total is different than when you submitted it. This happens because you reduced the quantity that you actually picked.

   Here are the order line details.

Item	Quantity	Status	UOM	Your Price	Amount
Sushi Tuna	3	Awaiting Billing	Case of 3	2	68

   Order Management uses the quantity that it receives from Shipping to recalculate the amount so it reflects that actual quantity that you shipped.
3. Click **Actions > Switch to Fulfillment View**.
4. In the Attributes area, click **Shipping**, then notice the values.

Attribute	Value
Shipped Quantity	3
Fulfilled Quantity	You shipped and fulfilled all 3 cases of tuna in the primary UOM, Each.
Secondary Shipped Quantity	34
Secondary Fulfilled Quantity	You shipped and fulfilled a total of 34 in the secondary UOM, pounds.

Note

- Order Management uses the standard conversion that you created and the value in the Quantity attribute on the line to set the Secondary Quantity.
- Order Management populates the Secondary Fulfilled Quantity after the fulfillment completion step finishes. For details, see *Guidelines for Setting Up Orchestration Process Steps*.

See How the Order Line Status Gets Updated

1. Make sure you have the privileges that you need to administer Oracle Maintenance Management.
2. Go to the Maintenance Management work area, then click **Tasks > Manage Assets**.
3. Click Show Filters, then search for the values.

Attribute	Value
Item	Sushi Tuna
Sales Order Number	525940

4. In the search results, click the **link** in the Number attribute, such as **100100375360194**.

   100100375360194 identifies the asset that inventory management created for the item.
5. On the Asset page, notice how the feature has brought values from Order Management into Maintenance Management.

Attribute	Value
Asset Number	100100375360194
Description	The world's tastiest tuna.
Item	Sushi Tuna
Quantity	3 Case of 3
Secondary Quantity	34 Pounds
Customer	Computer Service and Rentals

   These values describe details about the item that you shipped to your customer.
6. Click **Last Sales Order Details**. Its the fourth icon from the top.
7. Notice how the Last Sales Order Details page, contains a breakdown of the pricing details. For example:

Attribute	Value
Charge Definition	Sale Price
Priced Quantity UOM	Pounds
Product Unit Price	2
Primary	Contains a check mark.
Price Element	Your Price

Recall that the order line is still in the Awaiting Billing status. It means that Order Management has sent the line to Oracle Receivables, and the order line is ready to invoice.

Now let's see how we get that to Closed status.

8. Go to the Scheduled Processes work area, then run the Import AutoInvoice scheduled process to invoice the sales order.

Attribute	Value
Business Unit	Vision Operations
Transaction Source	Distributed Order Orchestration
From Sales Order Number	525940
To Sales Order Number	525940

For details, see *Update Intercompany Receivables Invoice Import Details*.

9. Go to the Order Management work area and open sales order 525940.
10. Notice the status on the order line.

Attribute	Value
Item	Sushi Tuna
Status	Closed

Attribute	Value
	It means the Oracle Receivables has successfully invoiced the line.

Examine the Invoice Details on the Fulfillment Line

1. Click **Switch to Fulfillment View**, then notice that various areas in the fulfillment line have the same details about the primary and secondary that you see on the order line, such as the General area, Shipping, Item Details, and so on.
2. In the Fulfillment Lines area, click your **fulfillment line**, then notice the value on the line.

Attribute	Value
Status	Shipped

3. Click **Actions > View Fulfillment Details**.
4. In the Fulfillment Details dialog, click **Billing**, then notice the values for the line.

Attribute	Value
Invoice Number	146021
Invoice Amount	68.00 USD These values indicate that the flow has successfully invoiced the sales order.

Order Management creates and sends a request to Oracle Accounts Receivable to create an invoice. Receivables uses the pricing unit of measure.

If You Use This Attribute on the Order Line To Price the Item	Receivables Will Use These Attributes From the Order Line When It Creates The Invoice
UOM	Fulfilled Quantity UOM
Secondary UOM	Secondary Fulfilled Quantity Secondary UOM

Examine Receivables

1. Go to the Billing work area under Accounts Receivable.

2. On the Billing page, search for the value.

Attribute	Value
Transaction Number	146021 Its the invoice number that you noted on the fulfillment line in Order Management.

3. On the Manage Transactions page, in the Transaction Number attribute, click 146021.
4. On the Review Transaction page, in the Invoice Details area notice the values on the invoice line.

Attribute	Value
Item	Sushi Tuna
UOM	Pounds Its the secondary UOM for the item. Order Management only sends the UOM that it used to price the item to Accounts Receivable. In this example, Order Management priced the item in the secondary UOM, so it doesn't send the primary quantity or primary UOM to Accounts Receivable.
Quantity	34 Its the quantity that Oracle Shipping actually shipped, in the secondary UOM.
Unit Price	2
Amount	68

Examine and Fix the Order Total

1. Examine the order total.
    - Go to the Order Management work area, then open sales order 525940.
    - On the Order Page, notice the value at the top of the page.

Attribute	Value
Total	86.40

Attribute	Value

The order total is 86.40 but the amount on the invoice is 68. Why?

- Click the warning icon next to the order total, then examine the warning.

```
The total amount for sales order 525940 isn't correct. The order fulfillment process updated the
 amount on one or more order lines in sales order 525940, but the order total doesn't include
 these updates. To update the total, someone with a job role to manage scheduled processes must go
 to the Scheduled Processes work area, then run the Update Sales Order Totals scheduled process.
 Specify order number 525940 when you run the scheduled process. You can also set it up to run
 automatically on a schedule.
```

This happens because Accounts Receivable recalculated the extended amount for the fulfillment line according to the actual quantity that Shipping shipped, but this change hasn't been sent to Order Management. Recall that you changed the quantity during shipping.

- Click the **86.40** total, then examine the price breakdown.

Attribute	Value
Total List Price	72.00
Total Tax	14.40
Pay Now	86.40

86.40 reflects the quantity that existed when you submitted the sales order, not that you actually shipped.

2. Get the order total back in sync with the invoice.

Go to the Scheduled Processes work area, then run the *Update Sales Order Totals* scheduled process. For this example, set these values when you run it.

Attribute	Value
From Order Number	525940
To Order Number	525940

We recommend that you run the process only after you invoice the sales order. This makes sure that the order total will match the invoice total.

3. Go to the Order Management work area, open your sales order, click **81.60** total, then verify that the sales order total now matches the invoice total.

Attribute	Value
Total List Price	68.00
Total Tax	13.60
Pay Now	81.60

Related Topics

- Privileges That You Need to Implement Order Management
- Residual Quantity Transactions with Dual UOMs

Calculate Credit for Referenced Returns

Specify whether to calculate credit according to the secondary quantity that you actually receive or to the quantity that your customer requests when they place a return order. This feature applies to a referenced return that has a dual UOM item that you price in the secondary UOM. For details, see *Overview of Setting Up Dual Units of Measure*.

If your customer returns an item that uses more than one unit of measure, and if the return references the original sales order, then Order Management sets the return quantity and the secondary return quantity to the actual quantity that you fulfilled on the original order, by default. Pricing calculates price on the return in the same way that it calculated price on the original order.

If you price the item in the secondary unit of measure, then Pricing calculates credit for a referenced return according to the secondary fulfilled quantity on the referenced order.

Consider an example.

- Assume your customer orders 3 cases of a dual UOM item, and the secondary unit of measure for the item is Pounds.
- You set up a rule that converts the primary unit of measure Case to the secondary unit of measure Pounds from 1 case to 12 pounds.
- You place an order for 3 cases and Order Management sets the secondary quantity to 36 pounds.
- You ship all 3 cases but you set the secondary quantity when you ship the item in Oracle Shipping to 34 pounds to account for some variability in the actual weight that you ship.
- The secondary quantity that you fulfil and invoice for the 3 cases is 34 pounds, where each case weighs 11.33333 pounds.
- You invoice the customer for 34 pounds.
- Sometime later, your customer returns 3 of the 3 cases that you shipped.

- Pricing calculates the Secondary Return Quantity as 34 pounds. Pricing uses this quantity to calculate credit regardless of the secondary quantity that you actually receive and deliver back to the warehouse. This way, the credit matches the amount that you invoiced the customer on the original order line.

In some situations, such as in the consumer goods and food processing industries, the quantity that you originally shipped might be less than the quantity that you receive on the return. You can use this feature to select the quantity you want to use to calculate credit for the referenced return. You can calculate credit according to the secondary quantity that you actually receive.

Assume the secondary quantity is 34 pounds on the original order, but its 35 pounds when the item on the return order actually arrives at your receiving dock. You can use this feature to record the quantity as 35 pounds instead of 34 pounds and calculate credit to your customer for 35 pounds. Your customer will receive a credit that's higher than what they paid on the referenced order.

Here's your setup.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Parameters
2. On the Manage Order Management Parameters page, set the Type of Secondary Quantity for Referenced Returns parameter.

Value	Description
Returned Quantity in the Secondary UOM	Calculate credit according to the quantity that you actually receive on the return order from your customer, in the secondary unit of measure.
Ordered Quantity in the Secondary UOM	Calculate credit according to the quantity that you shipped on the referenced order, in the secondary unit of measure.

You can also specify whether to accept a credit amount that's more than the invoiced amount.

1. Go to the Setup and Maintenance work area, then click **Tasks > Manage Transaction Types**.

    For details, see *Transaction Types*.
2. Search for, then open the Manage Transaction Types page.
3. On the Manage Transaction Types page, in the search area, enter `Invoice` in the Name attribute, then click **Search**.
4. In the search results, in the Name column, click the transaction type that you use in your implementation, such as Invoice_1.
5. On the Edit Transaction Type page, enable the Allow Overapplication option.

    If you don't enable Allow Overapplication, and if the credit amount is more than the original invoice amount, then the credit will fail, and you can't remove the failure from Oracle Receivables until you enable it. We recommend that you consult with your sales and finance departments about whether to enable this option.

Residual Quantity Transactions with Dual UOMs

If you have a dual UOM item, and if a transaction drives the on-hand quantity for the item in the primary UOM to zero, then Inventory Management automatically creates a residual quantity transaction, but the secondary on-hand quantity isn't zero.

If the secondary on-hand quantity is:

- Less than zero, the residual quantity transaction will have a residual quantity receipt.
- Greater than zero, the residual quantity transaction will have a residual quantity issue.

You can use the Defaulting Control attribute to specify how to handle these differences.

Value	Description
Fixed	Inventory Management will use the primary quantity and your UOM conversion to determine the secondary quantity.
Default or No Default	The deviation that you specify will determine the secondary quantity.

Let's take an example where you allow a 10% deviation in either direction. The conversion from primary to secondary is 1 to 1, such as KG and Liters. And, let's say you receive 100 KG and 100 Liters.

Now you do a few issues of the material, with some deviation from the standard conversion. Let's say you issue 50 KG and 51 Liters to a batch or shipment because the material expanded during fulfillment. That leaves you with 50 KG and 49 Liters on hand. Now you do another issue of 50 KG and 51 Liters. This leaves your on-hand quantity as 0 KG and -2 Liters. Obviously, that's not physically possible. If you have 0 KG, you must also have 0 Liters. Inventory Management will automatically create a residual quantity receipt for 0 KG and 2 Liters to bring your on-hand quantity in sync to 0 KG and 0 Liters. Similarly, if you had been left with 0 KG and a positive 2 Liters, Inventory Management will create a residual quantity issue.

You will usually see these residual quantity transactions when you allow a deviation. However, if you set Defaulting Control to Fixed, then the secondary on-hand quantity can sometimes get out of sync with the primary quantity and the UOM conversion because rounding errors accumulate. If a transaction leaves on-hand quantity at zero in the primary quantity, but not zero in the secondary quantity, then you'll have a residual quantity transaction.

> **Note:** The reverse isn't true. If a transaction leaves on-hand quantity as zero in the secondary quantity, but not zero in the primary quantity, then Inventory Management doesn't automatically create a residual quantity issue for the primary UOM. You can create a miscellaneous issue to make the primary UOM 0.

Learn how to set the Defaulting Control attribute. For details, see *Set Up Dual Units of Measure*.

Related Topics

- Overview of Setting Up Dual Units of Measure
- Track Items in More Than One Unit of Measure
- How Units of Measure, Unit of Measure Classes, and Base Units of Measure Relate to Each Other

Agreements

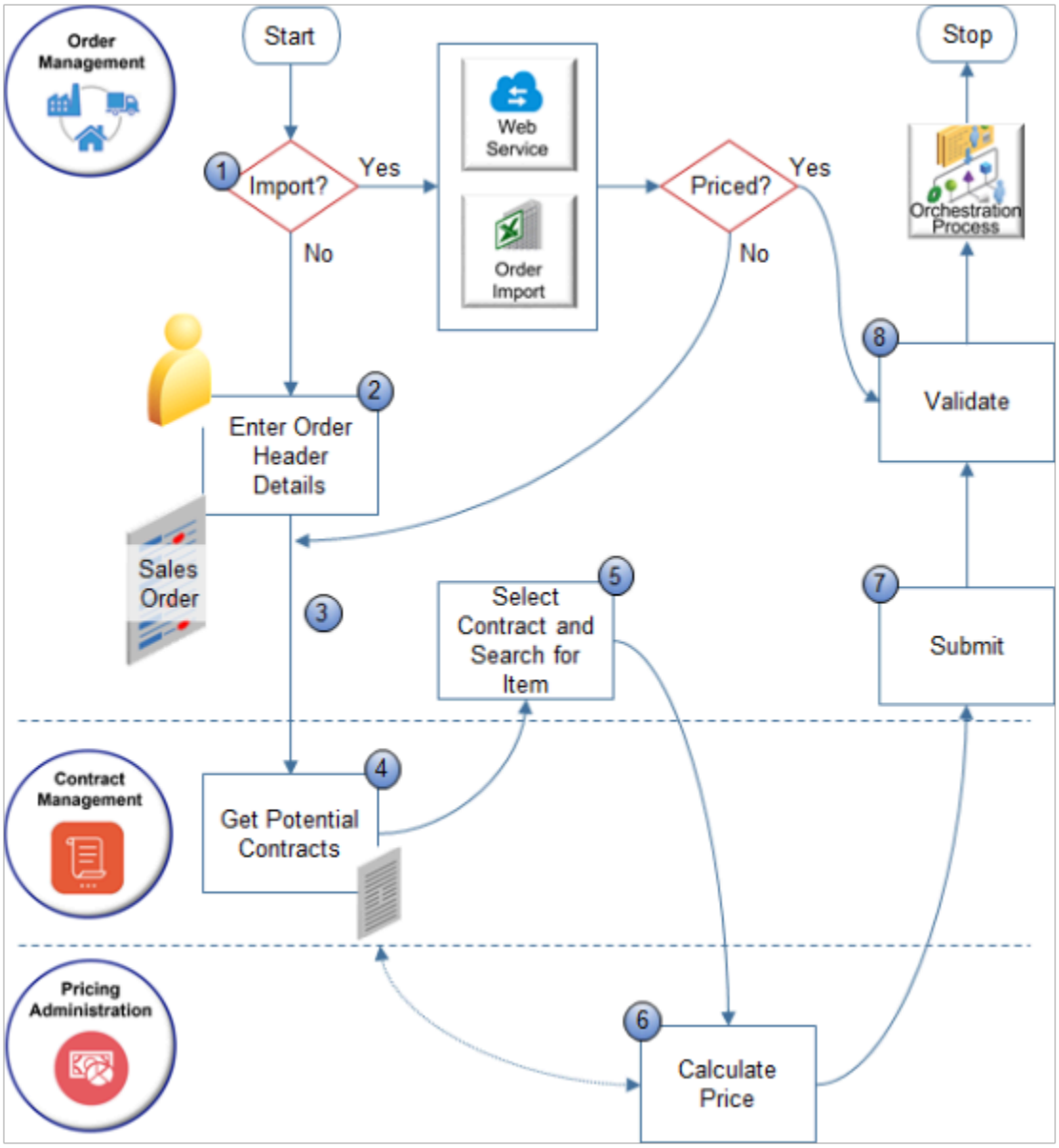
Overview of Setting Up Sales Agreements in Order Management

Set up Order Management so it uses a sales agreement that applies pricing terms when your customer buys from your company, such as offering a reduced price when buying a specific quantity of an item over time.

Here's what you can do with sales agreements:

- Set up a sales agreement that includes price adjustments in Enterprise Contracts.
- Reference a sales agreement from a sales order in Order Management.
- Apply contract pricing on a sales order.
- Automatically enforce and track contractual price obligations.
- Monitor price adjustments, starting with the sales agreement, and then throughout the sales order's fulfillment lifecycle.

Here's how it works:



Here's what the numbers mean:

1. If you use a web service or file to import the order, and if your source system:
    - **Already priced the order**. Skip steps 2 through 7. Instead, validate the order and start orchestration.
    - **Didn't price the order**. Go to step 3.
2. You set the customer and business unit on the order header.
3. Order Management sends the sold-to party, business unit, currency, and current system date to Contract Management.

4. Contract Management searches all of its contracts. It applies filters during the search.
    - Contract Type equals SalesAgreement.
    - Business Unit, Currency, and Primary Party equal the values that Order Management sends.
    - Contract Start Date happens on or before system date, and Contract End Date happens on or after system date.
    - Contract status equals Active.

    Contract Management sends the contracts that pass these filters to Order Management.

    Contract Management sends these attributes for each contract:

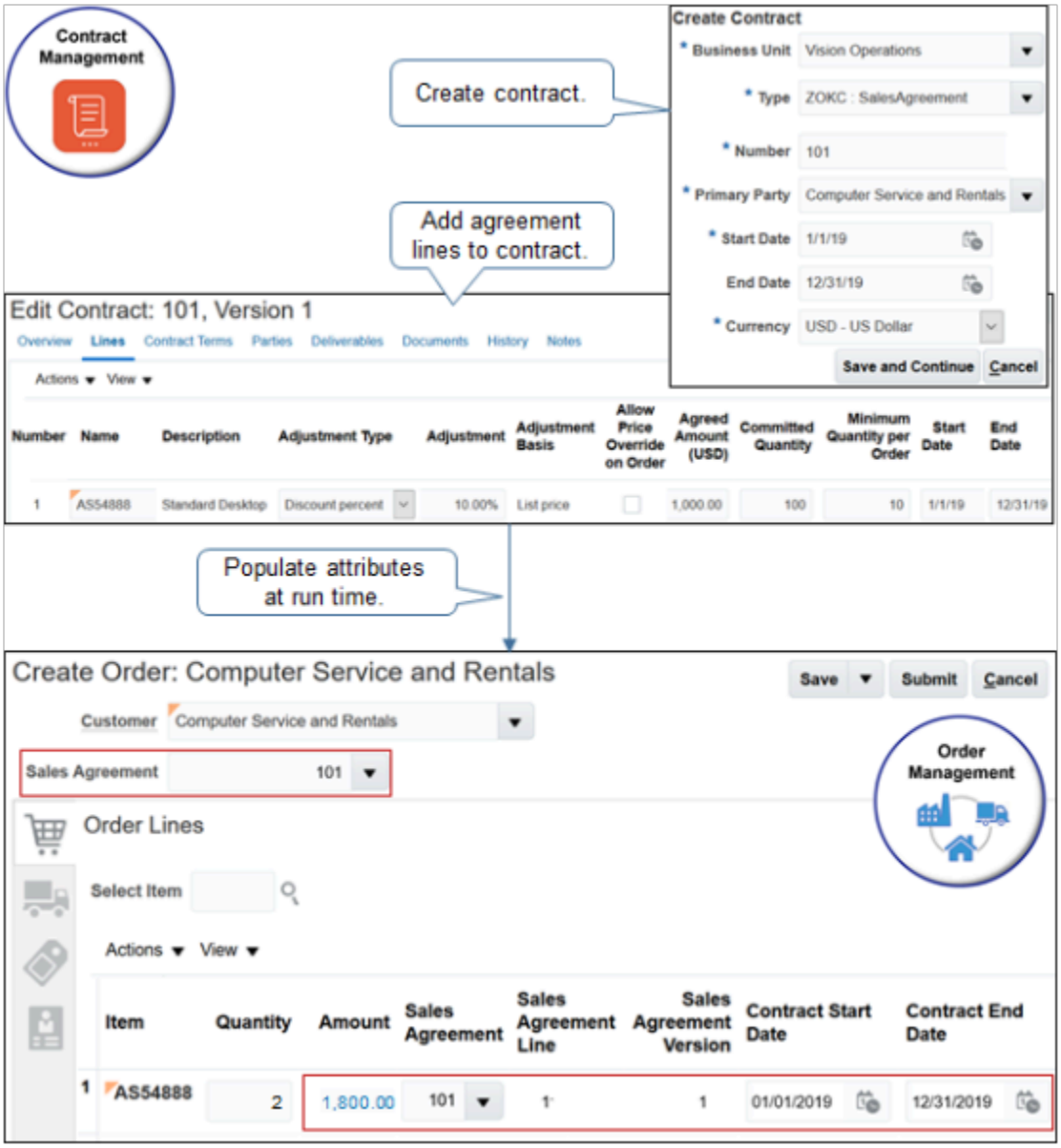
    - Contract ID
    - Version Number
    - Contract Number
    - Contract Name
    - Description
    - Start Date
    - End Date
5. Select contract.
    - Order Management displays the list of filtered contracts in the Sales Agreement attribute on the order header.
    - If Contract Management sends only one contract, then Order Management sets Sales Agreement to this contract, by default.
    - You select a contract in the Sales Agreement attribute. Order Management references this value later when it sets the sales agreement number on each order line.
    - Order Management searches for an item on the catalog line, then sends a request to Pricing Administration to calculate the price. Assume you already defined an agreement line for the item.
6. Pricing gets contract details from Contract Management, prices the order line, then sends the result to Order Management.
7. You click Submit.
8. Order Management validates the sales order to make sure the contract is active, the current system date happens within the contract date, that the contract line and version are valid, then starts orchestration.

Note

- Order Management processes the order line, and then creates a revision for each line as soon as you enter it. It repeats step 6 for each line.
- If you don't set an agreement on the order header, and if you do set an agreement on the order line, then Pricing prices the item in the same way that it prices without an agreement.
- If you set the agreement only on the order header, then Order Management cascades that agreement to all the order lines when you click Submit.

Set Up the Contract

Order Management uses the values that you set in the Contracts work area to populate attributes on the sales order.



Here's what Order Management does.

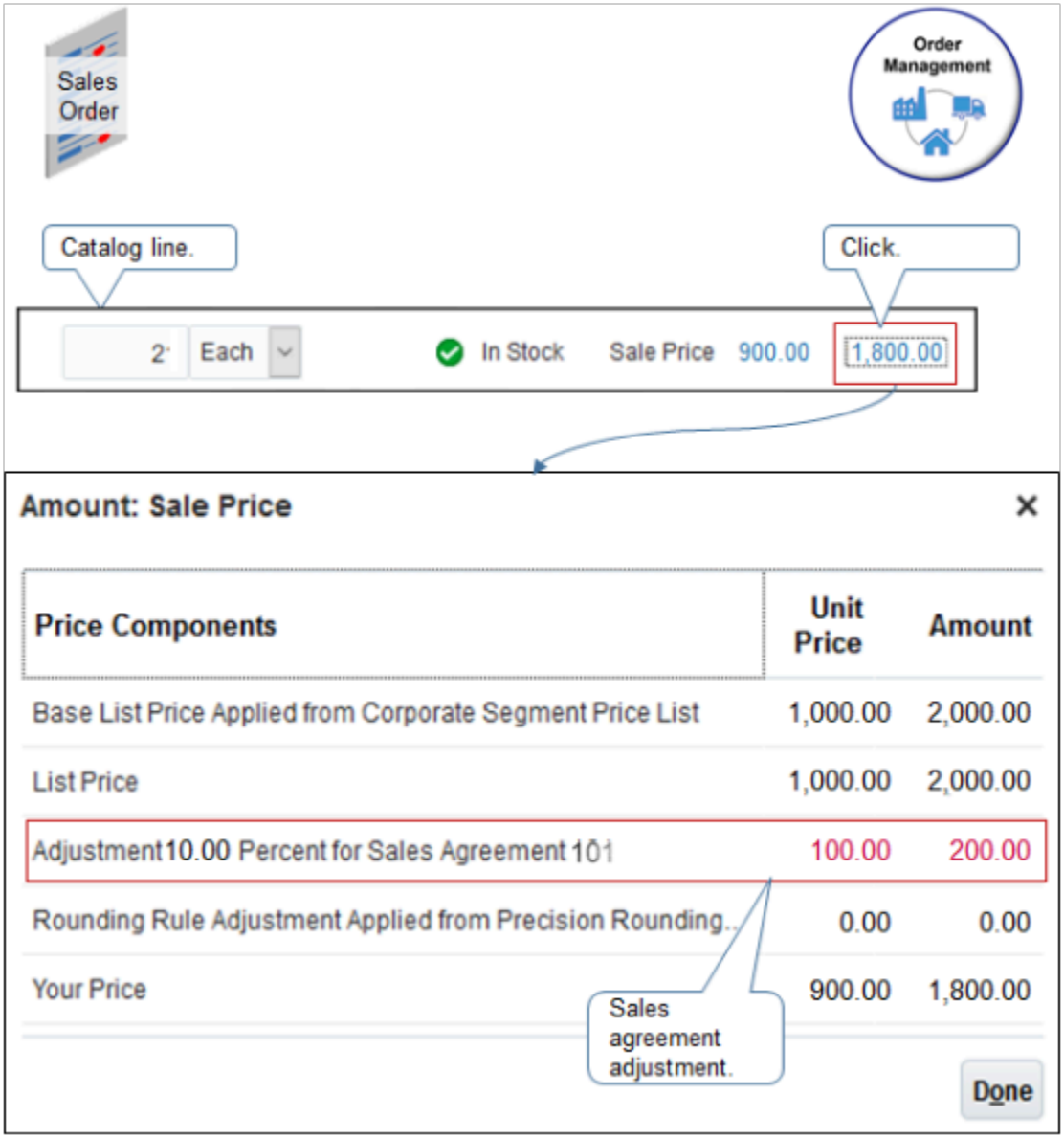
- Filters the values that it allows you to select in the Sales Agreement attribute.
- Gets values for several attributes in Contract Management, then displays them in the sales order.

Attribute in Contract Management	Attribute in Sales Order
Contract Number	Sales Agreement
Number on the Agreement Line	Sales Agreement Line
Version	Sales Agreement Version

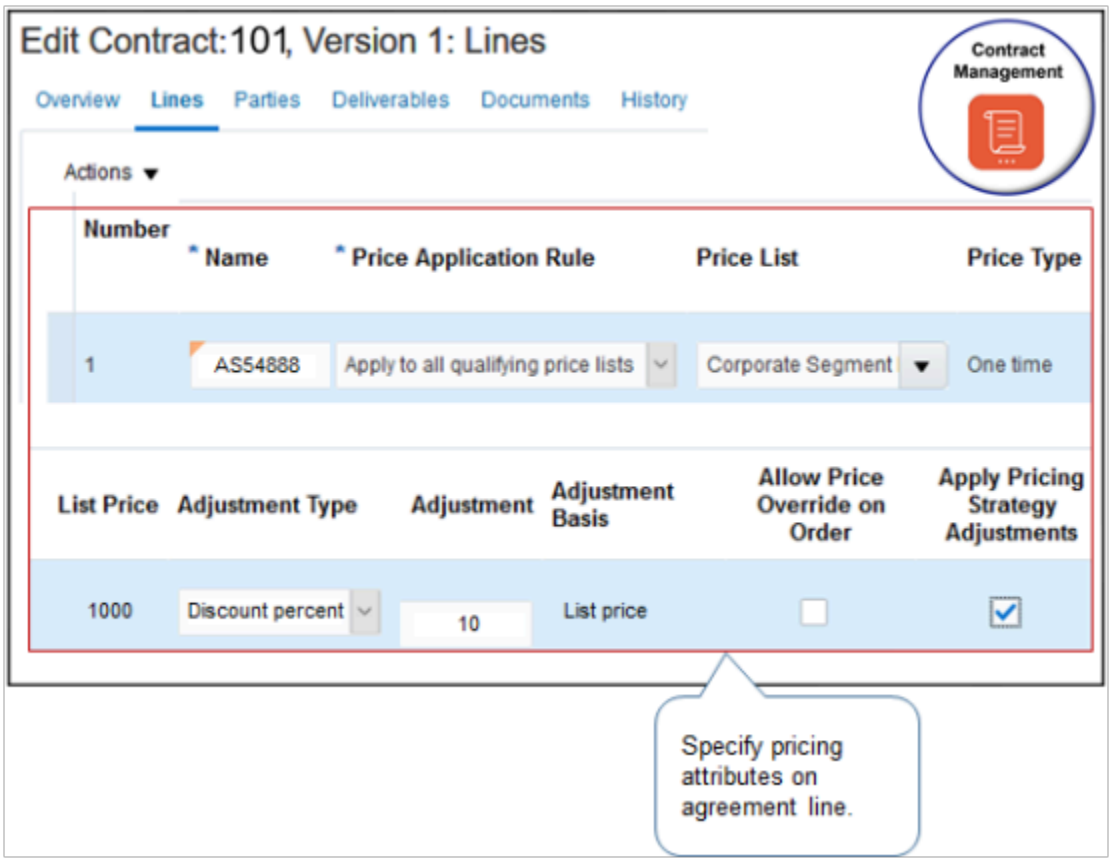
Attribute in Contract Management	Attribute in Sales Order
Start Date	Contract Start Date If you set Start Date on the agreement line in Contract Management, then Contract Start Date contains the agreement line date.
End Date	Contract End Date If you set End Date on the agreement line in Contract Management, then Contract End Date contains the agreement line date.

- If your Computer Service and Rentals customer orders a quantity of 10 or more of the AS54888 item, then Order Management applies a 10% discount on each order line.

Pricing calculates price for the item on the catalog line according to the price adjustment that you specify in the sales agreement. The Sales Agreement attribute on the order header specifies the agreement to apply.


Sales Order
Order Management
Catalog line.
Click.
2
Each
In Stock
Sale Price
900.00
1,800.00
Amount: Sale Price
Price Components
Unit Price
Amount
Base List Price Applied from Corporate Segment Price List
1,000.00
2,000.00
List Price
1,000.00
2,000.00
Adjustment 10.00 Percent for Sales Agreement 101
100.00
200.00
Rounding Rule Adjustment Applied from Precision Rounding...
0.00
0.00
Your Price
900.00
1,800.00
Sales agreement adjustment.
Done

Set Attributes That Affect Price



Note

Attribute	Value
Price Application Rule	Set to one of these values: • **Apply to Qualifying Price Lists**. Apply the price adjustment that you set up on the agreement regardless of whether the price list on the sales order matches the negotiated price list on the sales agreement. • **Enforce Price List and Apply Price**. Apply the price adjustment only if the price list on the sales order matches the negotiated price list on the sales agreement.
Price List Price Type List Price	Specify the price list to set the price for the item. For details, see *Manage Price Lists*.
Adjustment Type Adjustment Adjustment Basis	Specify the type of adjustment, such as discount amount or discount percent. For example, if the list price is $1000, and if you set Adjustment Type to Discount Percent, Adjustment to 10, and Adjustment Basis to List Price, then the adjustment equals one of these values: • $1000 list price multiplied by 10% equals 100 adjustment

Attribute	Value
	• $1000 list price minus a 100 adjustment equals an amount of 900 You can also create a tier adjustment according to the quantity or the extended amount. For an example that includes a value for each adjustment type, see *Add Tiers to Pricing Rules*.
Allow Price Override on Order	Allow your users to edit Your Price on the sales order.
Apply Pricing Strategy Adjustments	Apply adjustments that the pricing strategy calculates. For details, see *How Profiles, Segments, and Strategies Work Together*.

Integrate with Pricing Administration

Pricing uses the Apply Pricing Terms pricing algorithm to apply the pricing terms that it receives from Contracts.

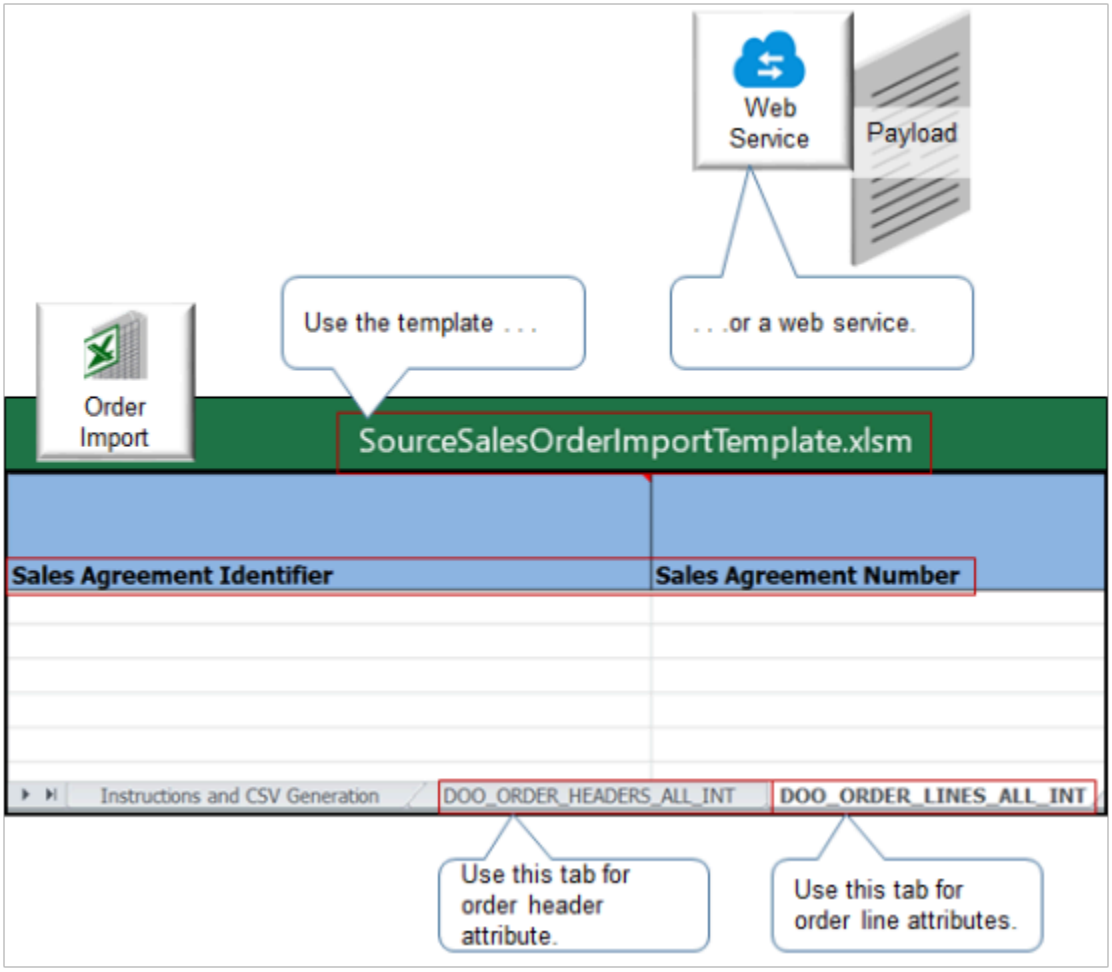
This algorithm evaluates and applies a list of pricing terms to each charge for the item. Here's what it does:

- Identify the charges it needs to apply the pricing term according to charge criteria.
- Get values for the adjustment basis, such as for percent.
- Apply the adjustment, such as apply a 10% discount.
- Apply an adjustment according to the promotion type, such as adjustment according to a promotion, the contract, or discount list.
- Adjust the running unit price.
- It the item is configured, then roll up charge components for pricing terms that apply to the configured item and configure options.

For an example that modifies pricing terms, see *Create Discounts That Accumulate or Cascade*.

Import and Integrate

Use different technologies to import or integrate agreements with a system that resides outside of Order Management.



Note

- You can use Oracle Application Development Framework (ADF), File-Based Data Import (FBDI), or REST API. You can also use electronic data interchange (EDI) in a Business-to-Business implementation.
- Use these technologies to create, edit, view, or revise sales orders.
- You must enable the Add Sales Agreements to Sales Orders opt-in feature to import or integrate.
- To learn about the attributes that you can import or integrate, see *Attributes You Can Use When Setting Up Sales Agreements in Order Management*.
- Use a POST operation with the Sales Orders for Order Hub REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.
- Use the Create Order operation or Stage Order operation with an ADF web service.

Set Up Configured Items

Contracts doesn't integrate with the Oracle Configurator, so you must set up each configure option individually on the contract line in the same way that you set up an item that isn't configured.

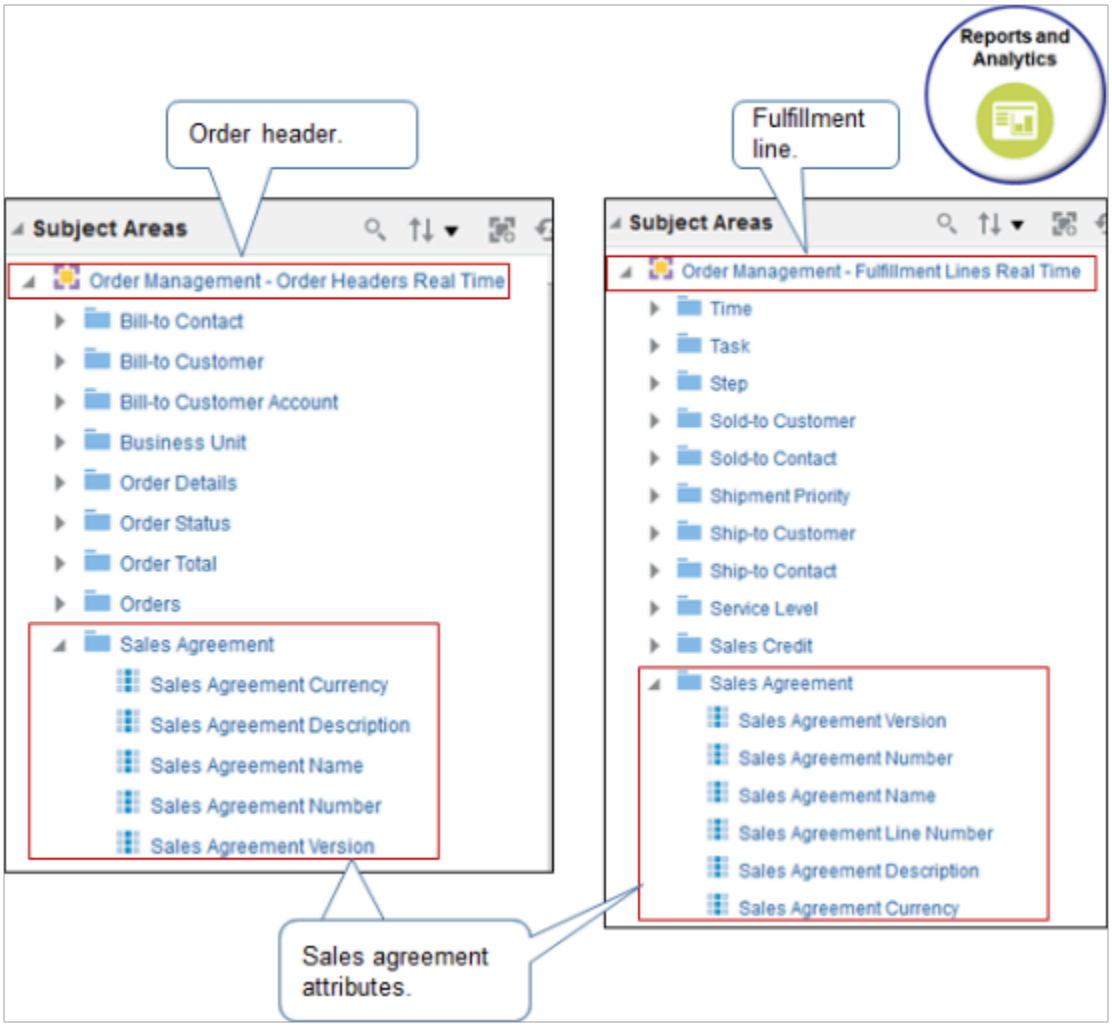
Pricing a configured item is similar to pricing an item that isn't configured. Assume you set the agreement on the order header to 101. If you search for a configured item on the catalog line, and then click:

- **Configure and Add**. Order Management displays the configurator, and Pricing uses agreement 101 to price each configure option.

  If sometime later, you click View Components on the order line, then, Order Management sets the agreement for each configure option to 101 in a dialog. If you change this agreement to 102, then Pricing uses agreement 102 to price the option.
- **Add**. Order Management adds an order line, sets the order line's agreement to 101, and prices the item and its configure options according to agreement 101.

  If you change the order line's agreement to 102, click Edit on the line and edit the configured item, then Pricing will use agreement 102 to price the configure options.

Report



You can use the Reports and Analytics work area to get the data that includes the sales agreement's attributes.

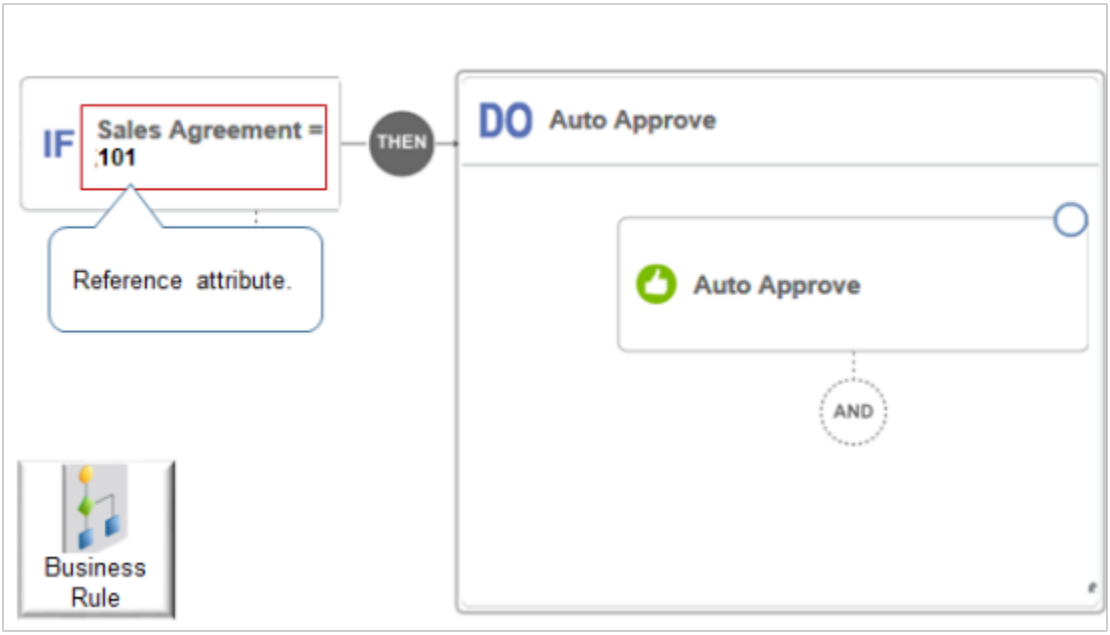
Expand one of these subject areas:

- Order Management Order Headers Real Time
- Order Management Fulfillment Lines Real Time

Next, expand Sales Agreement, and then examine the attributes that you can use in your report.

For details, see *Use Reports and Analytics with Order Management*.

Other Set Ups



You can reference the Sales Agreement attribute in a processing constraint, pretransformation rule, approval rule, or order management extension.

What You Can't Do

You can't:

- Reference more than one agreement from one order line.
- Reference an order line agreement for a coverage item.
- Use a party other than the Customer sold-to party on the order header to get agreements.
- Set up an agreement adjustment for a configure option from the Edit Contracts page.

  Use a parent customer's agreement with a child customer. Assume you set up your party data so customer x has a child customer y. You can set up a sales agreement only with x, not with y. If you need an agreement with y, then you must make y a primary customer in your party data first, and then set up an agreement with y.

Track Agreed Amount and Committed Quantity Order Management doesn't track or limit the quantity or amount that it consumes according to the Agreed Amount attribute or the Committed Quantity attribute on the contract line. Assume you go to the Contracts work area, open a contract for editing, click Lines, and then set Committed Quantity to 1000. At run time, Order Management consumes a quantity of 1100 for sales order x. Order Management will apply your

price adjustment on the full quantity of 1100 and on any subsequent quantity that it consumes for the current or future sales order.

Related Topics

- Use Reports and Analytics with Order Management
- Manage Price Lists
- Add Tiers to Pricing Rules
- How Profiles, Segments, and Strategies Work Together
- Create Discounts That Accumulate or Cascade

Set Up Sales Agreements in Order Management

Set up contracts so your users can add an agreement to a sales order.

Summary of the Set Up

1. Create the contract type.
2. Create the contract.
3. Set up Order Management.

Assume you're in the Vision Operations business unit, and you need to set up a contract with your Computer Service and Rentals customer. If your customer orders a quantity of 10 or more of the AS54888 item, then give them a 10% discount.

This topic uses example values. You might need different values, depending on your business requirements.

For background details, go to *Using Customer Contracts*, then search for Overview of Enterprise Contracts, or search for Create a Sales Agreement Line.

Create the Contract Type

1. Sign into Oracle Enterprise Contracts with the privileges that you need to create contracts.
2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Enterprise Contracts
    - Functional Area: Enterprise Contracts Base
    - Task: Manage Contract Types
3. On the Manage Contract Types page, click **Actions > Create**
4. In the dialog that displays, set the values, then click **Continue**.

Attribute	Value
Class	Agreement You must use this value.
Set	Common Set You can use any value.

Attribute	Value
Name	My Agreements You can use any value.
Intent	Sell You must use this value.
Allow Lines	Contains a check mark.
Pricing Integration	Oracle Order Management Pricing You must use this value.

5. On the Edit Contract Type page, set the values, then click **Save and Close**.

Attribute	Value
Buyer Role	Customer
Seller Role	Supplier
Contract Owner Role	Salesperson

You can set these attributes to any value.

Create the Contract

1. Go to the Contracts work area.
2. On the Contracts page, click **Actions > Create**.
3. In the Create Contract dialog, set the values, then click **Save and Continue**.

Attribute	Value
Business Unit	Vision Operations
Legal Entry	Vision Operations
Type	My Agreements
Number	101 This value will display in the Sales Agreement attribute throughout the Order Management work area.

Attribute	Value
Primary Party	Computer Service and Rentals
Start Date	1/01/2019
End Date	12/31/2019 Leave empty to specify a contract that never ends.
Currency	USD
Item Master	Vision Operations
Contract Class	Agreement
Intent	Sell

4. On the Edit Contract page, click **Lines > Actions > Add**, set the values, then click **Submit**.

Attribute	Value
Type	Product
Name	AS54888
UOM	Each
Adjustment Type	Discount Percent
Adjustment	10
Allow Price Override on Order	Enabled
Minimum Quantity for Order	10
Start Date	1/1/22

Attribute	Value
End Date	12/31/22

If you create more than one contract line for the same item, then make sure you use a different UOM on each line. For example, this is OK:

Contract Line	Name	UOM
1	AS54888	Each
2	AS54888	Dozen

Don't use the same UOM on different lines with the same item. For example, don't do this:

Contract Line	Name	UOM
1	AS54888	Each
2	AS54888	Each

If you use the same UOM on different lines with the same item, then Order Management will randomly select the line, and it might not be the line that you need.

Set Up Order Management

1. Make sure you have the privileges that you need to administer Order Management.
2. Enable the Add Sales Agreements to Sales Orders opt-in feature.

   If you enable, then Order Management displays sales agreement attributes throughout the Order Management work area. For details about how to enable, see *Opt Into Features in Order Management*.
3. Set the Automatically Set Values on Sales Agreement Attributes parameter.

   For details, see *Manage Order Management Parameters*.
4. Promote pricing algorithms.
   - Make sure you have the privileges that you need to administer pricing.
   - Go to the Pricing Administration work area, then click **Tasks > Manage Algorithms**.
   - On the Manage Algorithms page, click **Actions > Promote All**.

     If you extended a pricing algorithm in an earlier release, then you must also create a new version and reconcile changes. For details, see *Promote Pricing Algorithms Into the Latest Update*.

Related Topics

- Opt Into Features in Order Management
- Manage Order Management Parameters
- Overview of Enterprise Contracts
- Create a Sales Agreement Line
- Promote Pricing Algorithms Into the Latest Update

Attributes You Can Use When Setting Up Sales Agreements in Order Management

Get details about attributes you can use when you use a web service to communicate or a file to import sales agreement details.

You can include these attributes in the PricingTerm entity of the Price Sales Transaction service data object when you use the Pricing Administration work area. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

Include these pricing attributes in your payload.

Attribute	Data Type	Description
ApplyToEntityCode	string	Abbreviation that identifies the entity where Pricing applies the pricing term. Value is LINE. This value references the Order FLine attribute in Order Management.
ApplyToEntityId	Long	Entity where Pricing applies the pricing term. This value references the FLine Id attribute in Order Management.
PricingTermId	Long	Identifies entity PricingTermSDO.
SourceIdentifierId1	string	Optional. Identifies the entity for a pricing term. For example, Sales Agreement Header Id or Promotion Header Id. Order Management sends the Contract Id attribute.
SourceIdentifierId2	string	Optional. Identifies the entity for a pricing term. For example, Sales Agreement Line Id or Promotion Line Id.

Attribute	Data Type	Description
		Order Management sends the Contract Line Id attribute.
SourceIdentifierId3	string	Optional. Identifies the entity for a pricing term. For example, Sales Agreement Major Version Id. Order Management sends attribute Contract Major Version.
SourceIdentifierId4	string	Optional. Identifies the entity for a pricing term. For example, Sales Agreement Status.
SourceIdentifierId5	string	Optional. Identifies the entity for a pricing term. For example, to support a key that includes more than one part.
SourceName	string	Value that uniquely identifies the source of an entity for a pricing term. For example, Sales Agreement or Promotions.
SourceTypeCode	string	Value that uniquely identifies the source code of an entity for a pricing term. For example, SALES_AGREEMENT or PROMOTIONS.
TermId	long	Identifies the term in the sales agreement.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together

Trade Compliance

Overview of Setting Up Trade Compliance

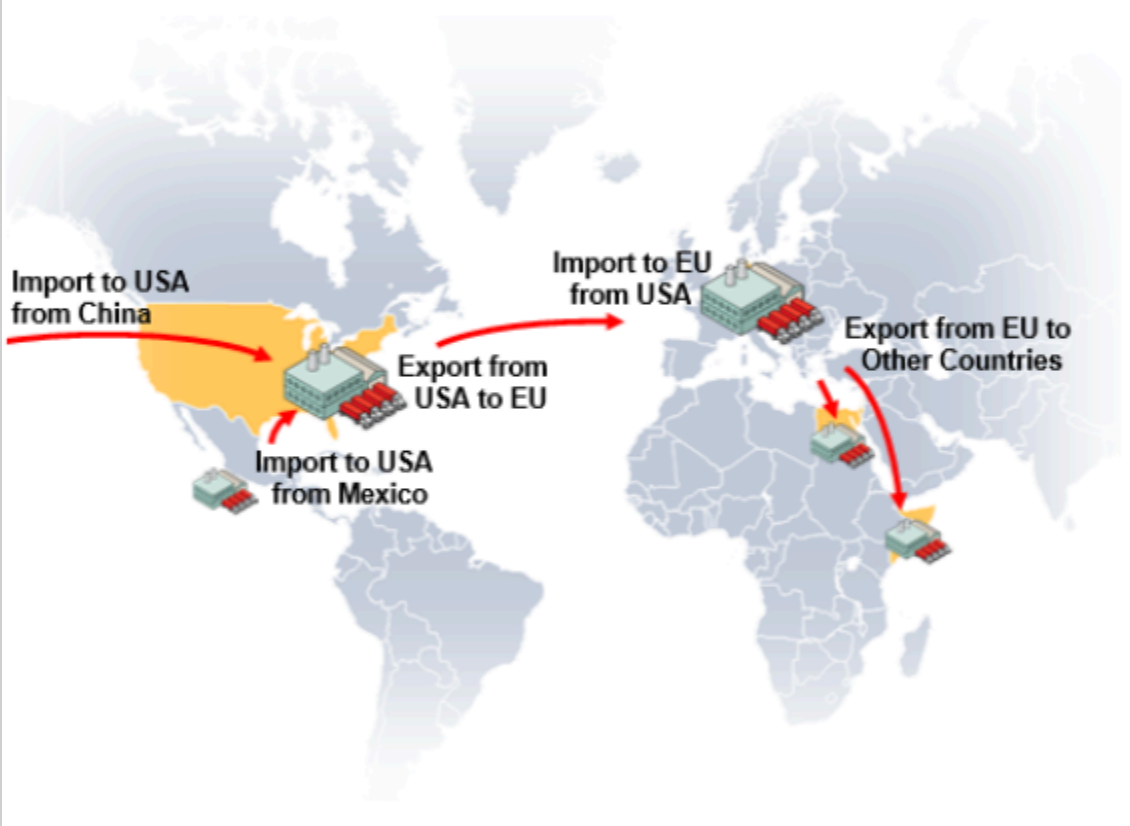
Use Order Management with your trade management solution to create and manage trade transactions that comply with global import trade rules, export global trade rules, and other trade regulations that the country or region requires.

Some governments and companies impose rules and regulations on trade with different countries, people, companies, financial institutions, and so on. Use trade compliance to meet these rules.

- Trade compliance is a structure of rules that makes sure trade between countries or regions happens only according to the approved laws and guidelines that these countries or regions use.

- Manage legal, regulatory, and corporate requirements for each transaction that crosses a government border, such as across states, provinces, regions, or countries, according to the unique requirements of each country, region, company, and so on.
  For example, set up trade controls that apply United States rules and Chinese rules on each transaction that happens between a company that resides in the United States and another one that resides in China.
- Apply trade controls on various items, such as munitions, computer equipment, licenses, license exceptions, documents, registrations, and so on.
- Manage trade compliance policy, hold transactions until they clear trade compliance screening, and so on.
- Screen each sales order for restricted parties and sanctioned countries when submitting the sales order in Order Management.
- Don't screen sales order at submit, but do screen before shipping.
- Screen each sale order at different points during a long fulfillment cycle.

Here's an example flow that illustrates how you can use your trade management solution with Order Management.



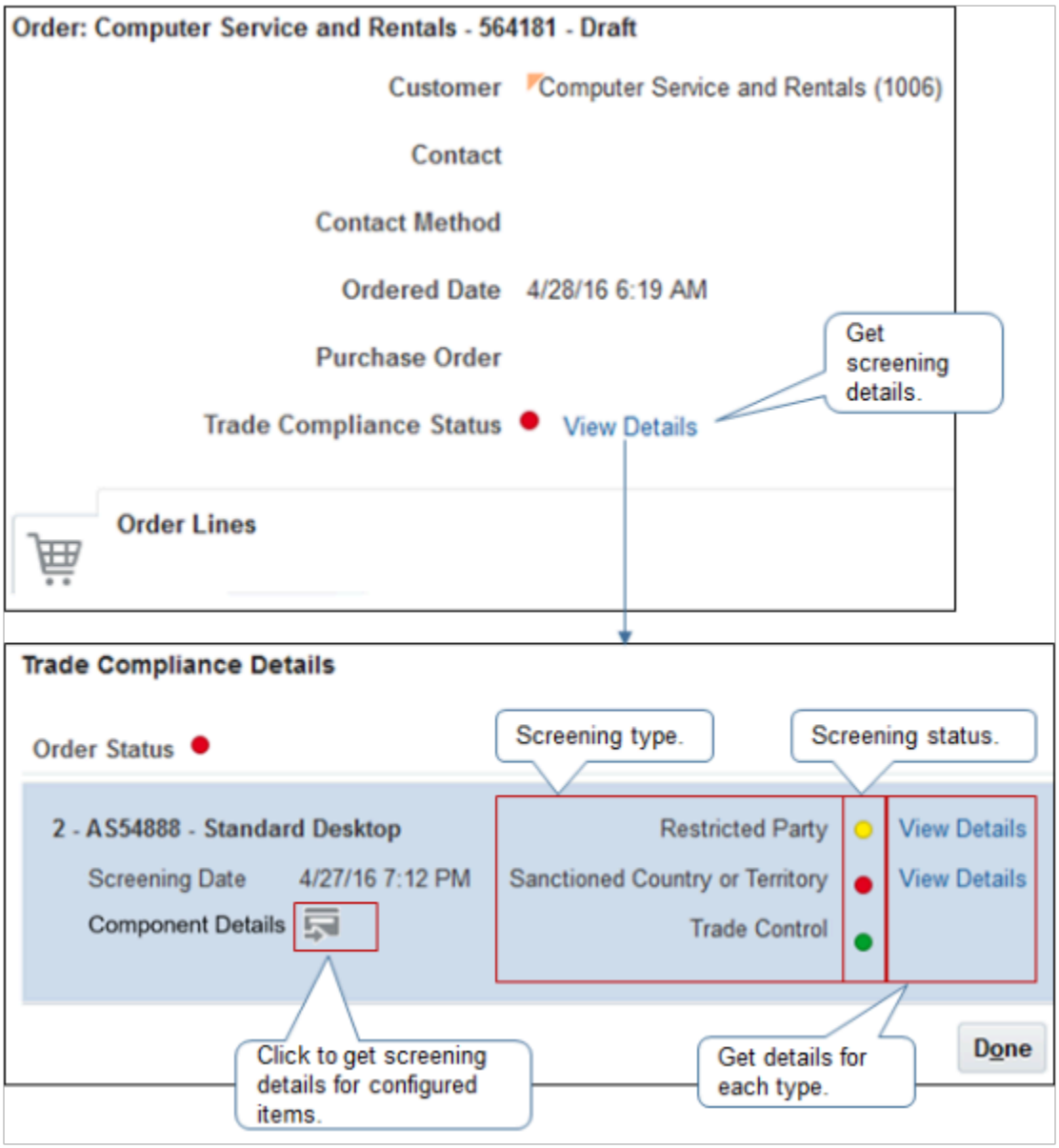
This supply chain imports raw materials from more than one source into a factory in the United States, sends a partially finished assembly to a factory in Europe, then sends the final assembly to distribution centers in more than one country in Africa.

Each import and export point might require a different set of trade compliance rules for each transaction. For example, you can manage compliance according to trade compliance policies.

- Product classification, such as weapons or dangerous chemicals
- Export and import embargo
- Status on the Denied and Restricted Parties List

- Trade agreement, such as NAFTA (North American Free Trade Agreement)
- Restricted party

Your users can manage compliance in the Order Management work area.



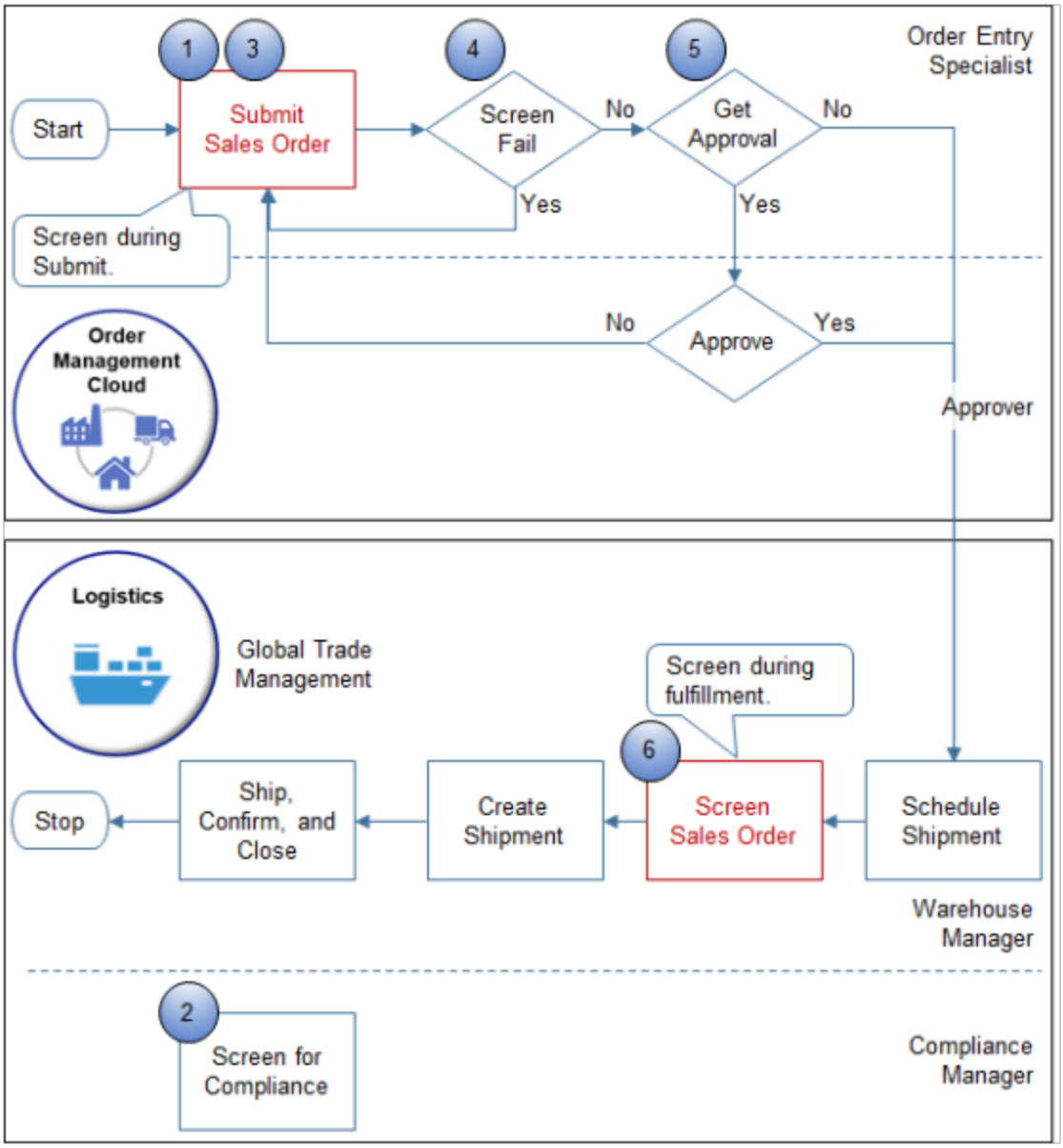
Related Topics

- How Trade Compliance Works in Order Management
- Set Up Trade Compliance in Order Management
- Set Up Trade Compliance so it Happens During Order Fulfillment
- Set Up Trade Control Types
- Manage Sales Orders That Require Trade Compliance Screening

How Trade Compliance Works in Order Management

Managing a sales order for trade compliance includes privileges from more than one job role, such as Order Entry Specialist, Order Manager, Compliance Manager, and Warehouse Manager.

Here's an example where Order Management applies trade compliance during screening.



Note

1. An Order Entry Specialist creates a sales order in the Order Management work area, then clicks Submit. Order Management validates the sales order in the same way it validates any sales order, then sends it to Global Trade Management for screening.
    - If you set the Check for Trade Compliance When User Submits Sales Order parameter to Yes, then Order Management sends a request to Global Trade Management to screen the sales order for trade compliance. For details, see *Manage Order Management Parameters*.
    - Global Trade Management can screen an item that isn't configured, a configured item, or a change order.
    - You can screen a source order that you import form a source system.
    - You can screen only for restricted part or sanctioned territory during order submit.
    - This integration can't screen a return line.
2. Global Trade Management screens the sales order.
    - Apply trade compliance policies that you set up for this integration to the sales order, then create a screening result. Global Trade Management supports a variety of compliance screening types. For example, restricted party, sanctioned countries or territories, and trade control.
    - Set the trade compliance status to Passed, Under Review, or Failed, according to the screening result. You can't modify this value.
    - If the trade compliance status is Failed or Under Review, then provide the screening failure reason for each order line that fails screening.
    - Send the screening result to Order Management. You can use the screening result in a constraint or approval rule.
3. Order Management updates the trade compliance status on the sales order according to the most restrictive trade compliance status that applies on the order lines in the sales order.

    For example, if status is Failed for one order line, then the sales order status is Failed.

    Here's the hierarchy that Order Management uses to determine which status is most restrictive, where 1 is least restrictive, and 3 is most restrictive.

    1. Passed

    2. Under Review

    3. Failed

    Note

    - Global Trade Management typically finishes screening and sends the result to Order Management without delay. However, Global Trade Management might require a few minutes to screen a large or complex sales order.
    - If the sales order hasn't moved to order fulfillment, then the Order Entry Specialist can click Actions, then click Revert to Draft. Order Management will set order status to Draft and stop screening.
4. If screening fails, then Order Management sends the sales order back to the Order Entry Specialist because a predefined processing constraint prevents Order Management from submitting a sales order that includes a trade compliance exception. The Order Entry Specialist can modify or cancel the sales order.

    You can disable the constraint or create a new one that allows the sales order to proceed according to a condition. For details, see *Constrain Trade Compliance Screening*.
5. If screening doesn't fail, then Order Management sends the sales order to order fulfillment.

If the constraint allows the sales order to proceed according to a condition, then, as an option, you can set up an approval rule that uses the trade compliance result of the order header or order line to route the sales order to an Order Manager for approval.

6. As an option, you can also screen during order fulfillment according to an orchestration process step that you set up.

This flow uses Oracle Global Trade Management as the trade compliance solution. You can use your own solution. As an option, it identifies the points where you can screen for trade compliance during order fulfillment in Oracle Shipping. For details about Global Trade Management and how to set it up, see Oracle SCM Solutions, Global Trade Management at https://www.oracle.com/applications/supply-chain-management/solutions/logistics/global-trade-management.html.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

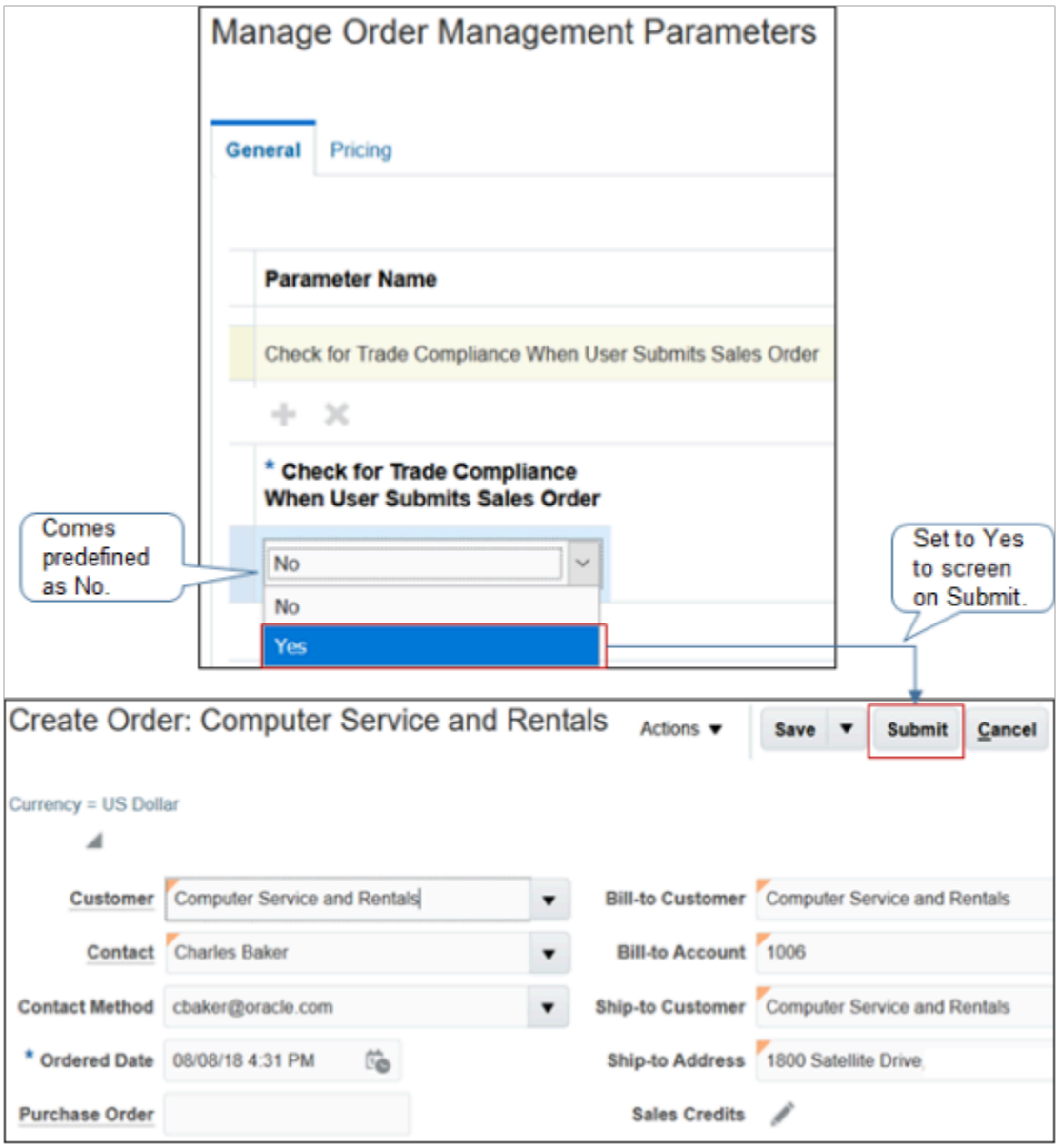
Related Topics

- Overview of Setting Up Trade Compliance
- Constrain Trade Compliance Screening
- Set Up Trade Compliance in Order Management
- Set Up Trade Compliance so it Happens During Order Fulfillment
- Overview of Setting Up Approval

Guidelines for Setting Up Trade Compliance

Apply guidelines when you set up trade compliance screening in Order Management.

Enable the Order Management Parameter

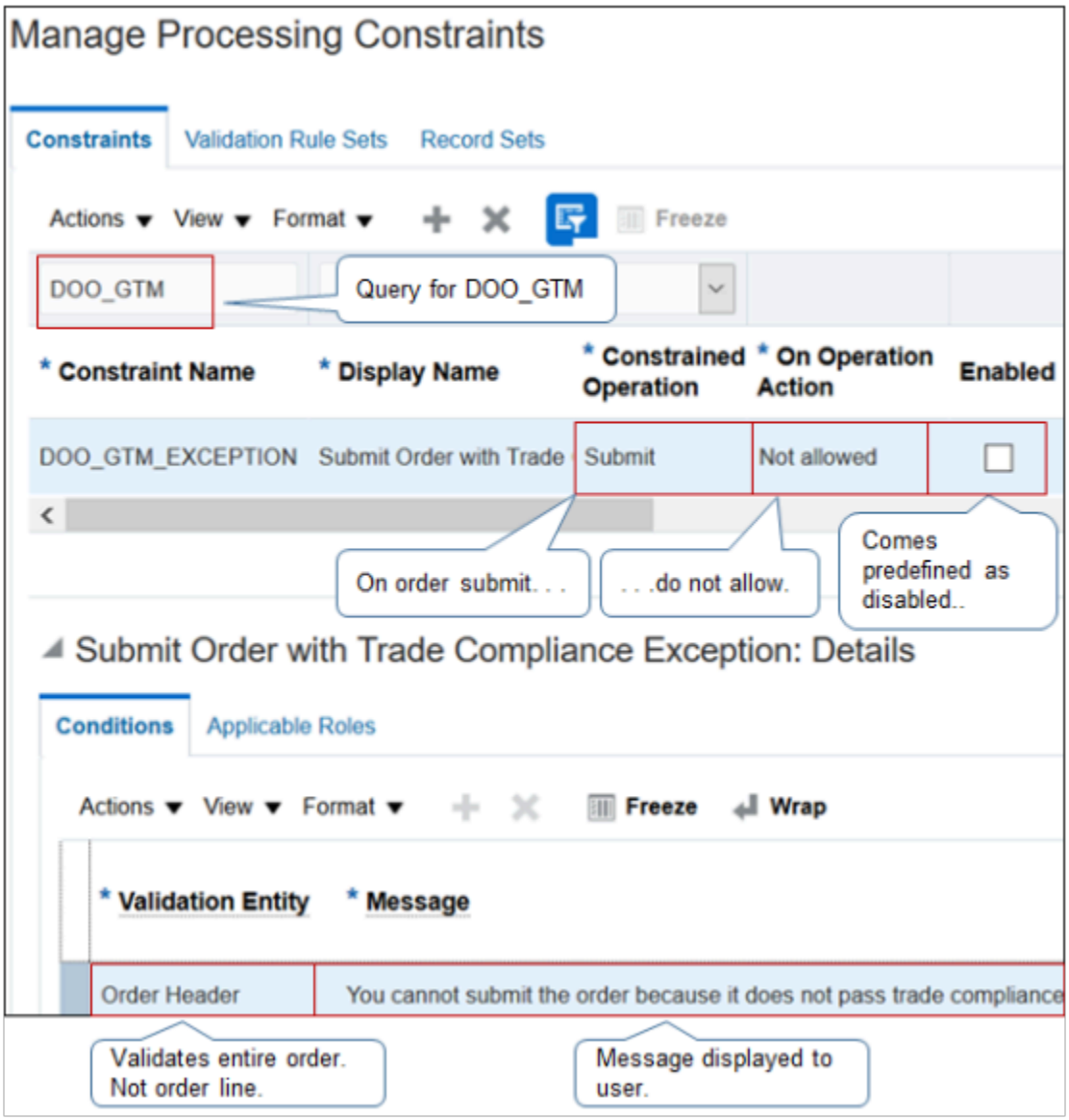


Note

- Use the Manage Order Management Parameters page in the Setup and Maintenance work area.
- As an option, to screen at order submit, set the Check for Trade Compliance When User Submits Sales Order parameter to Yes.
- This parameter comes predefined as No.

For details, see *Manage Order Management Parameters*.

Constrain Changes



Note

- Use the Manage Processing Constraints page in the Setup and Maintenance work area.
- As an option, use the DOO_GTM_EXCEPTION predefined processing constraint to implement a condition.

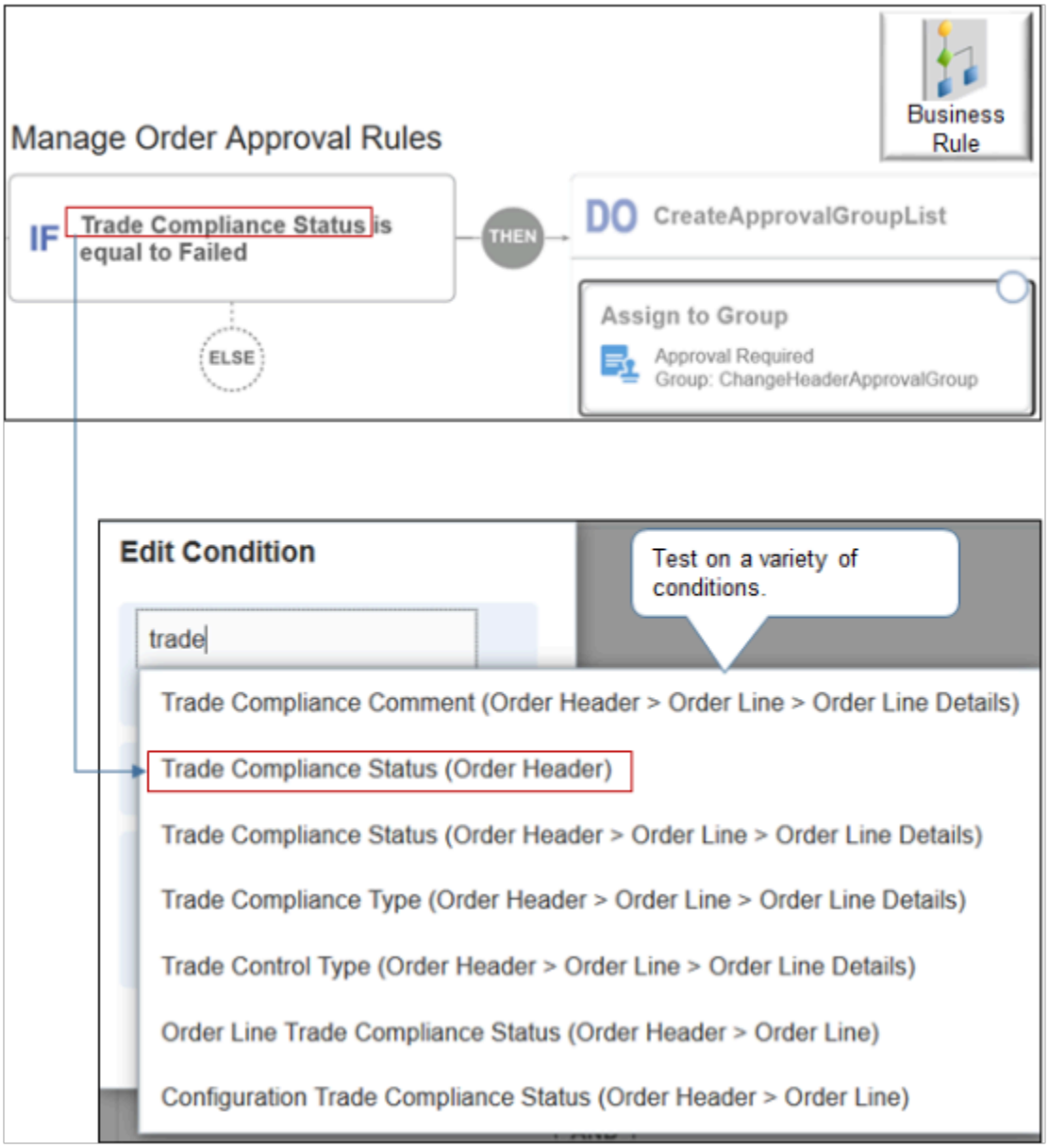
  ```
  if compliance status is not ORA_PASSED, then prevent submit on order header.
  ```

- Applies when compliance is under review or when compliance screening fails.
- Comes predefined as disabled.
- Applies to the order header. Rejects the entire sales order. Doesn't reject individual order lines.
- Displays a message in the Order Management work area when your user clicks Submit.

- You can't modify this constraint. If it doesn't meet your needs, then create your own constraint. For example, to apply a constraint only with a specific privilege.

For details, see *Manage Processing Constraints*.

Manage Order Approval



Note

- Use the Manage Order Approval Rules page in the Setup and Maintenance work area to create an approval rule.
- Approval comes predefined to display the screening result in the approver's Pending Notifications page or worklist.

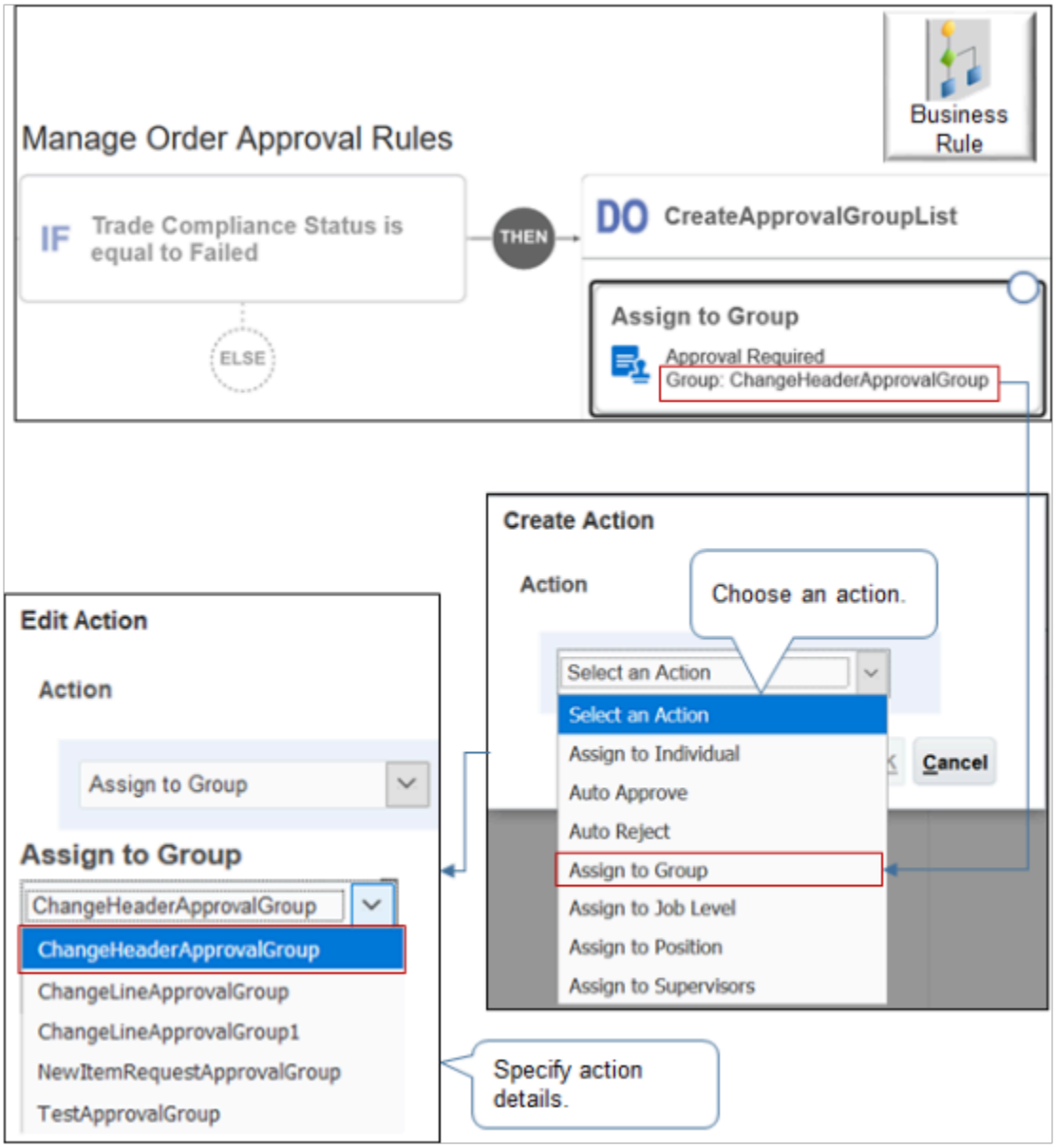
- As an option, you can create an approval rule to request review and approval for each sales order that doesn't pass screening.

  For example, if the sales order fails screening, then Order Management rejects the submit and sets the order status to Draft. You can create an approval rule that allows the sales order to proceed to order fulfillment while compliance is under review.

  ```
  If compliance check is under review, then allow order submit.
  ```

- Test on a variety of conditions, such as Trade Compliance Status of the order header, Trade Compliance Comment on an order line, and so on.

Select an action.



Note

- Select an action, such as Assign to Group, Assign to Individual, or Auto Approve.
- Specify details about the action. For example, if you select Assign to Group, then you can specify the group to assign.

For details, see *Guidelines for Setting Up Your Approval Rule*.

Integrate Order Management with Trade Management

You use the same set up that you use to integrate with Transportation Management, but with some important differences.

When Screening Happens	Description
At order submit	You must use Integration Cloud Service, and you must set the Invocation Mode attribute on the connector to Business Event.
During order fulfillment	You can use some other integration service, and you can set Invocation Mode to Asynchronous Service or Synchronous Service. However, you must set up your own integration and connector. You can't use the example integrations that Integration Cloud Service provides as a starting point.

For details, see *Guidelines for Integrating Order Management with Transportation Management*.

Use Your Own Integration Service

If you use your own integration service instead of Integration Cloud Service, then make sure the payload that your integration sends to Order Management includes the required details.

Note

- Use TradeComplianceScreeningResultCode to provide the result for each request.
- Use TradeComplianceScreeningResultCode to provide the result for each fulfillment line.
- Use FulfillmentDetail to provide separate result details for each compliance type. For example, provide a separate FulfillmentDetail for each type.
    - ORA_SANCTIONED_TERRITORY
    - ORA_RESTRICTED_PARTY
    - ORA_TRADE_CONTROL

- Set TradeComplianceScreeningResultCode of the overall result to the most restrictive value from the FulfillmentDetail sections.

  For example:
  - If
    - ORA_SANCTIONED_TERRITORY is ORA_PASSED.
    - ORA_RESTRICTED_PARTY is ORA_PASSED.
    - ORA_TRADE_CONTROL is ORA_FAILED.
  - Then
    - Set TradeComplianceScreeningResultCode of the overall result to ORA_FAILED.

  In this example, FulfillmentDetail is ORA_PASSED for all compliance types, so TradeComplianceScreeningResultCode is ORA_PASSED.
- If a compliance check happens more than one time for a compliance type, then make sure the payload includes more than one FulfillmentDetail for the compliance type.

  For example, if compliance checks ORA_SANCTIONED_TERRITORY for the ship to customer and for the bill to customer, then make sure the payload includes one ORA_SANCTIONED_TERRITORY FulfillmentDetail for the ship to customer, and another ORA_SANCTIONED_TERRITORY FulfillmentDetail for the bill to customer.

Structure Your FulfillmentDetail

You must include:

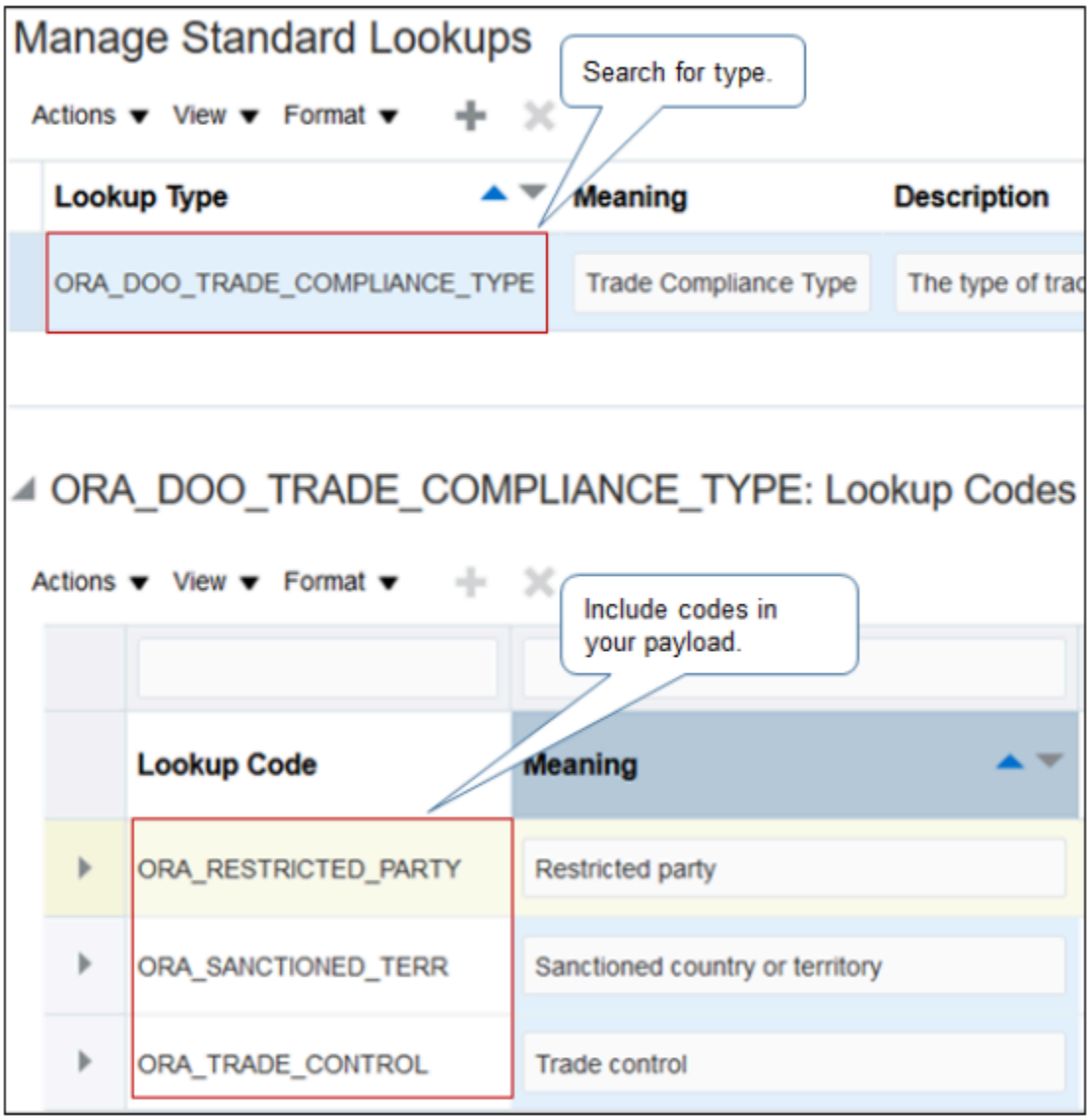
- One TradeComplianceTypeCode for each FulfillmentDetail.
- One TradeComplianceScreeningResultCode for each FulfillmentDetail.

Here are the optional attributes.

- Include one or more comments. For example:
    - If compliance type is Restricted Party, then use a comment to identify the customer on the restricted party list.
    - If compliance type is Sanctioned Country or Territory, then use a comment to identify the country or territory.

  In this example, the screening violation happens in the International Traffic in Arms Regulations trade agreement.
- Include one Trade Control Type for each FulfillmentDetail. Its applicable only for these compliance types.
    - Trade Control
    - Sanctioned Country or Territory

Use Lookup Codes for Your Compliance Types



Search for this predefined lookup on the Manage Standard Lookups page to verify the codes that your implementation uses.

Lookup Type	Meaning
ORA_DOO_TRADE_COMPLIANCE_TYPE	Trade Compliance Type

Note

- You can't modify this lookup or its codes.

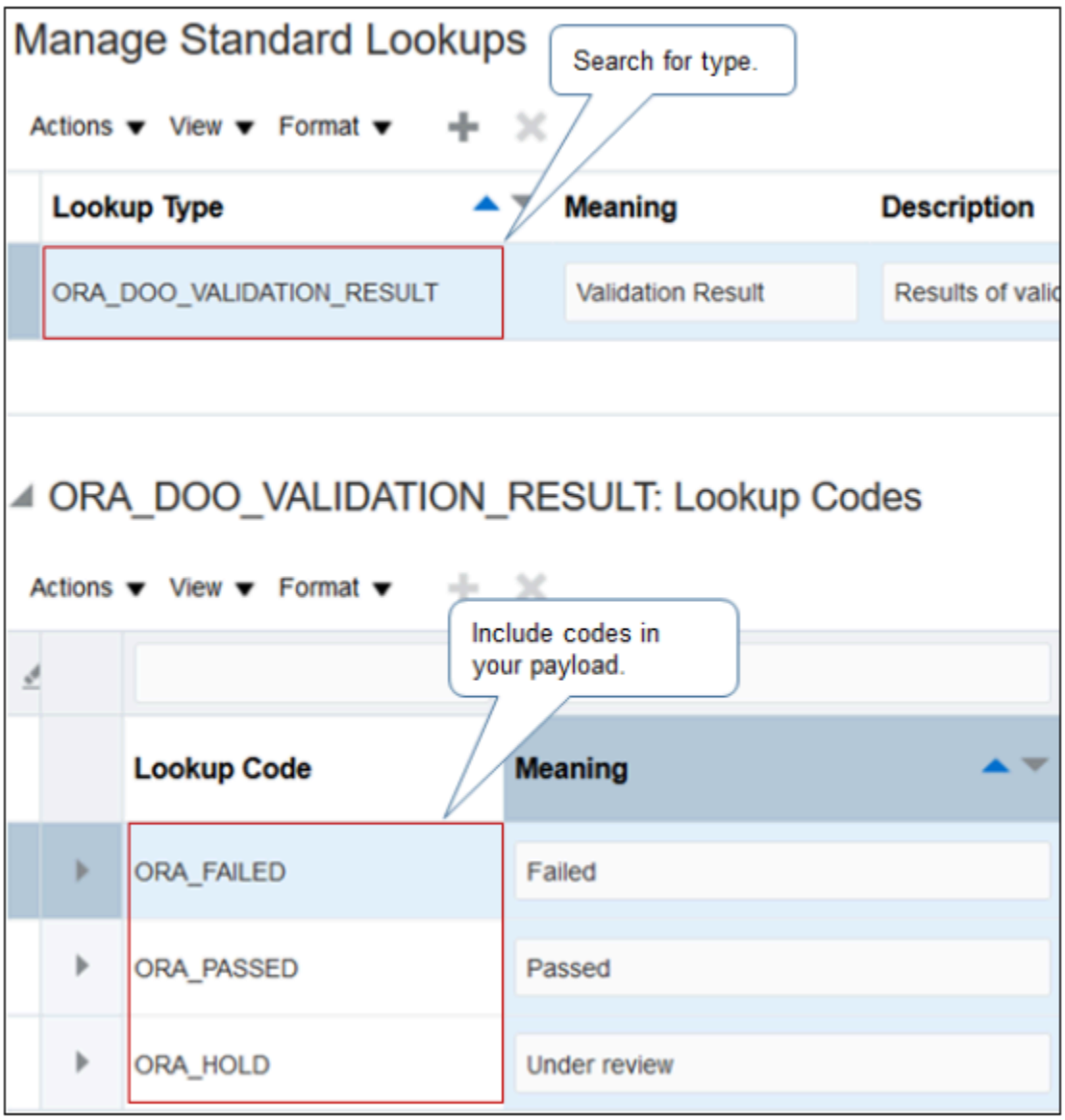
- Make sure your response payload includes each lookup code.

  At a minimum, your payload must include these codes.

  - ORA_RESTRICTED_PARTY
  - ORA_SANCTIONED_TERRITORY
  - ORA_TRADE_CONTROL

For details, see *Manage Lookups in Order Management* and *Set Up Trade Control Types*.

Include Lookup Codes for the Validation Result



Search for this predefined lookup type on the Manage Standard Lookups page.

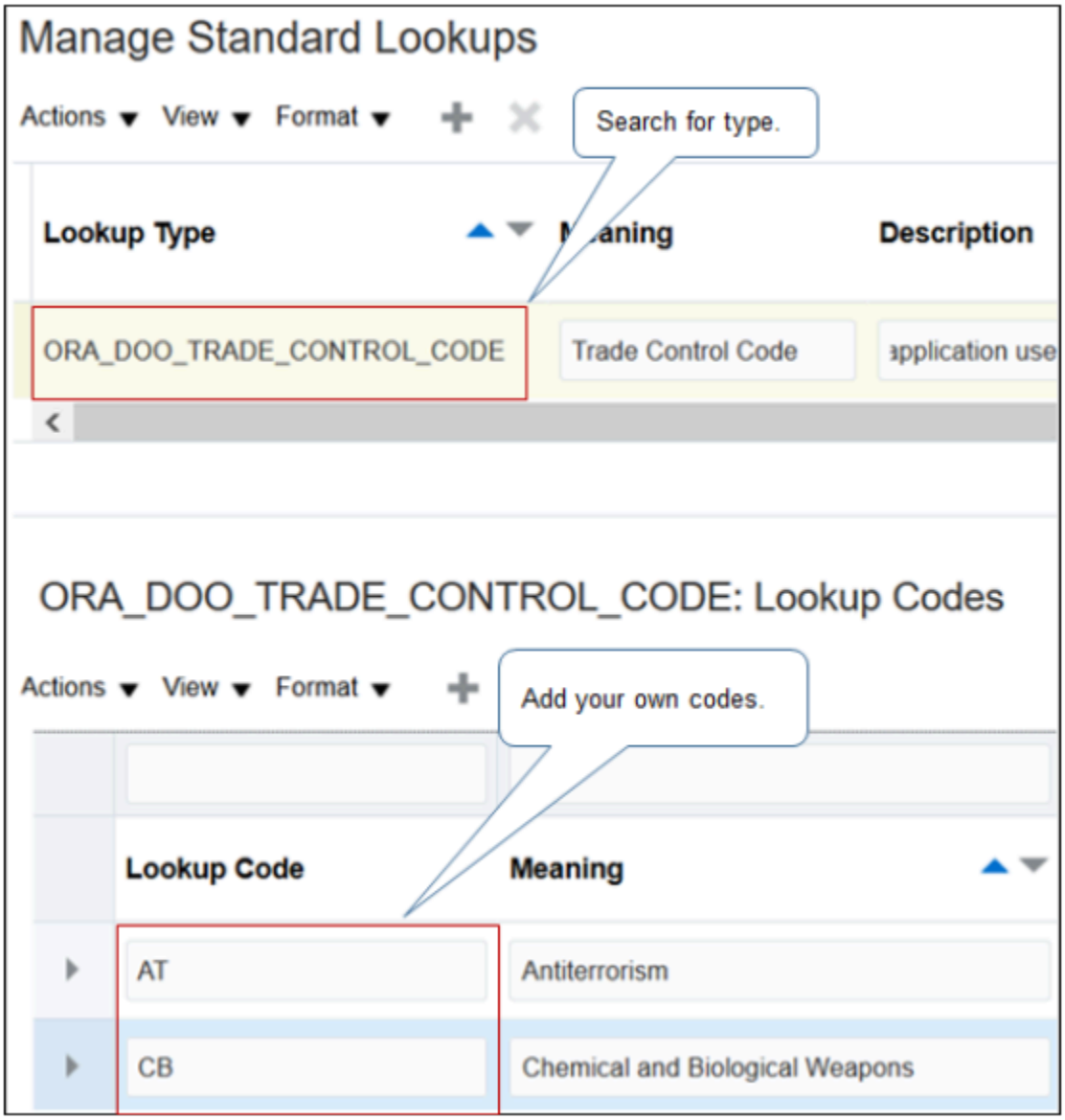
Lookup Type	Meaning
ORA_DOO_VALIDATION_RESULT	Validation Result

Note

- You can't modify this lookup or its codes.
- Make sure your response payload includes each lookup code. At a minimum, your payload must include these codes.
    - ORA_FAILED
    - ORA_PASSED
    - ORA_HOLD

Include Lookup Codes for Trade Control

The trade control type clarifies the government policy, document, agreement, and so on, that Trade Management uses when it verifies trade compliance. Export Administration Regulations and Atomic Energy Act is an example of a government policy.



Search for this predefined lookup on the Manage Standard Lookups page.

Lookup Type	Meaning
ORA_DOO_TRADE_CONTROL_CODE	Trade Control Code

This lookup type comes predefined without lookup codes. You can add your own codes. Here are the codes that we added in this example.

Lookup Code	Meaning
AT	Antiterrorism
CB	Chemical and Biological Weapons

Related Topics

- Overview of Setting Up Trade Compliance
- Manage Order Management Parameters
- Use Integration Cloud Service with Order Management
- Constrain Trade Compliance Screening
- Overview of Setting Up Approval

Set Up Trade Compliance in Order Management

Order Management comes predefined with trade compliance screening disabled. You can enable it.

Learn how to set up trade compliance screening to happen during order fulfillment. For details, see *Set Up Trade Compliance so it Happens During Order Fulfillment*.

Set up trade compliance screening so it happens before sending the sales order to order fulfillment.

1. Set up an integration between Order Management and your trade management system.

   You must use Integration Cloud Service.
2. Set the Check for Trade Compliance When User Submits Sales Order parameter to Yes. For details, see *Manage Order Management Parameters*.
3. Optional. Do more set ups, according to your business requirements.
   - Constrain compliance screening.
   - Require sales order approval.

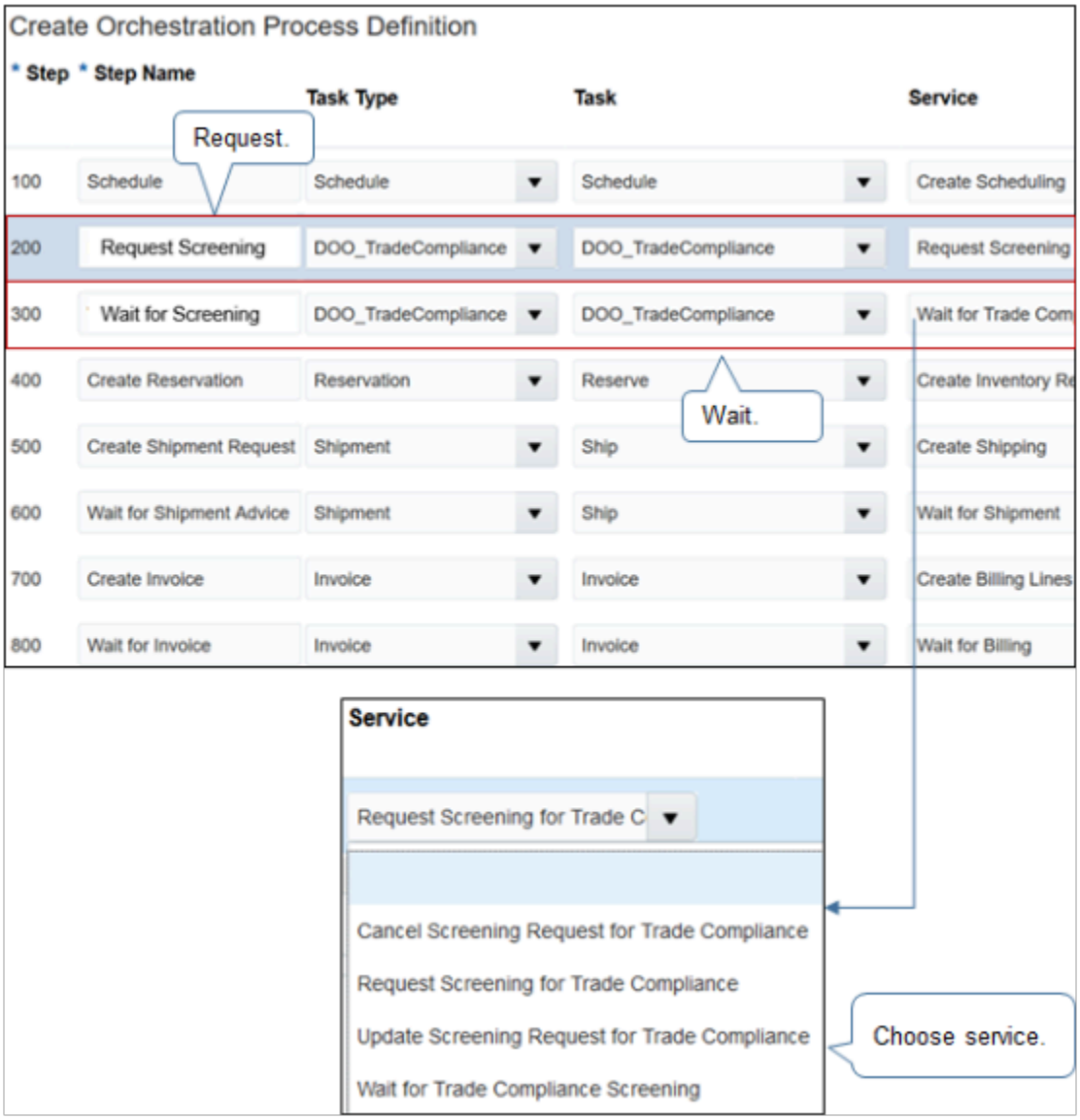
Related Topics

- Overview of Setting Up Trade Compliance
- Manage Order Management Parameters
- Use Integration Cloud Service with Order Management
- Constrain Trade Compliance Screening
- Overview of Setting Up Approval

Set Up Trade Compliance so it Happens During Order Fulfillment

Set up an orchestration process so it screens order lines for trade compliance during order fulfillment.

In this example, you create a duplicate of a predefined orchestration process, then add steps to the duplicate.



Note

- You must add a request step and a wait step.
    - The request step sends a request to Global Trade Management to screen the sales order.
    - The wait step pauses the orchestration process while it waits for the reply from Global Trade Management. The pause makes sure fulfillment doesn't finish without first clearing trade compliance check. The wait step also processes the response depending on the compliance status that Global Trade Management sends.
- Set up exit criteria for the wait step according to compliance status, and according to your business needs.
- As an option, add a pause step at some point after the wait step to allow for manual intervention. For example, assume you prefer to allow fulfillment to continue while an approver investigates the reason for screening failure, right up to the point of shipping the item. You can add a pause step immediately before Create Shipment Request.

- Add the request and wait steps at any point in the orchestration process. For example, screen before creating the reservation, screen before creating the shipment request, screen after creating the invoice, and so on.
- Use any text for the Step Name attribute.
- You must use the predefined value DOO_TradeCompliance for the Task Type attribute and the Task attribute.
- Select from a variety of services, such as Request Screening for Trade Compliance or Wait for Trade Compliance Screening.

This topic uses example values. You might need different values, depending on your business requirements.

Set up trade compliance so it happens during order fulfillment.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, search for the orchestration process.

Attribute	Value
Process Name	ShipOrderGenericProcess

    For details, see *Set Up Orchestration Processes*.
3. In the Search Results, click **Actions > Duplicate**.
4. On the Edit Orchestration Process Definitions page, set the values, accept all other default values, then click **Save**.

Attribute	Value
Process Name	Orchestration_Process_for_Compliance_Screening You can enter any value.
Process Display Name	Orchestration Process for Compliance Screening
Description	Screen sales order for trade compliance, then process it through order fulfillment.

5. In the Process Details area, create a step.

Attribute	Value
Step Name	Request Trade Compliance Screening You can use any text for the name.

Attribute	Value
Step Type	Service
Task Type	DOO_TradeCompliance
Task	DOO_TradeCompliance
Service	Request Screening for Trade Compliance
Exit Criteria	Leave empty.
Default Lead Time	1
Lead-Time UOM	Hour Set attribute Default Lead Time and attribute Lead-Time UOM according to your business requirements. Order Management will normally create and send the request in a few seconds. However, set Default Lead Time to 1 hour to account for an unexpected problem, such as unscheduled network downtime.
Update Service	Update Screening Request for Trade Compliance
Cancel Service	Cancel Screening Request for Trade Compliance
Use Transactional Item Attributes	Contains a check mark.
Use Flexfield Attributes	Contains a check mark.
Compensation Pattern	Do not define a rule.

Note

- This step calls the service that does compliance screening.
- You can place it at any location in the sequence of steps. For this example, assume you prefer to do screening after scheduling, so place it immediately after the Schedule step.
- The values for the Task attribute and the Service attribute are required to call the compliance screening service. Other attribute values are optional.

6. In the Process Details area, create a step immediately after the Request Screening for Trade Compliance step.

Attribute	Value
Step Name	Wait for Trade Compliance Screening Use any text for the name.
Step Type	Service
Task Type	DOO_TradeCompliance
Task	DOO_TradeCompliance
Service	Wait for Trade Compliance Screening
Exit Criteria	Passed DOO_PASSED Y Set up the exit criteria to continue processing or stop processing, according to your needs. For example, to require the Order Entry Specialist to modify an order line that doesn't pass trade compliance screening, use Passed DOO_PASSED Y to allow only Passed status as the exit criteria. If the compliance screening fails or is under review, then the Order Entry Specialist can examine the message details that Order Management displays, then decide on an action to take, such as revise the sales order. For another example, to continue processing order lines while the ship-to-customer is under compliance review, include status Under Review as part of the exit criteria. Order Management will process each order line through order fulfillment while trade management reviews the ship-to-customer for trade compliance.
Default Lead Time	1
Lead-Time UOM	Days Set the Default Lead Time attribute and the Lead-Time UOM attribute according to your needs. This example uses a 1 day lead time to allow Global Trade Management to process screening that depends on an action in the fulfillment system that requires a long time to resolve.

Note

- This step pauses the orchestration process so it waits for the compliance screening service to finish screening.
- The values for attributes Task, Service, and Exit Criteria are required. Other attribute values are optional.

- Compliance screening is a long-running task, so you can do compliance screening only through a long-running task.

7. In the Process Details area, click **Status Conditions > Orchestration Process Status Values**, then add values.

Status Value	Expression
Trade Compliance Screening Is Pending	"DOO_TradeCompliance" = "DOO_AWAITING_COMPLIANCE_RSLT"
Trade Compliance Screening Passed	"DOO_TradeCompliance" = "DOO_PASSED"
Trade Compliance Screening Failed	"DOO_TradeCompliance" = "DOO_FAILED"
Trade Compliance Under Review	"DOO_TradeCompliance" = "DOO_UNDER_REVIEW"

   The exit criteria is DOO_PASSED for the Wait for Trade Compliance Screening step in this example.
8. Optional. To pause fulfillment while waiting for compliance review, or while doing more review and approval, add a pause step after the Wait step.

Related Topics

- Overview of Setting Up Trade Compliance
- Constrain Trade Compliance Screening
- Set Up Trade Control Types
- Set Up Trade Compliance in Order Management
- How Trade Compliance Works in Order Management

Constrain Trade Compliance Screening

Specify how Order Management constrains trade compliance during order entry.

Order Management comes predefined to use the DOO_GTM_EXCEPTION processing constraint to reject each sales order that doesn't pass trade compliance screening when the Order Entry Specialist clicks Submit. This constraint prevents Order Management from sending a sales order that isn't trade compliant to order fulfillment. If DOO_GTM_EXCEPTION doesn't meet your needs, you can disable it.

You can also create a new constraint. For example, create a constraint that rejects a sales order with status Failed but that doesn't reject an order with status Under Review.

In this example, you create a processing constraint that rejects each sales order that doesn't pass trade compliance screening.

This topic uses example values. You might need different values, depending on your business requirements.

Summary of the Steps

Constrain trade compliance.

1. Prepare your environment.
2. Disable DOO_GTM_EXCEPTION.
3. Create validation rule set.
4. Create processing constraint.

Prepare Your Environment

1. If you haven't enabled trade compliance screening to happen during order submit or order fulfillment, then disable any constraint that references the trade compliance status.
2. If you enabled trade compliance screening to happen only during order fulfillment, then disable any constraint that references the trade compliance status.

Disable DOO_GTM_EXCEPTION

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, locate predefined constraint DOO_GTM_EXCEPTION.
3. Make sure the Enabled option doesn't contain a check mark.

    The sales order will pass submit validation and Order Management will send the sales order to order fulfillment even if it doesn't pass compliance screening.

Create Validation Rule Set

1. On the Manage Processing Constraints page, click **Validation Rule Sets**.
2. Click **Actions > Add Row**, then set the values.

Attribute	Value
Name	Order with Trade Compliance Exception
Description	Creates a new constraint that prevents a sales order that doesn't pass trade compliance screening from moving to order fulfillment.
Short Name	TML TML is an abbreviation for trade management line. You can use any value.
Validation Type	Table
Entity	Order Header

Attribute	Value

3. In the Details area, click **Actions > Add Row**, then set the values.

Attribute	Value
Attribute Name	Trade Compliance Status
Validation Operation	Equal to
Value String	ORA_FAILED

4. Click **Save > Generate Packages**.
5. In the Confirmation dialog, click **OK**.

Create Processing Constraint

1. Click **Constraints**.
2. Click **Actions > Add Row**, then set the values.

Attribute	Value
Constraint Name	GTM_EXCEPTION_ORDER If you create a new constraint for trade compliance screening, then you must include the text `GTM` as part of the constraint name.
Display Name	Reject Sales Orders that Contain Trade Compliance Exceptions
Constraint Entity	Order Header
Constrained Operation	Submit
On Operation Action	Not allowed
Applicable Roles	All roles
Enabled	Contains a check mark.

3. In the Conditions tab, click **Actions > Add Row**, then set the values.

Attribute	Value
Group Number	1
Validation Entity	Order Header
Invert Validation Rule Set	Leave empty
Validation Rule Set	Order with Trade Compliance Exception
Scope	Any
Record Set	Header Default Record Set
Message	You can't submit the order line because it doesn't pass trade compliance screening.
Enabled	Contains a check mark.

4. Click **Save**.

Set Up Trade Control Types

Set up the trade control types that Order Management uses when it verifies a sales order.

The trade control type identifies the trade policy, such as antiterrorism or firearms convention.

Assume you must add the Chemical and Biological Weapons trade control type.

This topic uses example values. You might need different values, depending on your business requirements.

Summary of the Steps

1. Examine the predefined lookup.
2. Manage the predefined lookup.

Examine the Predefined Lookup

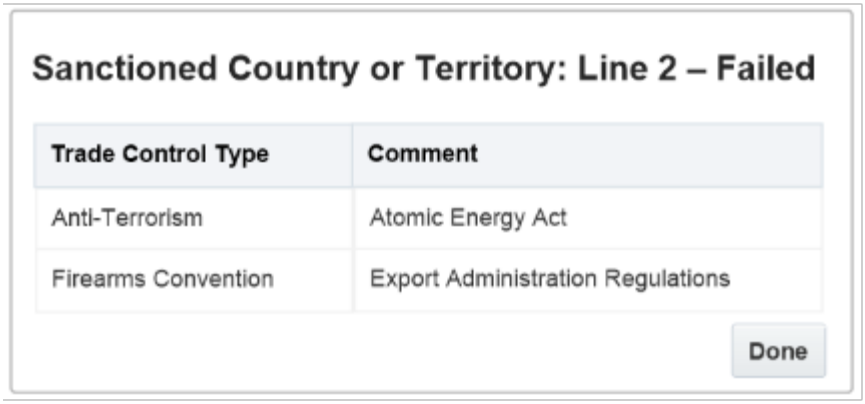
You can use a predefined lookup to set the trade control type.

1. Go to the Order Management work area, then click **Tasks > Manage Fulfillment Line Exceptions**.
2. Click a **row** that displays a yellow or red Trade Compliance Status.

3. In the Fulfillment Line details area, click **Trade Compliance** , then, next to a red or yellow trade compliance status, click **View Details**.

   The Manage Fulfillment Line Exceptions page only displays the View Details link next to a red or yellow trade compliance status.
4. Examine the predefined lookup that displays the trade control type.

   For example, a fulfillment line might reference a trade control, such as Anti-Terrorism, and details about the control, such as Atomic Energy Act.



Manage the Predefined Lookup

Manage the predefined lookup that lists the trade control type.

1. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Lookups
2. On the Manage Order Lookups page, in the Search area, enter the value, then click **Search**.

Attribute	Value
Lookup Type	ORA_DOO_TRADE

3. In the search results, click the row that contains the value.

Attribute	Value
Lookup Type	ORA_DOO_TRADE_CONTROL_CODE

4. In the Lookup Codes section, click **Actions > New**, then enter values.

Attribute	Value
Lookup Code	CHEMICALS_AND_WEAPONS
Display Sequence	1
Meaning	Chemical and biological weapons

5. Click **Save**.

Related Topics

- Overview of Setting Up Trade Compliance
- Set Up Trade Compliance in Order Management
- How Trade Compliance Works in Order Management
- Manage Lookups in Order Management
- Manage Sales Orders That Require Trade Compliance Screening

Coverages and Subscriptions

Set Up Coverages for Sales Orders

Set up and manage coverage items in Oracle Order Management. For example, set up the technical support coverage for a laptop computer.

Note

- If an order line contains a coverage, then you can't use an order management extension or a transformation rule to specify the values for a shipment set on that line. If you create an extension or rule that does attempt to set these values, then Order Management will skip the coverage line when it assigns the shipment set.
- If you import a sales order, and if your import includes a shipment set on an order line that contains a coverage item, then the import ignores the value for the shipment set on the line and creates or updates the line without the shipment set.
- If your import cancels a covered item, then Order Management will also cancel all coverage lines that aren't closed for the item.
- For an example that integrates Order Management with Oracle Receivables, see *Create One Invoice for Sales Orders with Items That Can and Can't Ship*.

Summary of the Set Up

1. Administer parameters.
2. Create the covered item.
3. Create the coverage item.

4. Set up pricing for the coverage item.
5. Map time units.
6. Test your set up.
7. Optional. Add fixed and open-ended coverage. For details, see *Add Fixed and Open-Ended Coverage to Sales Orders*.

For details, see *Coverages and Subscriptions in Sales Orders*.

In this example, you will set up variable coverage as an extended warranty for the Standard Desktop computer.

This topic uses example values. You might need different values, depending on your business requirements.

Administer Parameters

Administer the feature and the parameter that affect coverages and subscriptions.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Parameters
2. On the Manage Order Management Parameters page, set the value.

Parameter Name	Value
Coverage Start Date	To set the coverage's start date to: - The date when you ship the covered item. Set this parameter to Shipment Date. - The date when you deliver the covered item. Set this parameter to Delivery Date. You can use this setting only if you integrate with Transportation Management and you set up the integration so it sends the Actual Delivery Date to Order Management. Order Management ships only the configured item and not the configure options. Order Management updates the shipment date and delivery date only on the parent configured item and not the child configure options. If you enable coverage on a child configure option, then Order Management creates a relationship between the option and the coverage, and Order Management sets the Contract Start Date of the coverage for the child line to the parent configured item's shipment date or delivery date. **Pick-to-Order and Kits** If you set the Ship Model Complete attribute to Yes on the configuration model in the Product Information Management work area, then Order Management doesn't ship the model but does ship the configure options that the user selects. If you enable coverage for the model, then Order Management sets the model's Contract Start Date to the shippable configure options' Contract Start Date. Order Management applies the same behavior for a kit, except a kit has included items instead of configure options.

Create the Covered Item

Create and set up an item so a coverage item can cover it.

1. Go to the Product Information Management work area.
2. Click **Tasks > Create Item**.
3. In the Create Item dialog, set values, then click **OK**.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Root Item Class For details, see *Item Classes*.
Template	You can select any value. For this example, select Finished Goods.

4. On the Create Item page, enter the values, then click **Save**.

Attribute	Value
Item	PTO54222 Standard Personal Desktop Computer
Description	Standard Desktop computer.
Primary Unit of Measure	Each

5. Click **Specifications**, then, under Item Organization, click **Sales and Order Management**.
6. Set the value.

Attribute	Value
Sales Product Type	Select one. - **Goods**. A tangible covered item. - **Subscription**. An item that provides a product or service that recurs, and that includes a duration and period. For example, a one year subscription to a magazine, a 90 day subscription for cell phone service, and a six month subscription for software usage are each an example of a subscription. - **Empty**. For this example, select Goods.

Attribute	Value

7. Under Item Organization, click **Service**.
8. Set the value, then click **Save**.

Attribute	Value
Enable Contract Coverage	Yes You must enable contract coverage so a coverage item can cover the covered item.

Create the Coverage Item

You will create a coverage item that covers the PTO54222 Standard Personal Desktop Computer item. You set up the coverage item as an extended warranty, to display default values for duration and period, and to allow the Order Entry Specialist to modify these values.

1. Click **Tasks > Create Item**.
2. In the Create Item dialog, set values, then click **OK**.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Warranty/Services
Template	You can select any value. For this example, select Finished Goods.

3. On the Create Item page, set the values, then click **Save**.

Attribute	Value
Item	Variable Extended Warranty for Standard Desktop
Description	Variable warranty for the Standard Desktop computer.
Primary Unit of Measure	Year

4. Click **Specifications**, then, under Item Organization, click **Sales and Order Management**.
5. Set the value.

Attribute	Value
Sales Product Type	Select one. - Extended Warranty - Service Level Agreement - Software Maintenance - Preventive Maintenance You can use only these values for a coverage item. For this example, select Extended Warranty.

6. Under Item Organization, click **Service**.
7. Set the Service Duration Type attribute. For this example, set it to Variable.

Value	Description
Fixed	The Order Management work area will default the values that it displays for Duration and Period to the values that you set, then make them read-only.
Variable	Works the same as Fixed, except the Order Entry Specialist can edit Duration and Period.
Open Ended	Allows the Order Entry Specialist to edit Duration and Period, but doesn't display a default value for these attributes. The Order Entry Specialist must set them.

You can't set the Enable Contract Coverage attribute because this attribute allows a coverage item to cover the item that you're setting up. However, a coverage item can't cover another coverage item. For example, you can't create a 1 Year Standard Desktop Warranty that covers a 5 Year Standard Desktop Warranty.

8. Set the values, click **Save and Close**, then sign out.

Attribute	Value
Duration	5
Duration Period	Year

Attribute	Value

Note: We strongly recommend that you set these values. Order Management displays them in fields that aren't labeled on the catalog line. If you don't set them, then the fields will be empty, and your users might not understand how to use them.

Set Up Pricing for the Coverage Item

Assume you must allow the Order Entry Specialist to set the coverage so it recurs yearly or monthly, and to calculate the coverage like this:

- For yearly coverage, charge 20% of the covered item price.
- For monthly coverage, charge 10% of the covered item price.

You must set up a charge for each of these durations. For details, see *Pricing for Covered Items*.

Set up pricing for the coverage item.

1. Make sure you have the privileges that you need to administer pricing.
   Assume you already set up pricing for the PTO54222 Standard Personal Desktop Computer item so it calculates a pricing basis of $500.
2. Go to the Pricing Administration work area, then click **Tasks > Manage Price Lists**.
3. Search for Corporate Segment Price List.
4. In the search results, click **Action > Add Row**, set the values, then click **Save**.

Attribute	Value
Item	Variable Extended Warranty for Standard Desktop
Line Type	Buy

   If you:
   - Don't enter a value for Service Duration, then Pricing won't calculate a value for the Duration Extended Amount attribute.
   - Enter a value for Service Duration, then Pricing will calculate a value for the Duration Extended Amount attribute.

   If the Service Duration Type is Open Ended for the coverage item in the Product Information Management work area:
   - The Search Results area won't display a value for Service Duration Period or Service Duration, and won't allow you to edit these attributes.
   - And if Pricing prices the sales order instead of your source system pricing it, then the Order Entry Specialist must enter a value for Service Duration Period.
5. In the Associated Items column, click **Manage Covered Item**.

6. On the Manage Covered Items page, in search results, click **Add Row**, set the values, then click **Create Charge**.

Pricing UOM	Coverage UOM	Action Type
Each	Year	Add

7. In the Charge area, set the values, then click **Save and Close**.

Attribute	Value
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	20
Price Periodicity	Year

8. In the search results, click **Add Row**, set the values, then click **Create Charge**.

Pricing UOM	Coverage UOM	Action Type
Each	Month	Add

9. In the Charge area, set the values, then click **Save and Close**.

Attribute	Value
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	10
Price Periodicity	Month

Attribute	Value

10. Click **Save and Close** again, then sign out of Pricing.

Specify Charges for Each Duration

Duration comes predefined to display values.

- Year
- Quarter
- Month
- Week
- Day
- Hours
- Minutes

Note

- Order Management renders pricing details only for each charge that you specify. For example, if you don't specify a charge for Month, and if your user sets Duration to Month, then the order line for the coverage item won't display price details, Order Management will display an error message, and the user can't submit the sales order.
- If you set up a coverage item in the Product Information Management work area but don't set up pricing for it in the Pricing Administration work area, then your user might add the coverage item but not be able to submit the sales order because of a pricing error.

CAUTION: If you set up a coverage item in Product Information Management but incorrectly set up pricing for it in Pricing Administration, then your user might add the coverage item and successfully submit it without error, but the pricing values on the order line might not be correct, resulting in overcharging or undercharging the billing.

Map Time Units

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Time Unit Mappings
2. On the Manage Time Unit Mappings page, set the values.

User Unit	Base Unit	Conversion
YEAR	Year	1
MONTH	Month	1

User Unit	Base Unit	Conversion
DAY	Day	1
QUARTER	Month	3
WEEK	Day	7
YEAR	Month	12

Make sure you use these values. These values will accommodate most coverages and subscriptions in a typical Order Management implementation.

For details, go to *Implementing Oracle Subscription Management*, then search for Overview of Unit of Measure, or search for Time Unit Mappings.

3. Make sure the Active attribute for each row contains a check mark, then click **Save and Close**.

Test Your Set Up

Order Management uses time unit mappings to convert the duration of your coverage or subscription between different time units, and to set the Contract End Date attribute, which is required starting in Update 20C.

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area and create a sales order.
3. On the catalog line, search for the item, then click **Add**.

Field	Value
Select Item	PTO54222

4. On the catalog line, search for the item.

Field	Value
Select Item	Variable Extended Warranty for Standard Desktop

5. Wait for the search to finish, then verify that the catalog line displays these values.

Attribute	Value
Duration	5
Period	Year

Attribute	Value

These are the default values that you set up in Product Information Management.

6. Click **Select Covered Item**, select the order line to cover, then click **Add**.
7. Click **View > Columns**, then click **Period**.
8. Verify that these values display on the coverage line, by default:

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	100	100	500	5	Year

9. Change the quantity on the covered line from 1 to 2, then verify that the coverage line displays these values:

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
2	100	200	1000	5	Year

10. Change the quantity on the covered line from 2 to 1.
11. Change the Duration to 1, then verify that the coverage line displays these values:

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	100	100	100	1	Year

12. Change the Period to Month, then verify that the coverage line displays these values:

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	50	50	50	1	Month

13. Change the Period to Day, verify that the coverage line doesn't display price details, click **Actions > Reprice Order**, then notice the error message.

    Order Management will display an error for any charge that you don't set up. Recall that you didn't create a charge for Day.

Specify UOM Class for Subscriptions That Have Recurring Charges

If you set up a subscription that has a recurring charge, then you might need to set up a profile. If you don't, you might encounter an error.

```
Order Management can't price the sales order because it can't identify a charge for an item.

A matching price list can't be found for this transaction for the pricing strategy.
```

To avoid this problem, you must specify the default UOM class to use for the service duration. Here's how.

1. Set the Default Price Periodicity UOM Class pricing parameter to Time. For details, see *Manage Pricing Parameters*.
2. Set the profile value.
    - Go to the setup and maintenance work area, click **Tasks**, then search for Manage Administrator Profile Values.
    - On the Manage Administrator Profile Values page, search for the value.

Attribute	Value
Profile Display Name	SCM Common: Default UOM Class for Service Duration

    - In the Profile Values area, set the value.

Attribute	Value
Profile Level	Site
Profile Value	Time Order Management uses this value as the default UOM class for a service duration.

Related Topics

- Add Fixed and Open-Ended Coverage to Sales Orders
- Create One Invoice for Sales Orders with Items That Can and Can't Ship
- Coverages and Subscriptions in Sales Orders
- Item Classes
- Manage Pricing Parameters

Add Fixed and Open-Ended Coverage to Sales Orders

The behavior of fixed coverage and open-ended coverage is slightly different from the behavior for variable coverage. As an option, you will create a fixed coverage item, an open-ended coverage item, then test them.

Summary of the Set Up

1. Create the coverage item.
2. Set up pricing.
3. Test your set up.

Create the Coverage Item

1. Use the Product Information Management work area to create the coverage item.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Warranty/Services
Template	Finished Goods
Item	Fixed Extended Warranty for Standard Desktop
Description	Fixed warranty for the Standard Desktop computer.
Primary Unit of Measure	Year
Sales Product Type	Extended Warranty
Service Duration Type	Fixed
Duration	5
Duration Period	Year

2. Create another coverage item.

Attribute	Value
Organization	Vision Operations

Attribute	Value
Number of Items	1
Item Class	Warranty/Services
Template	Finished Goods
Item	Open-Ended Extended Warranty for Standard Desktop
Description	Open-ended warranty for the Standard Desktop computer.
Primary Unit of Measure	Year
Sales Product Type	Extended Warranty
Service Duration Type	Open Ended

Set Up Pricing

1. Sign out, and then sign into Oracle Pricing.
2. Set up pricing for the fixed coverage item on the Corporate Segment Price List.

Attribute	Value
Item	Fixed Extended Warranty for Standard Desktop
Line Type	Buy
Pricing UOM on the Manage Covered Items page	Each
Coverage UOM	Year
Action Type	Add
Pricing Charge Definition	OAL Recurring Charge

Attribute	Value
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	20
Price Periodicity	Year

Order Management doesn't allow your user to change the Duration or Period for a fixed coverage item, so you only need to set up a charge for one period.

3. Add the open-ended coverage item to the Corporate Segment Price List.

Attribute	Value
Item	Open-Ended Extended Warranty for Standard Desktop
Line Type	Buy

4. Add the charge for Year.

Attribute	Value
Pricing UOM	Each
Coverage UOM	Year
Action Type	Add
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	20

Attribute	Value
Price Periodicity	Year

5. Add the charge for Quarter.

Attribute	Value
Pricing UOM	Each
Coverage UOM	Quarter
Action Type	Add
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	5
Price Periodicity	Quarter

6. Add the charge for Month.

Attribute	Value
Pricing UOM	Each
Coverage UOM	Month
Action Type	Add
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent

Attribute	Value
Coverage Basis	Coverage Basis Your Price
Calculation Amount	10
Price Periodicity	Month

7. Add the charge for Week.

Attribute	Value
Pricing UOM	Each
Coverage UOM	Week
Action Type	Add
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Covered Item Price Percent
Coverage Basis	Coverage Basis Your Price
Calculation Amount	7
Price Periodicity	Week

8. Save your set up, and then sign out of Pricing.

Test Your Set Up

1. Sign into Order Management, then create a sales order.
2. On the catalog line, search for the item, then click **Add**.

Field	Value
Select Item	PTO54222

3. On the catalog line, search for the item.

Field	Value
Select Item	Fixed Extended Warranty for Standard Desktop

4. Wait for the search to finish, then notice that the catalog line displays 5 Year as a read-only value.
5. Click **Select Covered Item**, select the order line to cover, then click **Add**.
6. Verify that you can't edit the Duration or Period.
7. On the catalog line, search for the item.

Field	Value
Select Item	Open-Ended Extended Warranty for Standard Desktop

8. Wait for the search to finish, then notice that the catalog line displays two empty fields, and that you must set them before you continue. If you don't, then Order Management displays an error.

   An open-ended coverage item is similar to a variable coverage item, except Order Management displays the default values for Duration and Period on the catalog line for a variable coverage item, and doesn't display any values for an open-ended coverage item. The Order Entry Specialist must set these values for an open-ended coverage item.
9. Set the values on the catalog line.

Attribute	Value
Duration	1
Period	Day

10. Click **Select Covered Item**, select the order line to cover, then click **Add**.
11. Verify that the coverage order line for the open-ended coverage doesn't contain prices. You didn't set up a charge for Day, so no pricing will display, and you can't submit the sales order.
12. Set Period to Year, then verify that these values display on the coverage line.

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	100	100	100	1	Year

13. Set Period to Month, and then verify that these values display on the coverage line:

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	50	50	50	1	Year

14. Set Period to Week, then verify that these values display on the coverage line.

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	35	35	35	1	Year

15. Set Duration to 2, then verify that these values display on the coverage line.

Quantity	Your Price	Amount	Amount for Total Duration	Duration	Period
1	35	35	70	1	Year

Related Topics

- Coverages and Subscriptions in Sales Orders

Set Up Orchestration Processes for Coverage Items

You can set up an assignment rule so it uses the same orchestration process instance to process the covered item and the coverage item.

You can also set it up so it uses one orchestration process instance to process the covered item and another orchestration process instance to process the coverage item, depending on your needs.

For details, see *Coverages and Subscriptions in Sales Orders*.

Use the Same Orchestration Process Instance

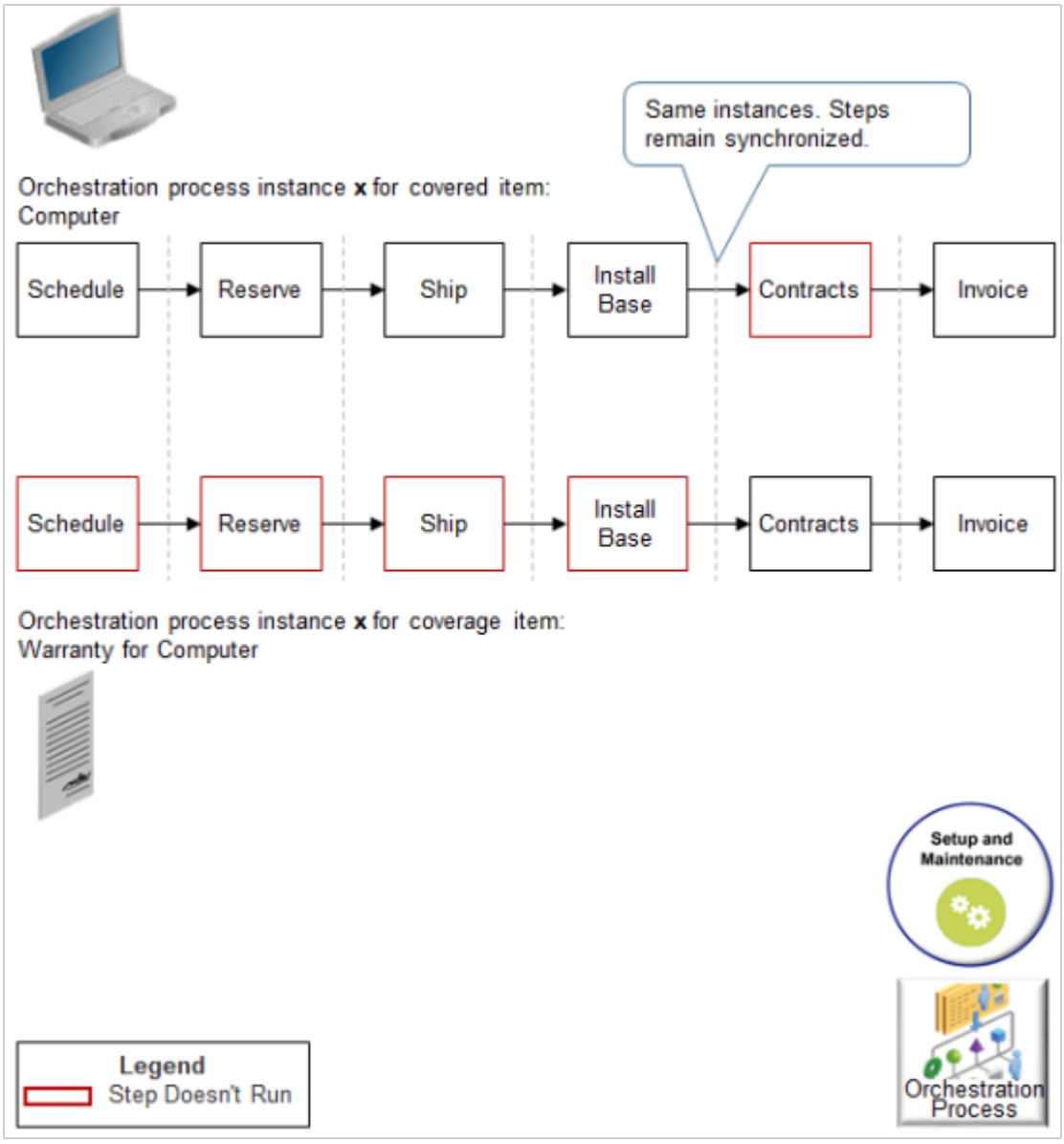
If you add the covered item and coverage item in the same sales order at the same time when you create or revise the sales order, then you can assign the same orchestration process instance to process the covered item and the coverage item together.

- Make sure the orchestration process uses the same steps, in the same sequence, and at the same time for the covered item and for the coverage item.
- Make sure processing for the covered item and coverage item remain synchronized, and make sure processing for the covered item doesn't get too far ahead or behind processing for the coverage item.

- Order management assigns the same orchestration process number to the covered item and to the coverage item.

Try It

1. Set up your orchestration process. Here's an example.



Note

- A red step means the step doesn't apply to the item so the orchestration process skips the step. For example, you don't schedule, reserve, or ship a warranty, so those steps don't run for the warranty.
- The covered item and the coverage item move together through the same steps, at the same time.
- If a step applies to the covered item but not the coverage item, then the coverage item will be in a Not Started status, and remain in this status until it reaches a step that applies to the covered item and the coverage item.

In this example, the Warranty for Computer coverage item is in the Not Started status all the way to the Invoice step because the Invoice step is the first step in the process that applies to the covered item and the coverage item.

2. Set up an assignment rule to make sure Order Management uses the same orchestration process for the covered item and for the coverage item.

   You set up the rule one time during setup. The orchestration process instance that runs in the runtime environment is different for each sales order or fulfillment line.

   For details, see *Assign Orchestration Processes*.
3. Set up an expression in the line selection criteria for each orchestration process step that must not run so the step doesn't call the fulfillment task service.

   The red steps in the diagram must not run.

   - You don't ship a coverage item, such as a warranty, so you set up an expression that prevents the shipping step from processing the coverage item. Add a line selection criteria that specifies to process only shippable lines. Add the criteria to a Scheduling task or Shipping task. Predefined orchestration processes already use this line selection criteria.
   - The Contracts step doesn't need to run for a covered item, so add a line selection criteria that specifies to run this step for each line only when the Sales Product Type specifies a coverage item.

   Learn how to set up an expression. For details, see *Select Fulfillment Lines for Orchestration Process Steps*.

This orchestration process is only for illustration purposes. You might need a different set up for your assignment rules and expressions.

Use Different Orchestration Process Instances

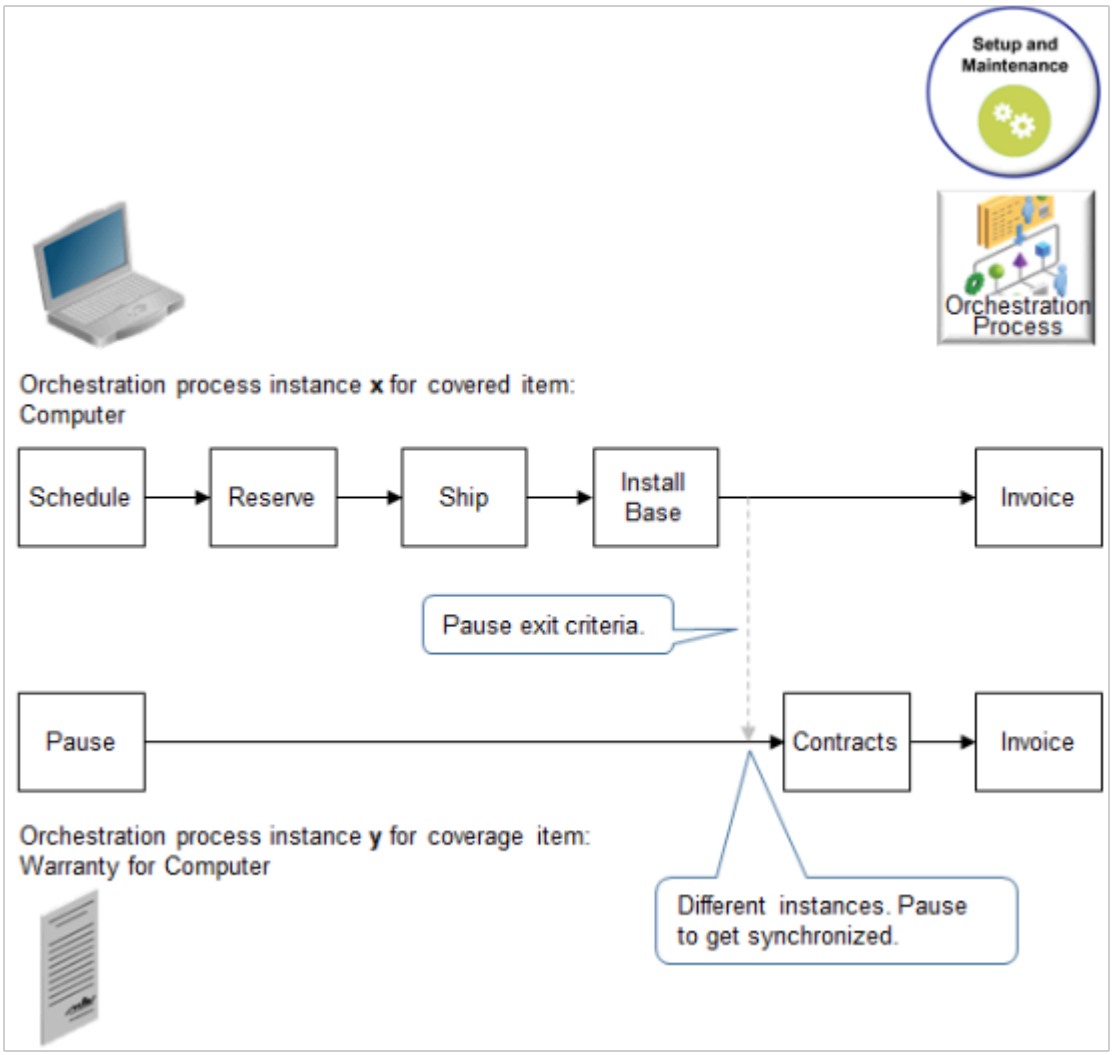
It might be necessary to use one orchestration process instance to process the covered item and a different orchestration process instance to process the coverage item. For example:

- Your business requirements demand that you process them separately.
- An assignment rule assigns separate process instances when creating the sales order.
- You use the Revise Order action in the Order Management work area to revise the sales order, add coverage to an existing covered item, then submit the revision. Order Management assigns a separate orchestration process instance even if the assignment rule assigns the same instance that processes the covered item.
- Order management assigns one number to the orchestration process instance that processes the covered item and a different number to the instance that processes the coverage item.

Try It

1. Set up your orchestration process.

   Here's an example that illustrates how you can use a pause task to make sure the coverage item and covered item go to billing at the same time.



   Note

   - Orchestration process instance x can process the covered item independently of the coverage item.
   - Orchestration process instance y can process the coverage item independently of the covered item.
   - The coverage item will proceed directly to the pause task, then wait until the covered item finishes the Install Base step.
   - The Install Base step finishes, the orchestration process releases the pause on the coverage item, and then the covered item and coverage item move together to the Invoice task.

2. Add a pause task.
    - If your orchestration process includes an invoice step, then make sure you do the billing step for the covered item and the coverage item at the same time.

      Fulfillment for the covered item is almost always behind fulfillment for the coverage item because the covered item goes through shipping but the coverage item doesn't. So, to do billing at the same time, add a pause step immediately before the invoice step in the orchestration process that processes the coverage item. This way, fulfillment for the coverage item pauses while it waits for fulfillment for the covered item to get to the billing step.
    - The pause task makes sure the orchestration process instance that processes the coverage item doesn't get too far out in front of the instance that processes the covered item. For example, the Contracts step must not process the coverage item until after the covered item ships and the Install Base step runs.
    - You can set the pause exit criteria to a condition, such as `Shipped Quantity is greater than zero.`
    - Order Management doesn't allow updates to the order line once billing starts. Adding a pause task makes sure that the orchestration can apply any actions you do on the covered item to coverage item, such as update, split, or cancel.
        - Assume you set the quantity on the covered item and the coverage item to 5, then submit the sales order.
        - Assume the coverage item proceeds immediately to billing.
        - You create an order revision, revise the quantity on the covered item to 4, and submit it.
        - The revision goes through fulfillment and proceeds to billing. However, the quantity for the coverage item is 5, its already in billing, so now the quantities for the coverage item and the covered item don't match, causing an error in the billing step. Pausing the coverage prevents this problem.
    - For details, see *Pause Orchestration Processes Until Events Happen*.

Here's an example If statement for the pause task.

Pseudocode for the If statement.

- If the fulfillment line contains a coverage item, and if the fulfillment line that the coverage item covers is shippable but hasn't yet shipped

Here's an example Then statement for the pause task.

The pause continues until actualShipDate contains a date, which indicates that shipping has shipped the covered item and the orchestration process can proceed to the billing step.

For details about the code that you see in this rule, see *Pause Orchestration Processes Until Events Happen*.

Related Topics

- Guidelines for Assigning Orchestration Processes
- Select Fulfillment Lines for Orchestration Process Steps
- Overview of Pausing Orchestration Processes
- Set Up Coverages for Sales Orders
- Coverages and Subscriptions in Sales Orders

Set Up Subscriptions for Sales Orders

Setting up a subscription for a sales order is similar to setting up coverage for a sales order.

- You can include a duration and period.
- You can set up a subscription as a configured item.
- You enable a subscription for contract coverage similar to how you enable a covered item so it's available to associate with a service.
- You can specify another recurring price for on-going, periodic billing that bills the subscription for the periods that happen after the first pay period.

- If you import a subscription, then the import must include the start date, end date, duration, duration UOM.

Assume you must set up an annual subscription for Visions Magazine at a rate of $5 for each week.

Set up your coverage. For details, see *Set Up Coverages for Sales Orders*. Do that setup, but with these differences:

1. Use the Product Information Management work area to create a subscription.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Warranty/Services
Template	Finished Goods
Item	Subscription to Visions Magazine
Description	Subscription for Visions Magazine, a weekly publication for trade professionals.
Primary Unit of Measure	Week Pricing typically must reference a primary UOM that it can measure numerically, such as number of weeks, number of users, or number of hosted environments. If you can't use a predefined UOM, then you must create a new one.
Sales Product Type	Subscription
Service Duration Type	Variable You can use Fixed, Variable, or Open-Ended. Use Variable in most situations because it lets you suggest a default value, but also provides the Order Entry Specialist the flexibility to adjust or negotiate price for each customer. If you use Open Ended, then renewal isn't required and the subscription continues until the Order Entry Specialist ends it. For example, assume your customer buys a subscription to an electrical utility that must continue indefinitely until the customer actively ends it.
Duration	52
Duration Period	Week

2. Use the Pricing Administration work area to set up pricing for the subscription on the Corporate Segment Price List.

Attribute	Value
Item	Subscription to Visions Magazine
Line Type	Buy
Pricing UOM	Week
Service Duration Period	Week
Service Duration	52

3. Add the charge for Week.

Attribute	Value
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Price
Base Price	5
Calculation Amount	20
Price Periodicity	Week

4. Sign out of Pricing Administration, and then sign into Order Management.
5. Add the Subscription to Visions Magazine item to the sales order, then verify that the order line displays these values.

Attribute	UOM	Your Price	Amount	Amount for Total Duration	Duration	Period	Billing Frequency	Number of Billing Periods
1	Week	5	5	260	52	Week	One Time Billing	1

Related Topics

- Set Up Coverages for Sales Orders
- Coverages and Subscriptions in Sales Orders

Set Up Subscriptions for Covered Items

In this example, set up a covered item for cloud up time, then set up a subscription that allows your users to add the up time.

Summary of the Steps

1. Add the User UOM.
2. Set up the agreement and the subscription.

This topic uses example values. You might need different values, depending on your business requirements.

Add the User UOM

Add a User UOM as a way to measure the type of usage the subscription and the covered item provides.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Product Management
    - Functional Area: Items
    - Task: Manage Units of Measure
2. On the Manage Units of Measure page, click **Manage UOM Classes**.
3. On the Manage UOM Classes page, click **Actions > Add**, set the values, then click **Save and Close**.

Attribute	Value
Class Code	USE
Class Name	User
Description	People who use a subscription
UOM Code	USR
Base UOM Name	Users
BASE UOM Description	Someone who uses a subscription

Set Up the Agreement and the Subscription

Set up your coverage. For details, see *Set Up Coverages for Sales Orders*. Do that setup, but with these differences.

1. Use the Product Information Management work area to set up the agreement.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Warranty/Services
Template	Finished Goods
Item	Vision Cloud Agreement
Description	Service agreement to provide up time to the Vision Cloud.
Primary Unit of Measure	Each
Sales Product Type	Service Level Agreement
Service Duration Type	Fixed
Duration	3
Duration Period	Year

2. Set up the subscription.

Attribute	Value
Organization	Vision Operations
Number of Items	1
Item Class	Warranty/Services

Attribute	Value
Template	Finished Goods
Item	Vision Cloud Subscriptions
Description	Subscriptions for end-user access to the Vision Cloud.
Primary Unit of Measure	Users
Sales Product Type	Subscription
Service Duration Type	Fixed
Duration	3
Duration Period	Year

3. Use the Pricing Administration work area to set up pricing for the agreement on the Corporate Segment Price List.

Attribute	Value
Item	Vision Cloud Agreement
Line Type	Buy
Pricing UOM (on the Edit Price List page)	Each
Pricing UOM (On the Manage Covered Items page)	Each
Coverage UOM	Users

4. Add the charge.

Attribute	Value
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Price
Base Price	30000
Price Periodicity	Year

5. Set up pricing for the subscription on the Corporate Segment Price List.

Attribute	Value
Item	Vision Cloud Subscriptions
Line Type	Buy
Pricing UOM	Users
Pricing UOM	Users

6. Add the charge.

Attribute	Value
Pricing Charge Definition	OAL Recurring Charge
Calculation Method	Price
Base Price	1000
Price Periodicity	Year

7. Sign out of Pricing Administration, then sign into Order Management.

8. Create a sales order, add the Vision Cloud Subscriptions item to the sales order, then verify that the order line displays these values.

Quantity	UOM	Your Price	Amount	Amount for Total Duration	Duration	Period
1	Users	1,000	1,000	3,000	3	Year

9. Search for the Vision Cloud Agreement item on the catalog line, click **Select Covered Item**, associate it with the Vision Cloud Subscriptions order line, click **Add**, then verify that the Vision Cloud Agreement order line displays these values.

Quantity	UOM	Your Price	Amount	Amount for Total Duration	Duration	Period
1	Users	1,000	1,000	3,000	3	Year

Related Topics

- Set Up Coverages for Sales Orders
- Set Up Subscriptions for Sales Orders
- Set Up Orchestration Processes for Coverage Items
- Coverages and Subscriptions in Sales Orders

Import and Update Source Orders That Include Coverages and Subscriptions

Import and update a source order that includes a covered item, coverage item, or subscription into Order Management.

This topic describes how to import or update source orders that include coverage. However, you can also use it to import subscriptions.

Import source orders that include coverage into Order Management.

1. Select a technology to import your source orders.
    - Web service in Oracle Application Development Framework (ADF).
    - Web service in Oracle Service-Oriented Architecture (SOA).
    - Business-to-business (B2B) messaging in Oracle Collaboration Messaging Framework.
    - File Based Import. For details, see *Import Orders Into Order Management*.

2. Make sure your source order includes the required attributes.

Attribute	Value
Document Reference Type	Specify the type. It creates a relationship between the covered item and the coverage item.
Document Line Id	Line Id in the source transaction. This Id identifies the coverage item.
Document Sub Line Id	Line Id in the source transaction. This Id identifies the coverage item. To purchase an item and apply coverage for it. o **At the same time**. This subline can reside in the same source order. o **At some later time**. This subline can reside in a closed order line from another sales order that you already imported.
Document Id	Source order Id of the covered line.
Document Number	Sales order number of the covered line.
Document Line Additional Id	Line Id from the source transaction. This Id identifies the coverage for the root parent of a configured item.

The import uses these attributes to establish a relationship between the coverage item and the covered item. The import process you use must map each of these attributes in the import payload of the source order to an attribute in a sales order in Order Management.

3. Make sure your source order includes a value for the Number of Billing Periods attribute for the coverage item.
4. Make sure your source order includes values for coverage item attributes, depending on whether the source order contains price details.

Contains Price Details	Doesn't Contain Price Details
Oracle Pricing won't price the item. o Duration Extended Amount must contain a value, and this value must use the same currency that the order header uses. o Contract Start Date must contain a value.	Oracle Pricing will price the item. Each of these attributes must contain a value. o Duration o Period

Contains Price Details	Doesn't Contain Price Details
○ Values for Duration, Period, and Contract End Date can be empty.	○ Contract Start Date

5. Verify settings in the Product Information Management work area.
    - Make sure you set up the Enable Contract Coverage attribute and Sales Product Type attribute so they support the covered item. For details, see *Set Up Coverages for Sales Orders*.
    - To specify a subscription item, set Sales Product Type to Subscription. To use a subscription as a covered item, enable Contract Coverage for the subscription.

      A subscription is similar to a coverage item. It includes a Duration, Period, Contract Start Date, and Contract End Date. In general, the rules that apply when you import a coverage item also apply when you import a subscription. For details, see *Set Up Subscriptions for Sales Orders*.
6. Set the values for duration and period differently according to your set up.

Service Duration Type in Product Information Management	Description
Fixed	Make sure the Duration and Period in the source order match the values you set for Duration and Period in Product Information Management.
Open Ended	If you import a source order that includes a coverage item, and if the coverage item doesn't include price details, then make sure the import includes values for Duration and Period.

7. Make sure the value of the coverage quantity equals the value of the covered quantity.
8. Make sure the value of the coverage UOM equals the value of the covered UOM.
9. If the coverage item covers a covered item that Order Management already submitted to order fulfillment, then make sure the status of the covered item is Closed.
10. Examine your import payload. Make sure the document references that identify the coverage line map to only a single covered line.

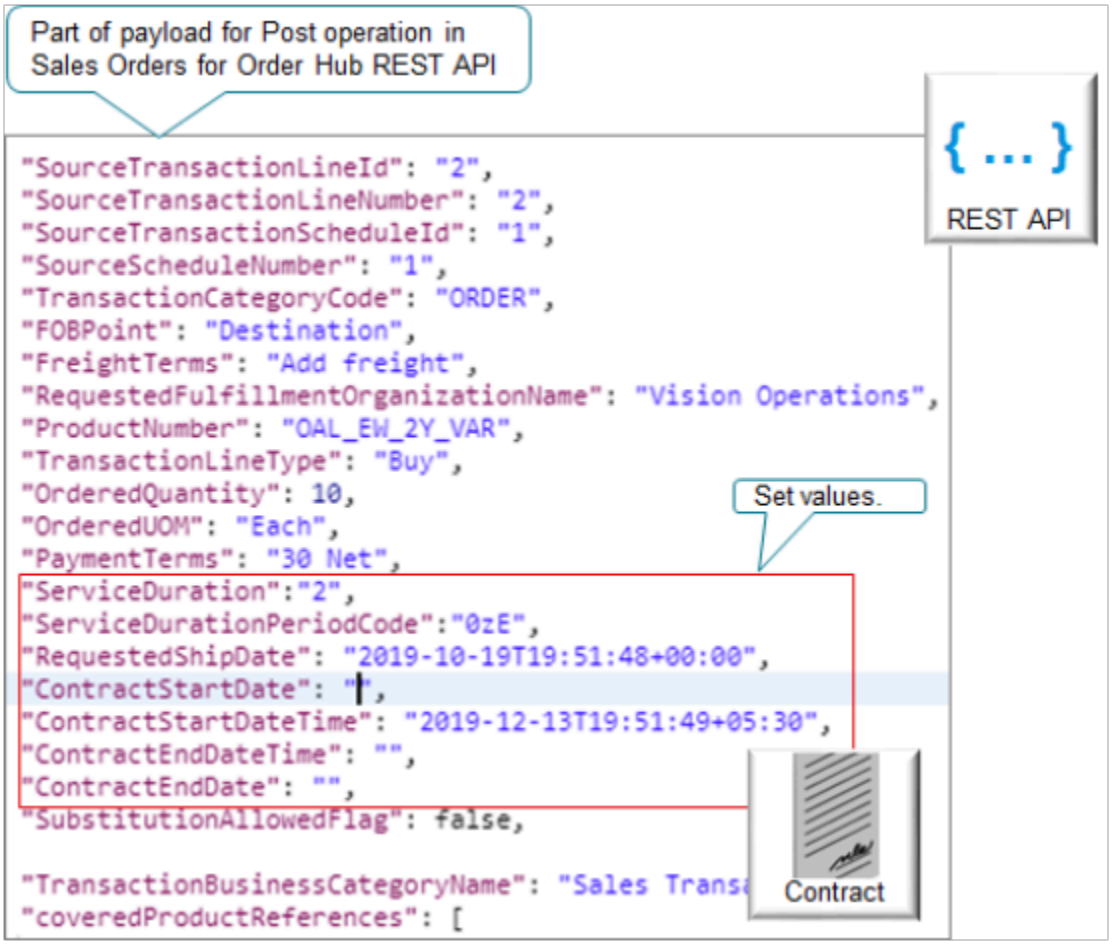
    One coverage line can cover only one covered line. A coverage line must not cover more than one covered line.

Import Date and Time

As an option, specify the time stamp for the contract start date and the time stamp for the contract end date for the coverage or subscription. The time stamp includes the date and time of day.

- If you import the start time stamp, duration, and period, then Order Management uses these values to automatically calculate the end time stamp for you. Order Management will set the end of day to 23:59:59, UTC (Coordinated Universal Time).
- If you import the start time stamp and the end time stamp, then Order Management uses these values to automatically calculate the duration and period.

For example, use REST API to import the contract duration, start time stamp, and end time stamp. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.



This example payload imports the OAL_EW_2Y_VAR warranty with.

- A duration of 2
- Period of `0zE`
- Contract start time stamp of 2019-12-13T19:51:49+05:30
- No value for the end time stamp

Note

- Import your time stamp in UTC.
- You can include an offset from UTC. For example, `2019-12-13T19:51:49+05:30` includes an offset of `+05:30`.
- The Order Management database stores time stamp values in UTC. The Order Management work area displays them in UPTZ (User Preference Time Zone). For details, see *Time Zone Differences in Order Management*.
- Use a whole number for the duration. Don't include a decimal value.

Use different attributes for different technologies.

Technology	Attributes
Sales Orders for Order Hub REST API or an Application Development Framework (ADF) web service.	• Use the ContractStartDateTime and ContractEndDateTime attributes to import your time stamps for the current release. • Use the ContractStartDate and ContractEndDate attributes for earlier releases. • Use the ServiceDuration attribute to import the duration of a coverage or subscription. • Use the ServiceDurationPeriodCode attribute to import the period of a coverage or subscription.
SourceSalesOrderImportTemplate FBDI template.	Use these attributes. • Contract Start Date • Contract End Date • Service Duration • Service Duration Code

Import Coverage with Configured Items That You Already Priced

If the source order contains price details for a configured item, and if you set each of these attributes in the source order to Y:

- Freeze Price
- Freeze Shipping Charge
- Freeze Tax

Then make sure the input payload includes coverage for each configure option that you enable for contract coverage.

If you set the Enable Contract Coverage attribute to Y on the Edit Item page in the Product Information Management work area, then the configure option is enabled for contract coverage.

Make sure your import does these validations:

- If the source order doesn't contain a coverage for each configure option, then reject the source order.
- If the coverage quantity and the UOM for each configure option don't match the line quantity and UOM for the covered item, then reject the source order.

Make sure it does these validations for each configure option that's enabled for contract coverage.

For details, see *Pricing for Covered Items*.

Import Coverage with Configured Items That You Haven't Priced

If the source order doesn't contain price details, and if you set any of these attributes in the source order to N:

- Freeze Price
- Freeze Shipping Charge
- Freeze Tax

Then Oracle Pricing will price it.

Note

- The source order must include coverage only for the configured item that's the root parent, and not for each configure option.
- If you enable the configure option for contract coverage, then your import process must create a coverage line for each configure option. Each configure option can be an option class or option item.
- If the input payload contains a coverage for the root parent and also for one or more configure options, then your import must reject the source order.

Import Coverage That Includes a Return

You can import a covered item that includes a return.

- You can keep the covered item and return only the coverage item. Make sure your import payload includes only the return lines for the coverage item.
- If the Contract End Date of the coverage line happens before the return date of the coverage line, then you can't return a coverage line.

Your return must include these details:

- The status on the covered line and on the coverage line are each Closed.
- Return quantity is greater than zero.
- Return quantity doesn't exceed the quantity that's available to return. For example, if the sales order included a quantity of 10, and if you already returned a quantity of six, then the return quantity can't exceed four.
- The return coverage line must have the Original_Sales_Order document reference type.
- The covered line must have the Source_Coverage_Covered_Association document reference type.

Note these points when you return a covered item, depending on whether the source order contains price details:

Contains Price Details	Doesn't Contain Price Details
If Oracle Pricing doesn't price the item, then you can return the coverage item that covers the covered item. Make sure your import payload includes these details. • The quantity and UOM of the coverage line equal the quantity and UOM of the covered line. For example, assume a sales order includes a quantity of 5 for the covered item and a quantity of 5 for the coverage item. If the return includes a quantity of 3 for the covered item, then make sure the coverage line in the return includes a quantity of 3. • Adds the returned quantity to the quantity that's available.	If Oracle Pricing prices the item, then the import payload can specify to return the covered item and the coverage lines. Make sure your import payload includes the return line for the covered item. Oracle Applications will create the return lines for the coverage lines.

Contains Price Details	Doesn't Contain Price Details
• Includes return charges.	

If your import payload returns only the coverage item, then note these points:

- The quantity for the coverage line must equal the total returnable quantity of the covered line. Your import payload can't return only part of the returnable quantity.
- The coverage return UOM in the import payload must equal the covered line UOM.
- If the source order contains price details, then the import payload must include the return charges.

Update Coverages and Subscriptions Through Web Services

Use attributes in the FulfillmentResponseService payload to update contract dates.

- ContractStartDate
- ContractStartDateTime
- ContractEndDate
- ContractEndDateTime

Note

If You Import	Description
Only the start date.	Order Management uses the duration and period on the fulfillment line to calculate the contract end date.
Contract start date and contract end date.	If you set Service Duration Type to Variable when you set up the coverage, and if you import the contract start date and the contract end date, then Order Management recalculates the duration on the fulfillment line. However, the revised duration doesn't result in a price change on the fulfillment line. Learn about Service Duration Type. For details, see *Set Up Coverages for Sales Orders*.
Only the contract end date.	Order Management ignores the request and doesn't update any dates because it requires the contract start date.

If you use FulfillmentResponseService to update contract dates, then make sure you specify the same contract dates in any subsequent updates you make to other attributes on the coverage line. If you don't specify contract dates in subsequent updates, then Order Management uses the Coverage Start Date parameter and the actual shipment date or actual delivery date to determine how to set the values for the contract dates. For details, see *Manage Order Management Parameters*.

Splitting Lines

You can't use a FulfillmentResponseService request to split:

- An order line that has a covered item, and then use the same request to update the contract dates on the coverage line that covers the covered item. Instead, use one request to split the line that contains the covered item, then send a subsequent request to update the coverage item.

- Only the coverage line. Order Management automatically processes the coverage line when it splits the covered line.

If Order Management already split a line, then you can't use Application Development Framework (ADF) to add a coverage to the line. You can use only the Order Management work area or REST API to add coverage to a split line.

For details about using FulfillmentResponseService, see *Connect Order Management to Your Fulfillment System*.

Import Billing Plans

If you import a billing plan, then make sure the PeriodicityCode attribute doesn't contain ONE TIME under the BillingPlans entity for the order line in your import payload.

Consider an example.

```
"shipToCustomer": [
 {
 "PartyId": 1006,
 "SideId": 1036,
 }
],
"lines": {
 {
 "SourceTransactionLineId": "CPQ_LINE_ID_001",
 "SourceTransactionLineNumber": "CPQ_LINE_NUM_001",
 "SourceTransactionScheduleId": "CPQ_SCH_ID_001",
 "SourceScheduleNumber": "CPQ_SCH_NUM_001",
 "ProductNumber": "CPQ_SCH_NUM_001",
 "InvoicingRule": "CPQ_SCH_NUM_001",
 "BillingTransactionTypeName": "CPQ_SCH_NUM_001",
 "PaymentTermsCode": "CPQ_SCH_NUM_001",
 "TransactionLineType": "Buy",
 "OrderedUOM": "Each",
 "PurchasingUOMCode": "Ea",
 "OrderedQuantity": 1,
 "ServiceDuration": 1,
 "ServiceDurationPeriodName": "YEAR",
 "ContractStartDateTime": "2020-10-01",
 "ContractStartEndDateTime ": "2021-09-30",
 "SubscriptionProfileName": "Subs_Profile",
 "ExternalPriceBookName": "CPQ_Price_List",
 "billingPlans": [
 {
 "BillingNumberOfPeriods": 12,
 "BillingPlanTypeCode": "PERIODIC",
 "PeriodicityCode": "MONTH",
 }
 ]
''charges" : [
 {
 "SourceChargeId": "C1-OT",
 "ApplyTo": "Price",
 "PriceType": "One_time",
```

Here we have `"PeriodicityCode": "MONTH"` in the billingPlans entity, which should work fine. But `"PeriodicityCode": "ONE TIME"` would fail.

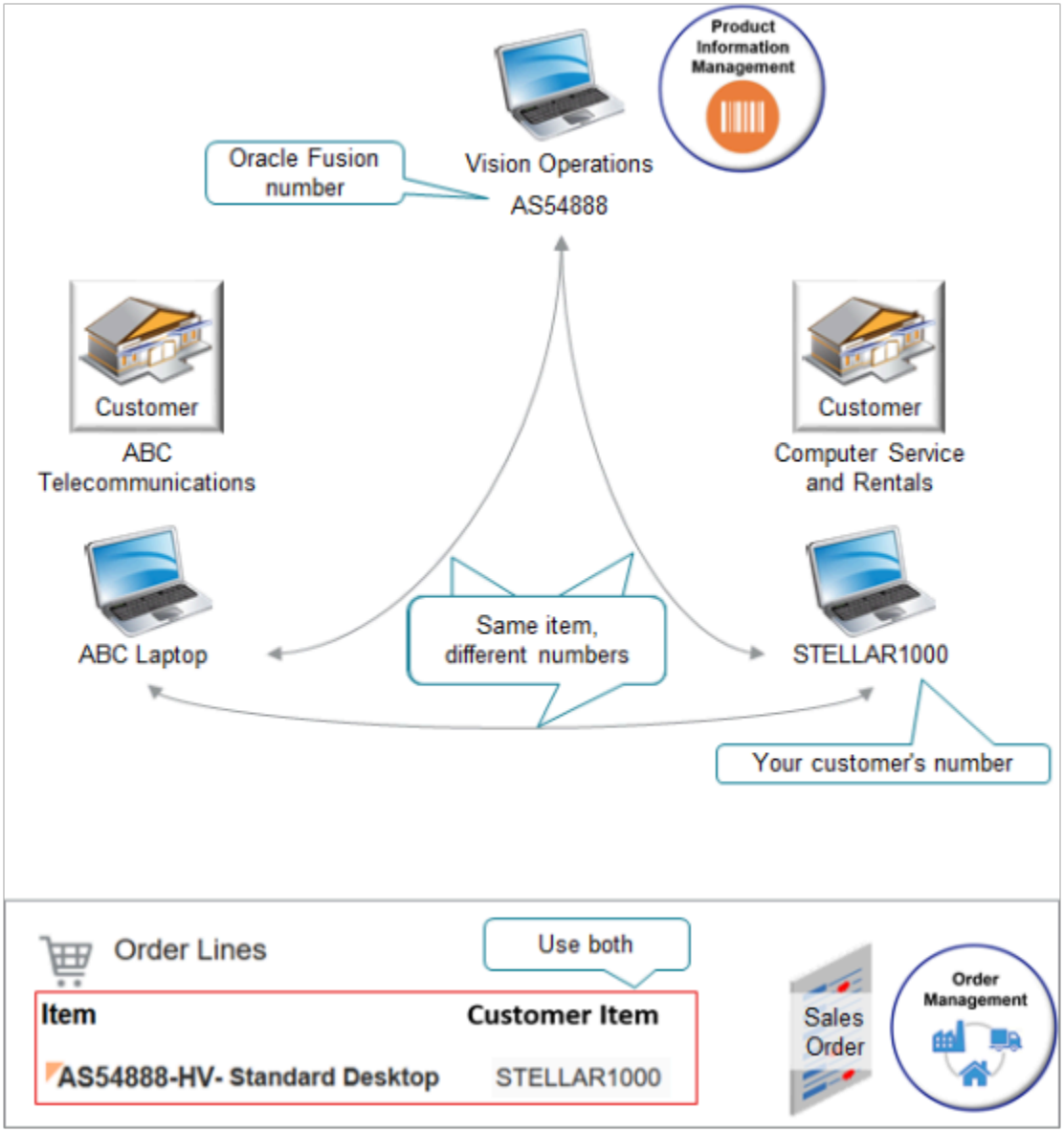
Related Topics

- Overview of Importing Orders Into Order Management
- Set Up Coverages for Sales Orders
- Manage Order Management Parameters
- Connect Order Management to Your Fulfillment System

Customer Items

Customer Items in Order Management

Use the Customer Part Number attribute to represent the customer item that your customer uses when that number is different from what Oracle Applications use.



Note

- A customer item is a number that your customer uses to represent an item when they need to use a number that's different from what Oracle uses. Your customers might use different numbers to identify the same item.
- For example, assume you set up an item in the Product Information Management work area and set the Item attribute to AS54888. You add the AS54888 to an order line when you create a sales order. However, your Computer Service and Rentals customer uses their own part number to brand the AS54888. They use part number STELLAR1000. You also sell the AS54888 to another customer, ABC Telecommunications, but they use the value ABC Laptop to brand the AS54888. STELLAR1000 and ABC Laptop are each an example of a customer item.
- You can set up Order Management so it includes the Oracle number and the customer number on the order line.
- You can use whatever value for the customer number that your supply chain requires, such as a trading partner item.

Realize these benefits.

- Reduce the amount of time that you need to create each sales order.
- Reduce the errors that might happen when you create a sales order.
- Search for sales orders more efficiently.
- Track orders in order fulfillment more efficiently.

You can use a customer item:

- In your back-to-back flows and drop shipment flows.
- When you create a referenced or an unreferenced return.
- With file-based data import, a web service, or REST API. For details, see *Import Customer Items Into Order Management*.
- With reports. Include the Customer Item and Customer Item Description in reports and analytics.
- When you create a pretransformation rule in Visual Information Builder.
- With a processing constraint to constrain changes that the Order Entry Specialist can make to a sales order, according to the customer item.
- With your downstream systems, such as Oracle Shipping and, for drop shipments, Oracle Procurement.

If you have a sales order that's open, then don't delete or end date the customer item relationship in Product Information Management until you close the order.

For more, see *Set Up Customer Items for Order Management*.

Use the Order Management Work Area

Assume you add the AS54888 to an order line when you create a sales order. However, your Computer Service and Rentals customer uses their own part number to brand the AS54888. They use part number STELLAR1000. You also sell the AS54888 to another customer, ABC Telecommunications, but they use the value ABC Laptop to brand the AS54888. STELLAR1000 and ABC Laptop are each an example of a customer item.

- Use **View > Columns** to display the Customer Item column in Order Management pages the first time that you log in after you turn on the feature.
- Use the customer item when you search for and add an item on the catalog line in the Order Management work area.
- Select and add a customer item when you use the Create Order page, Revise Order page, or the Unreferenced Returns dialog.
- The Customer Item column on the order line displays the customer item number and the customer item description, separated by a hyphen. For example, if the customer item number is `STELLAR1000` and the customer item description is `Cosmic Laptop Computer`, then the Customer Item column displays `STELLAR1000 - Cosmic Laptop Computer`.
- You can view the Customer Item on the Order Lines tab, Shipment Details tab, or the Billing and Payment Details tab.
- You can use a customer item with a standard item, coverage item, subscription, kit, or model. If your item isn't configured, then the search and select dialog will display the customer item when you specify these details on the order line. If your item is configured as an assemble-to-order item, pick-to-order item, or a kit, then only the top model can include the customer item. You can't include the customer item in any child lines.

- Use the Customer Item attribute or the Customer Item Description attribute to search for sales orders on the Manage Orders, Manage Fulfillment Lines, or Manage Return Fulfillment Lines pages. For example, go to the Manage Fulfillment Lines page, open Advanced Search, add the Customer Item attribute, then search on it.

Order Management displays details about the customer item according to the value that you set in the Customer attribute on the order header.

If an Item	Then You Will See
Doesn't have a customer item relationship for this customer.	An empty Customer Item Number column in the search results section of the search and select dialog that you use when you specify customer details on the order line.
Has more than one customer item relationship for this customer.	A separate row for each customer item in the search results.

Assume you set up customer item STELLAR1000 and customer item ABC Laptop for Oracle item AS54888 for your Computer Service and Rentals customer. If you search for AS54888, you will get two search results. You select either one.

Item	Customer Item
AS54888	STELLAR1000, Cosmic Laptop Computer
AS54888	ABC Laptop, ABC's Best Laptop

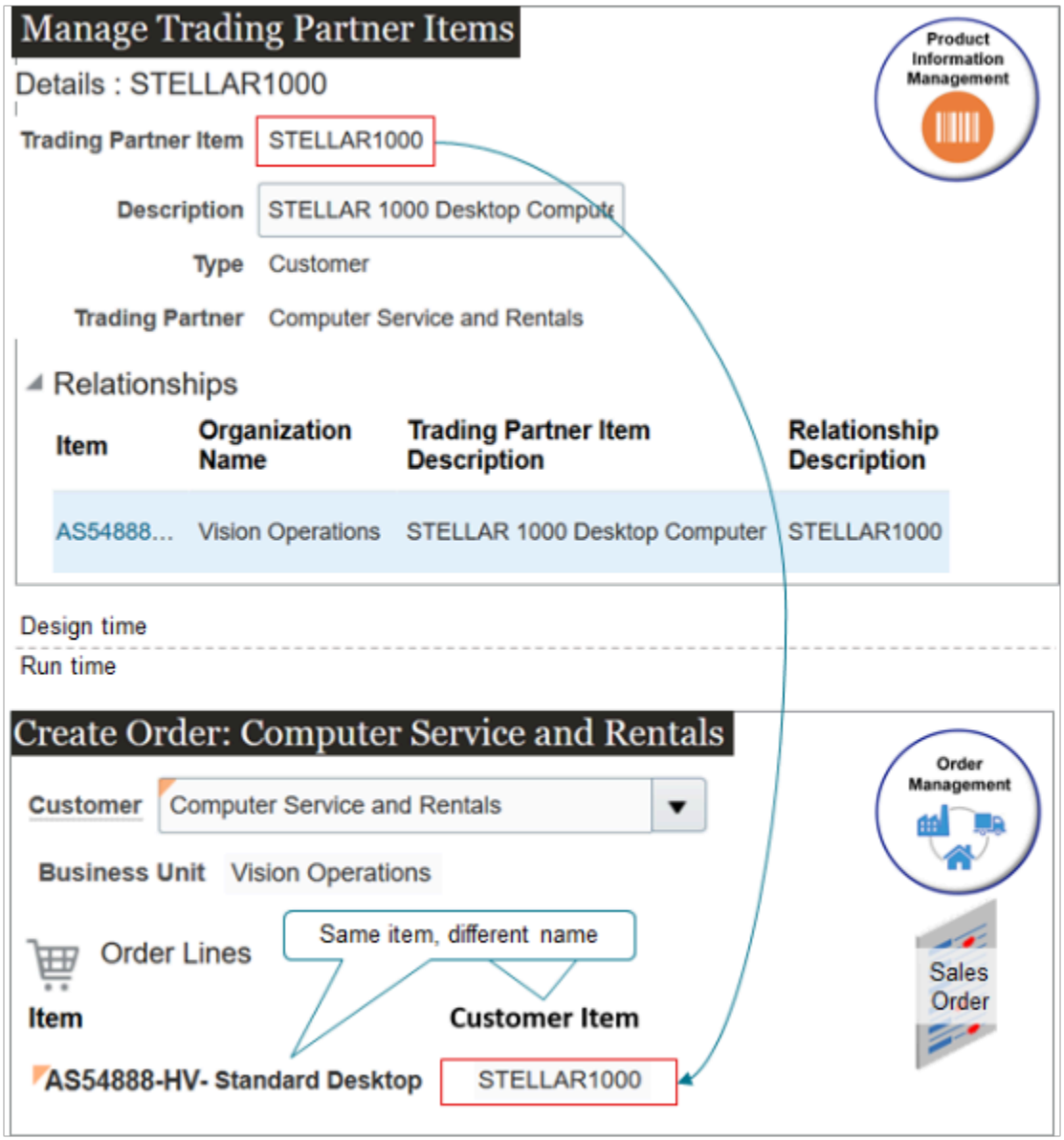
Related Topics

- Privileges That You Need to Implement Order Management
- Import Customer Items Into Order Management
- Use Order Profiles to Control Order Management Behavior
- Create Items
- Types of Item Relationship

Set Up Customer Items for Order Management

Set up Order Management so you can use the Customer Item attribute to represent the customer item.

Assume you must create a relationship between the STELLAR1000 customer item and the AS54888 Oracle item.



Note

- You use the Manage Trading Partner Items page in the Product Information Management work area to create a relationship between an Oracle item and your customer item. A customer item is a trading partner item where the Type equals Customer.
- Order Management uses your set up from Product Information Management to determine which item to add to the order line.
- Order Management gets the value from the Trading Partner Item attribute in Product Information Management and displays it in the Customer Item attribute on the order line.

Summary of the Setup

1. Set up the feature.
2. Set up search for your customer item.
3. Create your Oracle items.
4. Create relationships between Oracle items and customer item.

5. Test your setup.

Set Up the Feature

1. Make sure you have the privileges that you need to administer Product Information Management.

   If you don't have these privileges, then the Product Information Management work area won't display your product details and you won't be able to do this procedure. For details, see *Privileges That You Need to Implement Order Management*.
2. Enable the features.
   - Go to the Setup and Maintenance work area.
   - On the Setup page, select the Order Management offering, then click **Change Feature Opt In**.

     On the Opt In page, expand **Order Management**, then in the row that contains Items in the Name column, click the **pencil**.

     Expand **Search and Select Items More Efficiently**, then add a check mark to the Enable attribute for the features.

Feature	Description
Search and Select Items More Efficiently	Allows you to use the customer item feature with Order Management. Allows the Order Management work area to automatically suggest items when the Order Entry Specialist enters a value in the Customer Item attribute. For example, if the user enters STEL, then the work area will display all customer items and items from Product Information Management that include the value STEL, such as STELLAR1000, filtered according customer and business unit. If you don't enable this feature, then you can't use the customer item feature with Order Management. Also, the work area won't automatically suggest values, and the user must manually search for and select the item.
Use Your Customer's Part Number to Manage Sales Orders	Allows you to use the customer item feature with Order Management.

   - Click **Done > Done**.
3. Go to the Setup and Maintenance work area, then click **Tasks > Search**.
4. Search for, then open the Manage Profile Options task.
5. On the Manage Profile Options page, search for the value.

Attribute	Value
Profile Display Name	Get and Set Values for Customer Items If you enable this option, and if the user doesn't select a customer item in the Search and Select dialog that displays when the user clicks Add on the catalog line, add if you added more than

Attribute	Value
	one customer item relationship for the Oracle item in the Product Information Management work area, then Order Management will display a message. For example: `There is more than one instance of a customer item for item AS92111 and customer ABC Telecommunications. Click OK to add the item without selecting an instance of the customer item. To select an instance, click Cancel, use the magnifying glass to search for the item on the catalog line, then use the Search and Select dialog to select an instance.` If the user clicks OK, then Order Management will automatically get and set the value for the customer item according to the Sold-To Party and Customer Id on the sales order.

6. In the search results, in the Profile Values area, set the value.
   The profile comes predefined as Yes. If you want to disable it, set the value to No.

Set Up Search For Your Customer Item

You must create an index for your customer item. The index allows you to search for and select the customer item in the Order Management work area.

1. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Items
   - Task: Manage Item Keyword Search Attributes
2. On the Manage Item Keyword Search Attributes page, click **Actions > Select and Add**.
3. In the dialog that displays, in the Available Attributes area, use Query by Example to filter the results.

Attribute	Value
Keyword Group	Customer Items

4. Click **Add > OK**.

For details, see *Build Item Keyword Index*.

You can also manage changes that you make to the index over time. For details, see *Item Keyword Search Scheduled Process Actions*.

Create Your Oracle Item

1. Go to the Product Information Management work area.
2. Click **Tasks > Create Item**, set the values, then click **Save**.

Attribute	Value
Organization	Vision Operations

Attribute	Value
Item	AS54888
Description	Standard Desktop Computer

For details, see *Create Items*.

Create Relationships Between Oracle Items and Customer Item

1. Click **Tasks > Manage Trading Partner Items**.
2. On the Manage Trading Partner Items page, click **Create Trading Partner Item** (the plus icon).
3. In the Create Trading Partner area, set the values.

Attribute	Value
Trading Partner Item	STELLAR1000
Description	STELLAR 1000 Desktop Computer
Type	Customer
Trading Partner	Computer Service and Rentals
Status	Active

4. Create a relationship between the STELLAR1000 and the AS54888.
    - Expand the Relationships area, then click **Actions > Create**.
    - In the Create Customer Item Relationship dialog, set the values, then click **OK**.

Attribute	Value
Organization	V1 V1 is an abbreviation for Vision Operations.
Item	AS54888

Attribute	Value
Relationship Description	STELLAR1000

5. Click **Save > Done**.

Test Your Setup

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

2. On the catalog line, search for STELLAR1000. You created a relationship for more than one item, and you set the Get and Set Values for Customer Items profile option to True, so verify that Order Management displays a dialog asking whether you want to select an instance.
3. In the dialog, click **OK**, then verify that Order Management adds the AS54888.
4. Verify that the order line includes the Customer Item attribute, and that the attribute contains STELLAR1000.
5. Click Submit, then verify that the Manage Orders page and the Manage Fulfillment Lines page displays the Customer Item attribute, and that the attribute contains STELLAR1000.

Update the Customer Item

If you update the Trading Partner Item attribute in Product Information Management, then Order Management will update the Customer Item attribute on the sales order.

Consider an example.

1. You set the Trading Partner Item attribute to STELLAR1000 in Product Information Management on Monday.
2. You create and submit sales order 57687 on Tuesday. Order Management ships and closes the item. The Customer Item attribute on the fulfillment line contains STELLAR1000.
3. You change the Trading Partner Item attribute to STELLAR1001 in Product Information Management on Wednesday.
4. You query for 57687 on Thursday in Order Management, and notice that the fulfillment line now contains STELLAR1001.

Updating the Trading Partner Item attribute doesn't affect any other part of the sales order.

Import Sales Orders

You can use these attributes when you import the original order, then use them again when you import the revision.

Technology	Attribute
Order Import Service web service or REST API	CustomerProductNumber

Technology	Attribute
File-Based Data Import	Customer Product Number

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Related Topics

- Privileges That You Need to Implement Order Management
- Import Customer Items Into Order Management
- Use Order Profiles to Control Order Management Behavior
- Create Items
- Types of Item Relationship

Accounting

Track Items as Assets in Order Management

Specify how to track an item as an asset in Order Management.

1. Go to the Product Information Management work area.
2. On the Product Information Management page, click **Tasks > Manage Items**.
3. Locate the item that you must modify, then open it for editing.
4. On the Edit Item page, click **Specifications > Service**.
5. Set the value.

Attribute	Value
Enable Asset Tracking	Set to one of these values: o Full Lifecycle o Customer Asset For details, see *Monitor Order Fulfillment*.

6. Click **Sales and Order Management**, then set the value, or leave it empty.

Attribute	Value
Sales Product Type	Use one of these values: o Goods o Software

If the Sales Product Type attribute contains one of these values, then Oracle Installed Base ignores the item even if you set the Enable Asset Tracking attribute to Customer Asset or to Full Lifecycle:

- Included Warranty
- Extended Warranty
- Service Level Agreement
- Software Maintenance
- Preventive Maintenance
- Installation
- Training
- Subscription
- One Time Service

7. Optional. The DOO_AssetManagement task type allows Order Management to send a request to Oracle Installed Base to create or update an asset. You can set up your own task that DOO_AssetManagement references, then reference your task from various steps in the different orchestration processes that you create. For details, see *Create Your Own Task Type*.

   Create your own task type:

   - Go to the Setup and Maintenance work area, then go to the task.
     - Offering: Order Management
     - Functional Area: Orders
     - Task: Manage Task Types
   - On the Manage Task Types page, click **View > Query by Example**, enter the value, then click **Enter** on your keyboard.

Attribute	Value
Task Type	DOO_AssetManagement

   - In the DOO_AssetManagement Details area, click **Tasks > Actions > Add Row**, set the values, then click **Save and Close**.

Attribute	Value
Code	Create_Vision_Asset
Name	Create_Vision_Asset
Display Name	Create Vision Asset

8. Create an orchestration process or revise the copy of an existing one:
    - Add a step that references the task type.

Attribute	Value
Task Type	DOO_AssetManagement

      Add this step after the shipment step and before the invoicing step. For example, if you copy, then modify the copy of the predefined DOO_OrderFulfillmentGenericProcess orchestration process, then add this step immediately before the Create Invoice step.
    - If you created your own task type earlier in this procedure, then set the value.

Attribute	Value
Task	Create Vision Asset

Note

- If you sell covered items and coverage items, and if you have a step that calls the DOO_Subscription task, then you must set up the orchestration process so it runs the DOO_AssetManagement task to send the covered item to Oracle Installed Base before it runs the DOO_Subscription task to send the coverage item to Oracle Subscription Management.

  If you use a serial or a lot to control the item, and if you set the Enable Asset Tracking attribute to Customer Asset, then Oracle Asset Management will automatically create an asset for the item from the Sales Order Issue inventory transaction even if your orchestration process doesn't reference the DOO_AssetManagement task type. Oracle Inventory Management will process the transaction, and then Asset Management will create the asset in the customer location, but it won't set the customer's start date for the asset, and it won't create a warranty for the asset. If you need that start date and warranty, then make sure your orchestration process has a step that references the DOO_AssetManagement task type.

For background details, see *Overview of Assets*.

Related Topics

- Overview of Orchestration Processes
- Fulfillment Tasks
- Monitor Order Fulfillment

Set Up Business Units for Selling Profit Centers

Set the selling profit center on a sales order line that's different from the business unit on the order header so you can sell items that belong to more than one profit center in a single sales order.

- Order Management updates the selling profit center every time a downstream system updates the warehouse.

- If the user changes the warehouse in the Order Management work area, then Order Management doesn't update the selling profit center.
- Order Management prices the order line each time the value in the Selling Profit Center attribute on the order line changes.
- Order Management communicates the selling profit center to downstream applications, such as Shipping, Costing, Sales Financial Orchestration, Tax, and Receivables.
- You can reference the selling profit center in the Order Information Service, an order management extension, or in a service mapping.

Here are some notes about tax.

- Use this feature when you have more than one tax registration across regions, such as across states in the United States. Your users can set the business unit on the order line for the profit center but continue to use a single business unit in the order header.
- The Selling Profit Center affects the tax that Oracle Financials calculates for each order line according to tax rules that the taxing authority imposes and that you set up.
- Tax determinants depend on the selling profit center, so Order Management sets the default value for tax determinants each time it updates the business unit for the selling profit center.
- Learn how to set up tax. For details, see *Implementing Tax*.

Set up business units for selling profit centers.

1. Enable the Specify Business Unit for Selling Profit Center for Goods and Services Tax feature.
2. Set the Business Unit for Selling Profit Center parameter.

   Order Management will set the value of the Selling Profit Center attribute on the order line to the value that you specify in this parameter. It sets the attribute value when the Order Entry Specialist adds an order line to a sales order.

If You Set the Parameter To	Order Management Will
Order Management	Set the Selling Profit Center attribute on the order line to the same value that the Business Unit attribute on the order header contains.
Shipping Organization	Set the Selling Profit Center attribute on the order line to the business unit that the shipping inventory organization references. You can use this value for these flows. o Normal shipment. o Drop shipment. o back-to-back shipment. o Ship but don't bill. o Bill but don't ship. o Return material authorization.

   For details, see *Manage Order Management Parameters*.

Import Assessable Value

You can't use REST API with the Specify Business Unit for Selling Profit Center for Goods and Services Tax feature. For example, if you use REST API to import the assessable value but don't freeze the tax, then Pricing will overwrite whatever value that you import when it calculates the tax for your item. If you freeze the tax, then Pricing will use the assessable value that you import.

You can also use the Order Management work area to set the assessable value, or use some other technology to import it, such as file-based data import or a web service.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

For details, see *Freeze Price on Sales Orders* and *Import Tax on Sales Orders*.

Related Topics

- Opt Into Features in Order Management
- Manage Order Management Parameters

Tax

Apply Tax According to Customer Site

Set up tax so Order Management can add tax to a sales order according to your customer's physical site. This is the way you set up the default tax that Order Management applies on a sales order for most customers.

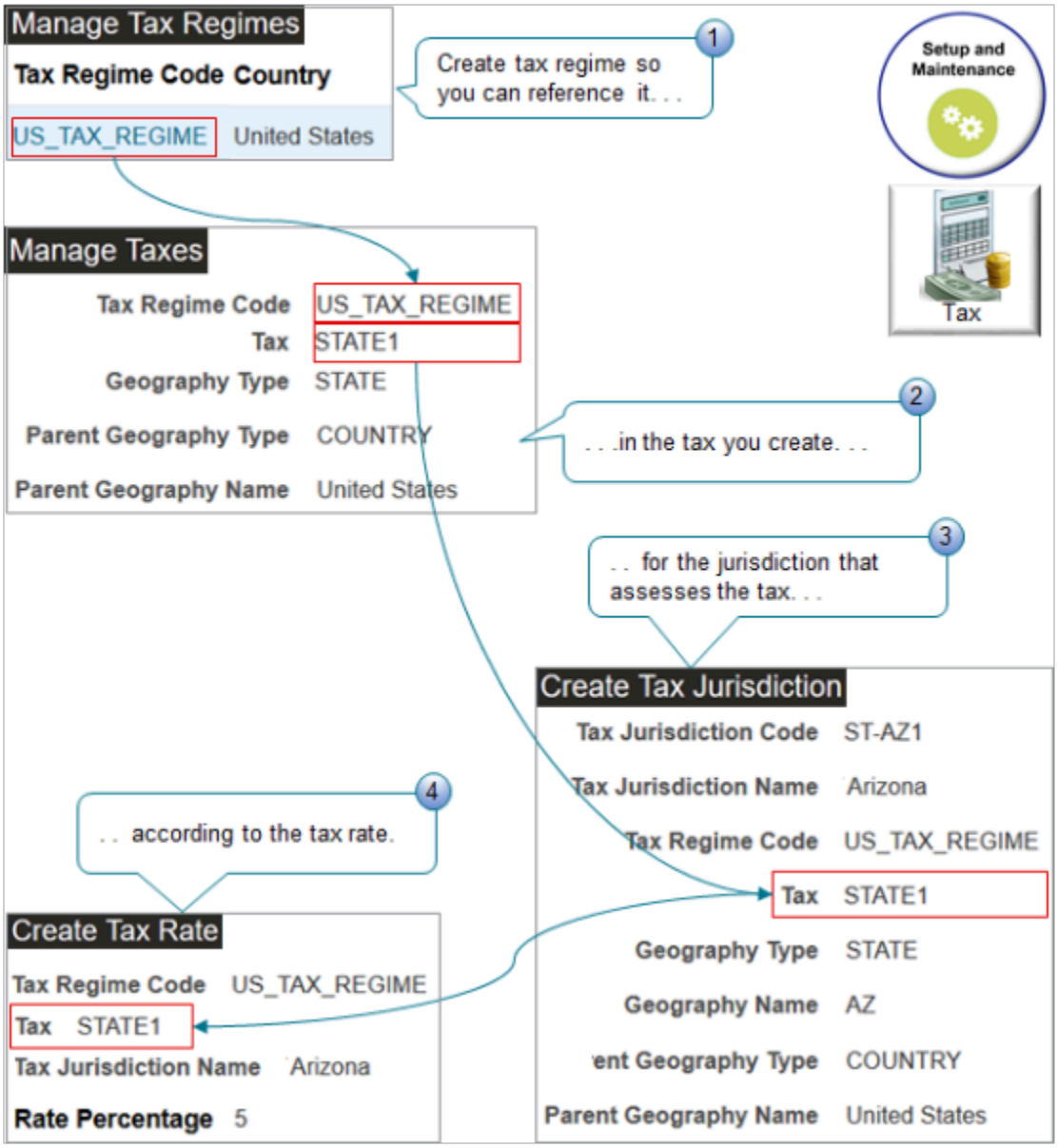
Here's an example of how you can set up tax.

Note

- A tax regime, such as the United States, contains tax jurisdictions, such as the Arizona Department of Revenue.
- The tax jurisdiction assess a tax on your sales according to its tax rate, and depending on your customer's address.
- For example, if the tax rate in the Arizona tax jurisdiction is 5%, and if the net price on the sales order is $100, then the tax is $5.

You have a wide range of choices when setting up tax, depending on your supply chain's tax requirements. For example, you can specify tax according to the tax jurisdiction that applies at the ship-to site, the bill-to site, or a range of other locations.

Here's the flow that you use to set up the example in this topic.



Here are the tasks you use.

Task	Description
1. Manage Tax Regimes	Create a tax regime named USA_TAX_REGIME for the United States.
2. Manage Taxes	Create a tax named STATE1 at the state level for Arizona.
3. Create Tax Jurisdiction	Create a tax jurisdiction for the state of Arizona.
4. Create Tax Rate	Create the 5% tax rate to apply for the STATE1 tax.

Each task maintains the tax hierarchy. For example:

- The tax rate is in the USA_TAX_REGIME and the Arizona jurisdiction.
- The Arizona jurisdiction and the STATE1 tax are in the USA_TAX_REGIME.

You create a relationship between your customer's site and the tax regime.

Summary of the Set Up

1. Create the tax regime.
2. Create the tax.
3. Create the tax status.
4. Create the tax jurisdiction.
5. Add the customer site.
6. Verify geography and collect data.
7. Test your set up.

For this example, assume.

- You are in the Vision Operations business unit.
- Your sales order must calculate tax according to the ship-to address on the sales order.
- You set up tax for site 1036 of your Computer Service and Rentals customer, and their ship-to address is 2164 Broadway Tempe, AZ 85282.
- The tax jurisdiction in Tempe charges a 5% tax.

For background details about how to set up tax, see *Administering Tax Reporting* on Oracle Help Center.

For other technical details, including lots of helpful screen prints, see *Order Management Tax Best Practices and General Questions (Doc ID 2619517.1)*.

Create the Tax Regime

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Regimes
2. On the Manage Tax Regimes page, click **Actions > Create**.
3. On the Create Tax Regimes page, set the values.

Attribute	Value
Tax Regime Code	US_TAX_REGIME
Tax Regime Name	US_TAX_REGIME
Regime Level	Country
Country	United States

Attribute	Value
Tax Currency	USD
Minimum Accountable Unit	0.01
Tax Precision	2
Tax Inclusion Method	Standard Noninclusive Handling
Allow Tax Rounding Override	Contains a check mark.
Allow Override and Entry of Inclusive Tax Lines	Contains a check mark. Enable this option so the Order Entry Specialist can override inclusive tax on the order line.

4. In the Configuration Options and Service Subscriptions area, on the Configuration Options tab, set values, then click **Save and Close > Done**.

Attribute	Value
Party Name	FUSION_AP You must set up your party before you do this procedure.
Party Type	First Party Legal Entity
Country	United States
Configuration for Taxes and Rules	Common Configuration with Party Overrides
Configuration for Product Exceptions	Common Configuration

Create the Tax

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Taxes
2. On the Manage Taxes page, click **Actions > Create**.
3. On the Create Tax page, set the values.

Attribute	Value
Tax Regime Code	US_TAX_REGIME
Configuration Owner	Global Configuration Owner
Tax	STATE1
Tax Name	STATE1
Geography Type	State
Parent Geography Type	Country
Parent Geography Name	United States
Override Geography Type	US_STATE_ZONE_TYPE_101
Tax Currency	USD
Tax Minimum Accountable Unit	0.01
Tax Precision	2
Conversion Rate Type	Corporate
Rounding Rule	Down

4. In the Controls and Defaults area, set the values.

Attribute	Value
Tax Inclusion Method	Standard Noninclusive Handling
Allow override and entry of inclusive tax lines Allow tax rounding override Allow override of calculated tax lines Allow entry of manual tax lines	Make sure each of these options contain a check mark. They allow Order Management to modify tax at run time.
Allow creation of multiple jurisdictions	Contains a check mark.
Allow tax exceptions	Contains a check mark.
Allow tax exemptions	Contains a check mark.

5. Click **Tax Rule Defaults**, then set the values.

Attribute	Value
Place of Supply	Ship to, Use Bill to If Not Found. You're setting tax according to the tax rate in the tax jurisdiction of the ship-to address. If Order Management can't find the ship-to address at run time, then it will use the tax rate that the tax jurisdiction applies at the bill-to address.
Tax Applicability	Applicable
Tax Registration	Ship-to Party
Tax Calculation Formula	STANDARD_TC TC means tax calculation.
Taxable Basis Formula	STANDARD_TB TB means taxable basis.

6. In the Indirect Defaults area, set the values.

Attribute	Value
Tax Jurisdiction	ST-AZ1 ST means state. AZ means Arizona.
Tax Status	STANDARD
Tax Rate	STD

Create the Tax Status

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Statuses
2. On the Manage Tax Statuses page, click **Actions > Create**.
3. On the Create Tax Status page, set the values, then click **Save and Close > Done**.

Attribute	Value
Tax Regime Code	US_TAX_REGIME
Configuration Owner	Global Configuration Owner
Tax	STATE1
Tax Status Code	STANDARD
Tax Status Name	STANDARD
Set as Default Tax Status Allow tax exceptions Allow tax exemptions	Contains a check mark.

Attribute	Value
Allow tax rate override	
Parent Geography Name	United States

Create the Tax Jurisdiction

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Jurisdictions
2. On the Manage Tax Jurisdictions page, click **Actions > Create**.
3. On the Create Tax Jurisdiction page, set the values.

Attribute	Value
Tax Jurisdiction Code	ST-AZ1
Tax Jurisdiction Name	Arizona
Tax Regime Code	US_TAX_REGIME
Tax	STATE1
Geography Type	STATE
Geography Name	AZ
Parent Geography Type	COUNTRY
Parent Geography Name	United States
Set as Default Jurisdiction	Contains a check mark.

4. Click **Save**.

   You must click save before you do the next step.
5. On the Associated Jurisdiction Tax Rate Periods tab, click **Create**.

6. On the Create Tax Rate page, set values.

Attribute	Value
Configuration Owner	Global Configuration Owner
Tax Status Code	STANDARD
Tax Rate Code	STD-DEFAULT
Tax Rate Type	Percentage
Order to Cash Procure to Pay Expenses	Contains a check mark.

7. In the Rate Periods area, set the values, then click **Save and Close**.

Attribute	Value
Rate Percentage	5
Set as Default Rate	Contains a check mark.

8. On the Edit Tax Jurisdiction page, notice that the Associated Jurisdiction Tax Rate Periods contains the STD-DEFAULT rate you just created, then click **Save and Close > Done**.

Create the Standard Tax Rate

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rates and Tax Recovery Rates
2. On the Manage Tax Rates and Tax Recovery Rates page, click **Actions > Create**.
3. On the Create Tax Rate page, set values.

Attribute	Value
Tax Regime Code	US_TAX_REGIME

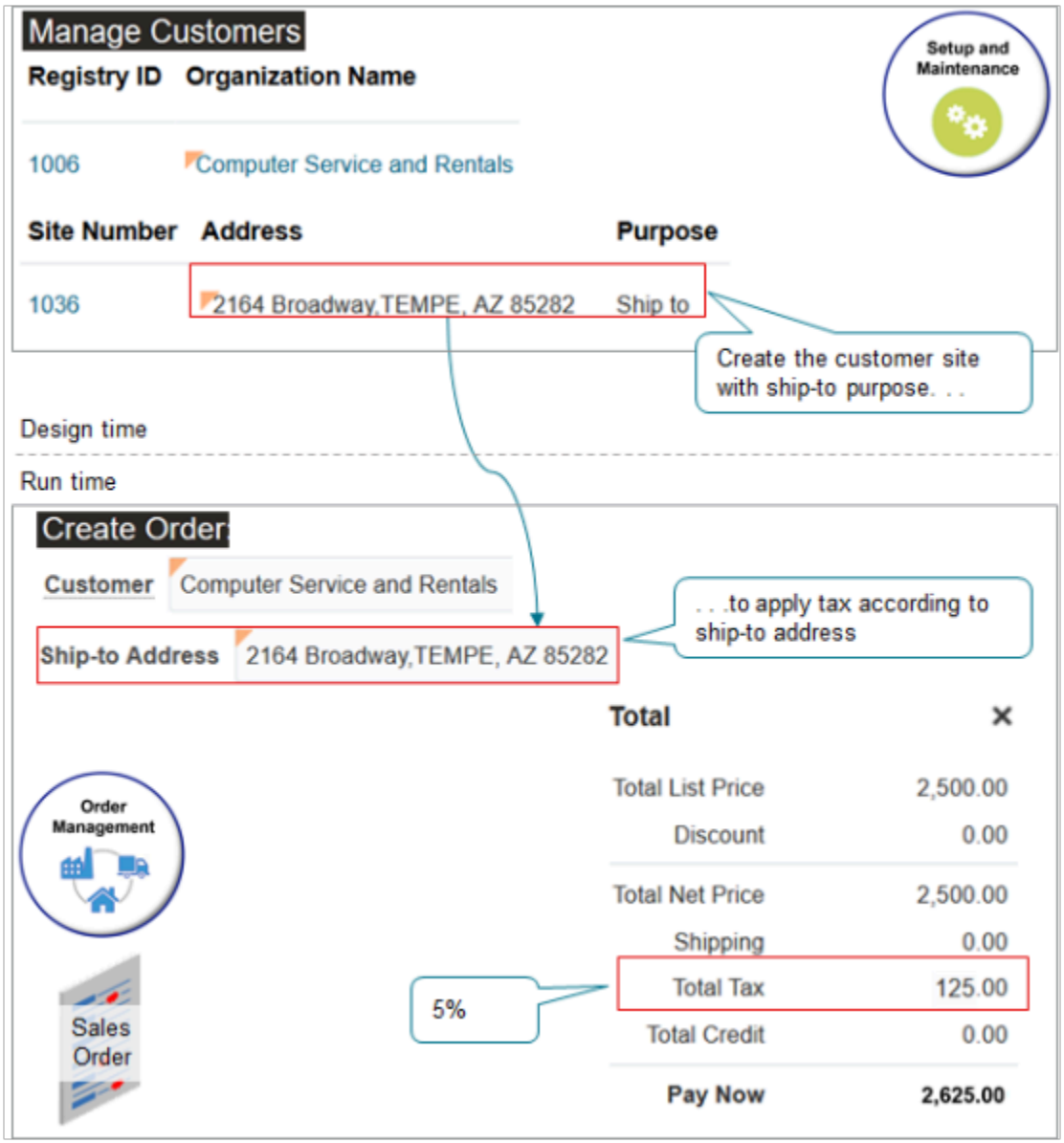
Attribute	Value
Configuration Owner	Global Configuration Owner
Tax	STATE1
Tax Status Code	STANDARD
Tax Jurisdiction Code	ST-AZ1
Tax Rate Code	STD STD means standard.
Tax Rate Type	Percentage
Order to Cash Procure to Pay Expenses	Contains a check mark.

4. In the Rate Periods area, set the values, then click **Save and Close > Done**.

Attribute	Value
Rate Percentage	10
Set as Default Rate	Doesn't contain a check mark.

Add the Customer Site

Use the Manage Customers task to create a site for your Computer Service and Rentals customer. You specify the purpose as ship-to so you can use the ship-to address on the sales order to determine the tax to apply.



Try it.

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Customers
    - Task: Manage Customers
2. On the Manage Customers page, search for the value.

Attribute	Value
Organization Name	Computer Service and Rentals

Attribute	Value

3. In the search results, in the Sites area, click **Actions > Create**.
4. On the Create Account Site page, set the values.

Attribute	Value
Account Address Set	Vision Operations Set
Country	United States
Site Number	1036
Address Line 1	2164 Broadway
City	TEMPE
State	AZ
Postal Code	85282

5. In the Address Purposes area, click **Actions > Add Row**, set the values, then click **Save and Close > Done**.

Attribute	Value
Purpose	Ship To

Verify Geography and Collect Data

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Enterprise Profile
    - Task: Manage Geographies

2. On the Manage Geographies page, search for the value.

Attribute	Value
Country Name	United States

3. In the search results, click the row that has United States in the Country Name column, then click **Actions > Manage Geography Validation**.
4. On the Manage Geography Validation page, in the Geography Mapping and Validation area, verify the values for the State.

Geography Type	Map to Attribute	Tax Validation
State	State	Contains a check mark.

Collect Data

1. Go to the Plan Inputs work area.

   Don't use the Plan Inputs task that's available in the Setup and Maintenance work area. Use the Plan Inputs work area instead.
2. In the Plan Inputs work area, click **Tasks > Collect Planning Data**.
3. In the Collect Planning Data dialog, set your source system, then move reference entities to selected entities.
   - Customer
   - Geographies
   - Organizations
4. Click **Submit**.

For details, see *Collect Planning Data for Order Management*.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Ship-to Address	2164 Broadway, Tempe, AZ, 85282

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.
4. Click the **total** at the top of the sales order, then verify that the tax rate is 5%, which is the STD-DEFAULT rate you set up according to the ship-to address.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Tax	125.00 125 is 5% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,625.00

5. Change the ship-to address to any other value, then click **Actions > Reprice Order**.
6. Click the **total** at the top of the sales order, then verify that the tax rate is 10%, which is the STD rate you set up to apply when the ship-to address isn't 2164 Broadway, Tempe, AZ, 85282.

Attribute	Value
Total List Price	2,500.00
Discount	0.00

Attribute	Value
Total Net Price	2,500.00
Shipping	0.00
Total Tax	250.00 250 is 10% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,750.00

Related Topics

- Overview of Collecting Promising Data for Order Management

Apply Tax According to Customer

Calculate tax according to the customer you select in the Customer attribute on the sales order.

Summary of the Set Up

1. Modify the customer set up.
2. Verify the party tax profiles.
3. Set up another customer.
4. Specify the default taxes.
5. Test your set up.

For this example, assume.

- You are in the Vision Operations business unit.
- Your sales order must calculate tax according to the Computer Service and Rentals customer.

Assume you already set up tax so Order Management applies it according to the customer site.

- You set up tax for site 1036 of your Computer Service and Rentals customer.
- The tax jurisdiction in Tempe charges a 5% tax.
- Tax is STATE1 at the state level for Arizona.
- Tax jurisdiction is for the state of Arizona.
- Tax rate is 5%.

For details, see *Apply Tax According to Customer Site*. You will reuse some of the objects that you already set up in that topic, such as the tax rate.

Modify the Customer Set Up

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Customers
    - Task: Manage Customers
2. On the Manage Customers page, search for the value.

Attribute	Value
Organization Name	Computer Service and Rentals

3. In the search results, in the Organization Name column, click **Computer Service and Rentals**.
4. On the Edit Customer page, click **Tax Profile**, set the value, then click **Save and Close**.

Attribute	Value
Tax Classification Code	6% TAX RATE Use upper case. The value is case sensitive.

5. On the Manage Customers page, in the Sites area, in the Site Number column, click **1036**.
6. On the Edit Site page, click **Tax Profile**, set the value, then click **Save and Close > Done**.

Verify the Party Tax Profiles

Verify that the tax classification code in the party tax profiles default to the value you set on the Manage Customers page.

1. On the setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Party Tax Profiles
2. On the Manage Party Tax Profiles page, set Search For to Third-Party Tax Profiles, then search for the value.

Attribute	Value
Party Name	Computer Service and Rentals

3. In the search results, click the value in the Party Number column.

4. On the Third-Party Tax Profile page, verify the value, then click **Done**.

Attribute	Value
Tax Classification Code	6% TAX RATE

5. On the Manage Party Tax Profiles page, set Search For to Third-Party Site Tax Profiles, then search for the value.

Attribute	Value
Party Name	Computer Service and Rentals

6. On the Third-Party Site Tax Profile page, in the search results, in the Party Site Number column, click **1036**.

   Assume 1036 is the site for the Arizona location.
7. On the Third-Party Site Tax Profile page, verify the value, then click **Done > Done**.

Attribute	Value
Tax Classification Code	6% TAX RATE

Set Up Another Customer

Let's say you have another customer named Computer Associates. Repeat the set ups you just made above but this time do the set up for Computer Associates, and set the tax rate to 5%.

Specify the Default Taxes

Specify the sequence to use when determining the default tax to apply. This helps to make sure Order Management applies a tax even if there's a problem with applying tax according to customer.

1. On the Setup page, click **Search**, search for, then open the Manage Application Tax Options task.
2. On the Manage Application Tax Options page, search for the values.

Attribute	Value
Business Unit	Vision Operations
Application Name	Receivables

3. In the search results, set the values, then click **Save and Close**.

Attribute	Value
Enable Defaulting Order	Contains a check mark.
Defaulting Order 1	Customer
Defaulting Order 2	Customer Site
Defaulting Order 3	Product

Here's the logic that this set up implements.

1. Use the set up you just made to apply tax according to the customer you select on the sales order.
2. If the tax set up according to customer isn't available, then apply tax according to the customer site.
3. If the tax set up for the customer site isn't available, then apply tax according to the item you add to the order line. Use the set up in the Product Information Management work area to apply the tax.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.

4. Click the **total** at the top of the sales order, then verify that the tax rate is 6%, which is the rate you set for your Computer Service and Rentals customer.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Tax	150.00 150 is 6% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,650.00

5. Create another sales order.

Attribute	Value
Customer	Computer Associates
Business Unit	Vision Operations

6. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

7. Click **Actions > Reprice Order**.

8. Click the **total** at the top of the sales order, then verify that the tax rate is 5%, which is the rate you set for your Computer Associates customer.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Tax	125.00 125 is 5% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,625.00

Related Topics

- Apply Tax According to Customer Site

Apply Tax Before You Calculate a Discount

Apply tax before you calculate a discount on the item.

Some discounts are taxable while others aren't, depending on the tax authority where you apply the tax. Either way, you can set up tax to meet your tax requirements.

Summary of the Setup

1. Create taxable basis formula.
2. Create a tax determining factor set.
3. Create the condition set.
4. Create the tax rule.
5. Test your set up.

Create Taxable Basis Formula

1. Create a discount list in the Pricing Administration work area, assign it to your item, then add it to a pricing strategy.

For details, see *Manage Discount Lists*.

2. Do the tax set up. For details, see *Apply Tax According to Customer Site*.

   You will reuse a number of objects from that topic, such as US_TAX_REGIME, the 5% tax rate, and the STATE1 tax.
3. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Financials
   - Functional Area: Transaction Tax
   - Task: Manage Tax Formulas
4. On the Manage Taxable Basis Formulas page, click **Create**.
5. On the Create Taxable Basis Formula page, set the values, and then click **Save and Close > Done**.

Attribute	Value
Configuration Owner	Global Configuration Owner
Tax Formula Code	Tax Before Discount
Tax Formula Name	Tax Before Discount
Taxable Basis Type	Assessable Value
Tax Regime Code	US_TAX_REGIME
Tax	STATE1

Create a Tax Determining Factor Set

1. On the Setup page, go to the task.
   - Offering: Financials
   - Functional Area: Transaction Tax
   - Task: Manage Tax Determining Factor Sets
2. On the Manage Tax Determining Factor Sets page, click **Create**.
3. On the Create Tax Determining Factor Set page, set the values.

Attribute	Value
Tax Determining Factor Set Code	TAX_BEFORE_DISCOUNT

Attribute	Value
Tax Determining Factor Set Name	TAX_BEFORE_DISCOUNT
Set Usage	Tax Rules
Tax Regime Code	US_TAX_REGIME

4. In the Associate Tax Determining Factors area, set the values, then click **Save and Close > Done**.

Attribute	Value
Determining Factor Class	Transaction Generic Classification This value specifies that you will use the input from a transaction to set the tax rate. A sales order is a type of transaction.
Determining Factor Name	Transaction Business Category

Create the Condition Set

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Condition Sets
2. On the Manage Tax Condition Sets page, click **Create**.
3. On the Create Tax Condition Sets page, set the values.

Attribute	Value
Tax Condition Set Code	DISCOUNT CONDITION SET
Tax Condition Set Name	DISCOUNT CONDITION SET
Tax Determining Factor Set Code	TAX_BEFORE_DISCOUNT

4. In the Tax Condition Set Details area, set the values, then click **Save and Close > Done**.

Attribute	Value
Tax Determining Factor Class	Transaction generic classification
Tax Determining Factor Name	Transaction Business Category
Operator	Equal To
Value or Start From Range	SALES_TRANSACTION

Create the Tax Rule

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rules
2. On the Manage Tax Rules page, click **Actions > Create**.
3. On the Create Determine Tax Applicability Rule page, set the values.

Attribute	Value
Configuration Owner	Global Configuration Owner
Tax Regime Code	US_TAX_REGIME
Tax	STATE1
Rule Code	TAX_BEFORE_DISCOUNT
Rule Name	TAX_BEFORE_DISCOUNT

4. In the Tax Determining Factor Set area, in the Code attribute, set the value, then click **Next**.

Attribute	Value
Code	TAX_BEFORE_DISCOUNT

5. On the Create Determine Tax Applicability Rule page, in the Tax Condition Set area, set the values, then click **Save and Next**.

Attribute	Value
Tax Condition Set Code	DISCOUNT CONDITION SET
New Condition Set Order	1
Result	Applicable
Enabled	Contains a check mark.

6. On the Edit Determine Tax Applicability Rule page, in the Rule Status and Order area, add a check mark to the Enabled attribute, then click **Submit > Done**.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.

4. Click the total at the top of the sales order, then verify that Order Management applies the tax rate according to the total list price, not the net price.

Attribute	Value
Total List Price	2,500.00
Discount	250.00
Total Net Price	2,250.00
Shipping	0.00
Total Tax	125.00 125 is 5% of 2500.
Total Credit	0.00
Pay Now	2375.00

Related Topics

- Apply Tax According to Customer Site
- Manage Discount Lists

Apply Tax After You Calculate a Discount

Apply tax after you calculate a discount on the item.

Create a discount list in the Pricing Administration work area, assign it to your item, then add it to a pricing strategy. For details about how to do this, see *Manage Discount Lists*.

At run time, Order Management calculates the discount first, and then applies the tax depending on your tax setup.

Assume you set up pricing so it calculates a 10% discount, and you set up tax to apply a 5% tax according to customer site. For details, see *Apply Tax According to Customer Site*.

Here's how Order Management prices the sales order.

Attribute	Value
Total List Price	2,500.00

Attribute	Value
Discount	250.00
Total Net Price	2,250.00
Shipping	0.00
Total Tax	112.50 112.50 is 5% of 2,250.
Total Credit	0.00
Pay Now	2362.50

Related Topics

- Apply Tax According to Customer Site
- Manage Discount Lists

Change Your Tax Rate

Change the tax rate that you apply on all new sales orders.

Assume your local taxing authority changes their tax rate from 6% to 6.5%, starting on October 9, 2020.

If the Ordered Date on the sales order happens.

- Before October 9, 2020, use the 6% rate.
- On or after October 9, 2020, use the 6.5% rate.

Assume you currently use a Tax Classification Code named A-RATE. You want to continue to use A-RATE but change its percentage from 6 to 6.5.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rates and Tax Recovery Rates

2. On the Manage Tax Rates and Tax Recovery Rates page, search for the value.

Attribute	Value
Tax Regime Code	VISION_TAX_REGIME
Tax	A-TAX
Tax Jurisdiction Code	Is Blank
Tax Rate Code	A-RATE

To get the values you need to search on, click **Actions > Export to Excel**. In Excel, locate your tax classification code in the Tax Rate Code column, then use values in the row that contains your tax classification code.

3. In the search results, click the value.

Attribute	Value
Tax Rate Code	A-RATE

4. On the Tax Rate page, in the Rate Periods area, notice the values.

Attribute	Value
Rate Percentage	6
Effective Start Date	31/01/20/ where o 31 is the day o 01 is the month o 20 is the year

This is the rate that's currently in effect. You will change it to 6.5.

5. Click **Edit**.
6. On the Edit Tax Rate page, in the Rate Periods area, set the values.

Attribute	Value
Rate Percentage	6

Attribute	Value
Effective End Date	8/10/20

7. Click **Add Row**, then set the values.

Attribute	Value
Rate Percentage	6.5
Effective Start Date	9/10/20

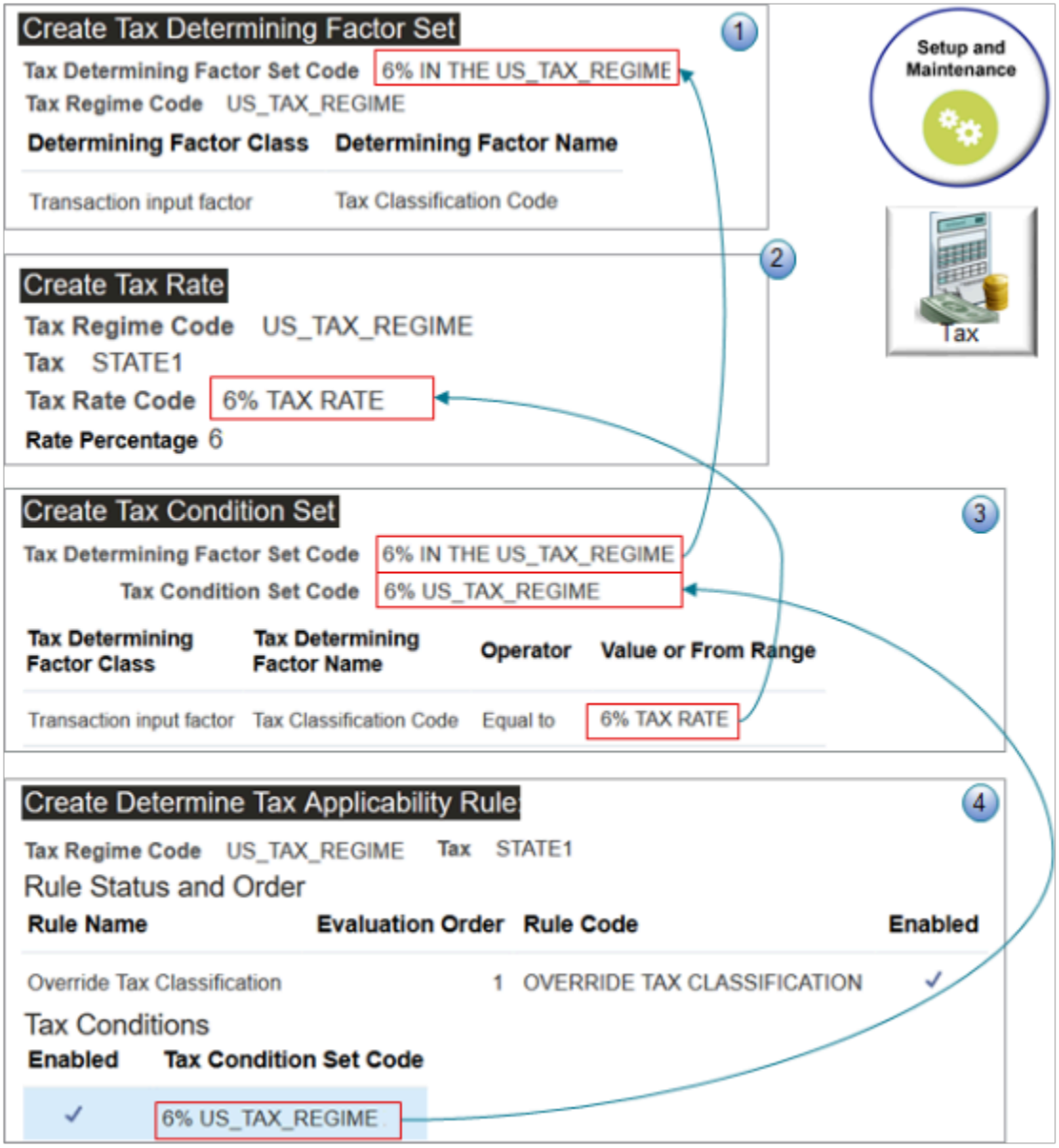
8. Verify your rates, then click **Save**.

Rate Percentage	Effective Start Date	Effective End Date
6.5	9/10/20	Empty
6	31/01/20/	8/10/20

Let Your Users Set the Tax Rate

Set up Order Management so the Order Entry Specialist can set the tax rate for a sales order.

The Order Entry Specialist can use the Tax Classification Code attribute on the order line to set the tax rate to apply for the line. Tax Classification Code is just a fancy name for a tax rate. For example, if the Order Entry Specialist sets the Tax Classification Code to 4%, then Order Management ignores any other tax that you have set up for the sales order, such as according to the customer site, and applies a 4% tax instead.



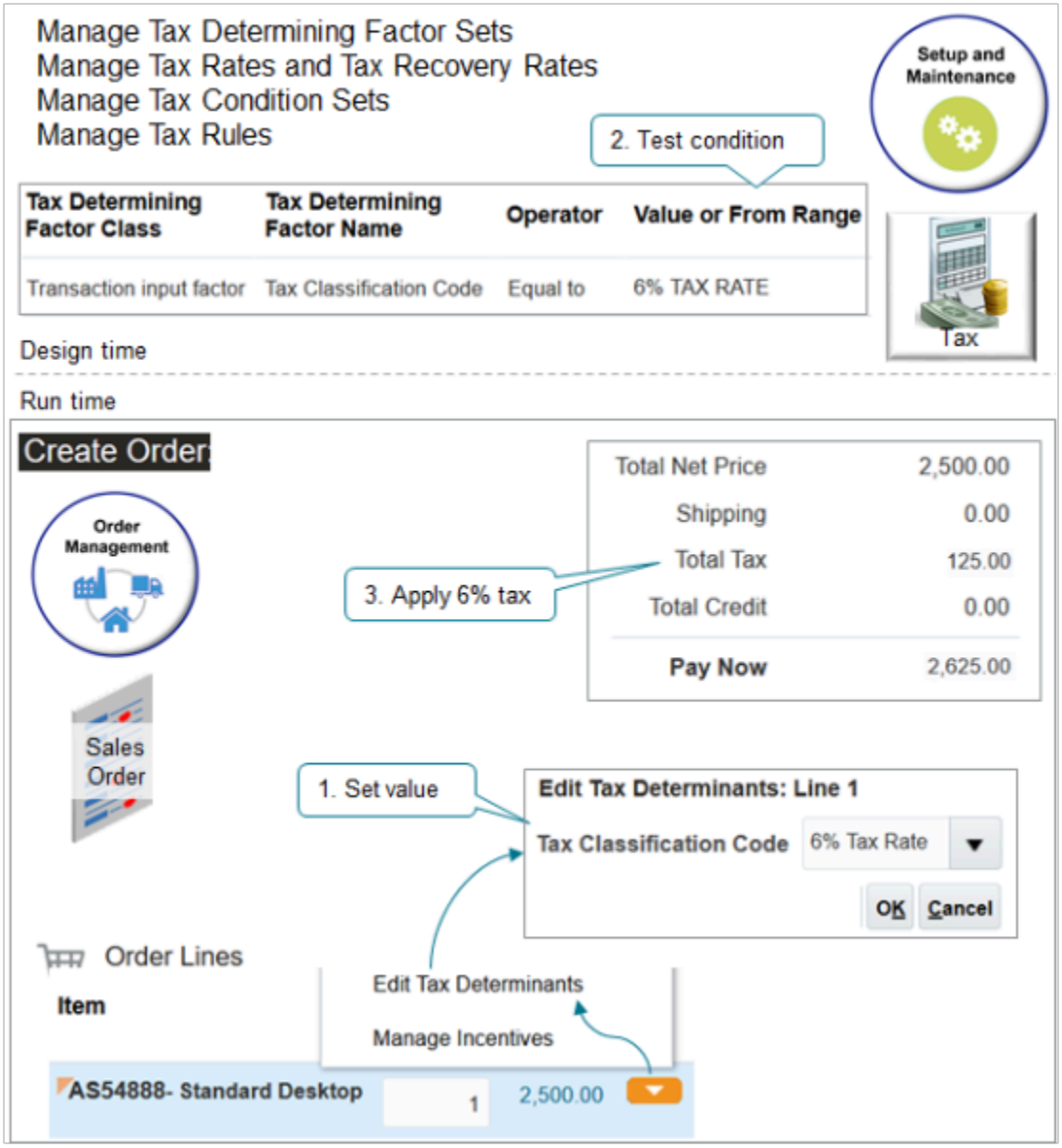
What the Numbers Mean

You use different tasks in the Setup and Maintenance work area.

-	Task	Value
1	Manage Tax Determining Factor Sets	Create a set of tax determining factors. A tax determining factor is a value that contributes to determining tax. The Tax Classification Code attribute on an order line is an example of a tax determining factor. This task is most useful in more complex implementations where you have more than one factor. However, you must create a set even if you only have one factor.

-	Task	Value
		For details, go to *Implementing Tax*, then search for Tax Determining Factor Sets, or search for Condition Sets.
2	Manage Tax Rates and Tax Recovery Rates	Create the tax rate you will apply when the Order Entry Specialist sets the value for the Tax Classification Code attribute. Notice that the determining factor set, tax rate, and tax rule are all in the same regime, the US_TAX_REGIME. Keep this in mind when you do your set up. You must use the same regime in these tasks so you can select the correct values in the attributes in each task.
3	Manage Tax Condition Sets	Specify the condition that determines whether to use the tax determining factor. You reference the factor in the header of the condition set, and you reference the tax rate in the condition. Here's the pseudocode for the condition you set up in this topic. • **`If the Order Entry Specialist sets the value of the Tax Classification Code attribute on the order line to 6% TAX RATE, then use the 6% IN THE US_TAX_REGIME factor set.`**
4	Manage Tax Rules	Specify the sequence to use when evaluating conditions and rules if you have more than one condition or more than one rule. This task is most useful in more complex implementations where you have more than one condition or rule. However, you must create a tax rule even if you only have one condition and one rule.

Here's what happens at run time.



An Order Entry Specialist creates a sales order in the Order Management work area, adds an order line, and sets a value in the Tax Classification Code attribute. The adjustment is an input from a transaction, the determining factor is the Tax Classification Code attribute, and the value is 6% Tax Rate, so the condition evaluates to true, and Order Management applies the 6% rate to sales order total.

Why would I want the Order Entry Specialist to set the tax rate? The Order Entry Specialist works directly with your customer and might have more accurate details. For example, assume in the year 2019 you set up tax according the site where your Computer Service and Rentals resides in Arizona at 5%. However, the Order Entry Specialist finds out while talking with Computer Service and Rentals when placing a sales order that the tax authority in Arizona increased the tax rate to 6% in 2020. The Order Entry Specialist can use the Tax Classification Code to adjust the tax.

Summary of the Setup

1. Create the factor set.
2. Create the tax rate.
3. Create the condition set.

4. Create the tax rule.
5. Collect data.
6. Test your set up.

Assume you need to allow the Order Entry Specialist to set the tax rate at 6%.

Assume you already set up the US_TAX_REGIME tax regime, the STATE1 tax, and the STANDARD tax status code when you set up tax for the customer site. For details, see *Apply Tax According to Customer Site*.

Create the Factor Set

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Determining Factor Sets
2. On the Manage Tax Determining Factor Sets page, click **Create**.
3. On the Create Tax Determining Factor Set page, set the values.

Attribute	Value
Tax Determining Factor Set Code	6% in the US_TAX_REGIME
Tax Determining Factor Set Name	6% in the US_TAX_REGIME
Set Usage	Tax Rules
Tax Regime Code	US_TAX_REGIME

4. In the Associate Tax Determining Factors area, set the values, then click **Save and Close > Done**.

Attribute	Value
Determining Factor Class	Transaction Input Factor You can select from a wide variety of classes, such as geography, party, inventory, transaction, and so on. Transaction Input Factor specifies that you will use the input from a transaction to set the tax rate. A sales order is a type of transaction.
Determining Factor Name	Tax Classification Code Here you're specifying the name of the attribute in the sales order that the Order Entry Specialist will use to adjust the tax rate.

Create the Tax Rate

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rates and Tax Recovery Rates
2. On the Manage Tax Rates and Tax Recovery Rates page, click **Actions > Create**.
3. On the Create Tax Rate page, set values.

Attribute	Value
Tax Regime Code	US_TAX_REGIME
Configuration Owner	Global Configuration Owner
Tax	STATE1
Tax Status Code	STANDARD
Tax Jurisdiction Code	-
Tax Rate Code	6% Tax Rate
Tax Rate Type	Percentage
Order to Cash Procure to Pay Expenses	Contains a check mark.

4. In the Rate Periods area, set the values, then click **Save and Close > Done**.

Attribute	Value
Rate Percentage	6
Set as Default Rate	Doesn't contain a check mark.

Create the Condition Set

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Condition Sets
2. On the Manage Tax Condition Sets page, click **Create**.
3. On the Create Tax Condition Sets page, set the values.

Attribute	Value
Tax Condition Set Code	6% US_TAX_REGIME
Tax Condition Set Name	6% US_TAX_REGIME
Tax Determining Factor Set Code	6% IN THE US_TAX_REGIME

4. In the Tax Condition Set Details area, set the values, then click **Save and Close > Done**.

Attribute	Value
Tax Determining Factor Class	Transaction Input Factor
Tax Determining Factor Name	Tax Classification Code
Operator	Equal To
Value or Start From Range	6% Tax Rate

Create the Tax Rule

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rules
2. On the Manage Tax Rules page, click **Actions > Create**.
3. On the Create Determine Tax Applicability Rule page, set the values.

Attribute	Value
Configuration Owner	Global Configuration Owner
Tax Regime Code	US_TAX_REGIME
Tax	STATE1
Rule Code	Override Tax Classification
Rule Name	Override Tax Classification

4. In the Tax Determining Factor Set area, in the Code attribute, set the value, then click **Next**.

Attribute	Value
Code	6% in the US_TAX_REGIME

5. On the Create Determine Tax Applicability Rule Override Tax Classification page, in the Tax Condition Set area, set the values, then click **Save and Next**.

Attribute	Value
Tax Condition Set Code	6% US_TAX_REGIME
New Condition Set Order	1
Result	Applicable
Enabled	Contains a check mark

6. On the Edit Determine Tax Applicability Rule page, in the Rule Status and Order area, add a check mark to the Enabled attribute, then click **Submit > Done**.

Collect Data

1. Go to the Plan Inputs work area.

   Don't use the Plan Inputs task that's available in the Setup and Maintenance work area. Use the Plan Inputs work area instead.

2. In the Plan Inputs work area, click **Tasks > Collect Planning Data**.
3. In the Collect Planning Data dialog, set your source system, then move reference entities to selected entities.
    - Customer
    - Geographies
    - Organizations
4. Click **Submit**.

    For details, see *Collect Planning Data for Order Management*.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Ship-to Address	2164 Broadway, Tempe, AZ, 85282

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.
4. Click the **total** at the top of the sales order, then verify that the tax rate is 5%, which is the STD-DEFAULT rate you set up according to the ship-to address.

Attribute	Value
Total List Price	2,500.00
Discount	0.00

Attribute	Value
Total Net Price	2,500.00
Shipping	0.00
Total Tax	125.00 125 is 5% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,625.00

5. Click the **Actions down arrow** on the order line, then click **Edit Tax Determinants**.
6. Set the Tax Classification Code to 6% Tax Rate.
7. On the order header, click **Actions > Reprice Order**.
8. Click the **total** at the top of the sales order, then verify that the tax rate is 6%.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Tax	150.00 150 is 6% of 2,500, so this rate is correct.
Total Credit	0.00
Pay Now	2,650.00

Related Topics

- Apply Tax According to Customer Site
- Overview of Collecting Promising Data for Order Management
- Tax Determining Factor Sets and Condition Sets

Make Sales Order Tax and Invoice Tax the Same

Use Inventory Management and a scheduled process to make the tax on the sales order the same as the tax on the invoice.

Summary of the Set Up

1. Create the sales order.
2. Manage the item in inventory.
3. Create the invoice.
4. Verify tax on the invoice.

In this example, assume you already set up tax so it defaults to 5%.

Create the Sales Order

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.

4. Click the **total** at the top of the sales order, then notice the tax.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Tax	125.00 125 is 5% of 2,500.
Total Credit	0.00
Pay Now	2,625.00

5. Click **Submit**, then notice the order number.

   For this example, assume the order number is 526715.

Manage the Item in Inventory

Create Pick Wave

1. Open a window in a different browser.

   You use two browsers in this example. One to manage the sales order as the Order Entry Specialist and another to manage the item in inventory as an administrator.
2. Make sure you have the privileges that you need to manage inventory in the Inventory Management work area.

   For details, see the Inventory Manager chapter in *Security Reference for Manufacturing and Supply Chain Materials Management*.
3. Go to the Inventory Management work area, and click **Tasks**.
4. Set **Show Tasks** to Shipments, then click **Create Pick Wave**.
5. On the Create Pick Wave page, set the value, then click **Release Now**.

Attribute	Value
Order	526715
Customer	Computer Service and Rentals

Attribute	Value

Confirm Pick Slip

1. Click **Tasks > Confirm Pick Slips**.
2. On the Confirm Pick Slips page, set **Saved Searches** to Pick Slip.
3. Search for the value.

Attribute	Value
Order	526715

4. Click **Tasks > Confirm Pick Slips**.
5. On the Confirm Pick Slips page, search for the values.

Attribute	Value
Order Type	Sales Order
Order	526715

6. In the search results, click the **link** in the Pick Slip column.
7. On the Confirm Pick Slip page, click **Show More**, then note the value for the shipment.

   For this example, assume the shipment is 2381778.
8. Set the values, then click **Confirm > Confirm and Close > Done**.

Attribute	Value
Ready to Confirm	Contains a check mark
Picked Quantity	1

Manage Shipment

1. On the Inventory Management page, click **Tasks > Manage Shipments**.
2. On the Manage Shipments page, search for the value.

Attribute	Value
Shipment	2381778

Attribute	Value

3. On the Edit Shipment page, set the value, then click **Ship Confirm**.

Attribute	Value
Shipping Method	Select any value.
Shipped Quantity	1 Set this value in the Lines area.

4. In the Lines area, verify that the Line Status attribute equals Interfaced.

Create the Invoice

Create the invoice for the sales order.

1. Make sure you have the privileges that you need to administer Order Management.
2. Go to the Scheduled Processes work area, then click **Schedule New Process**.
3. Run the Import AutoInvoice scheduled process.

Parameter	Value
Business Unit	Select the business unit that you set on the sales order, such as Vision Operations.
Transaction Source	Distributed Order Orchestration
From Sales Order Number	526715
To Sales Order Number	526715

For details, see *Update Intercompany Receivables Invoice Import Details*.

Verify Tax on the Invoice

1. Go back to the browser you used to create the sales order.
2. Search for your order, then click **Actions > Switch to Fulfillment View**.
3. On the Order page, click Fulfillment Lines, then click **Action > View Fulfillment Details**.
4. In the Fulfillment Details dialog that displays, click Billing, then notice the value in the Invoice attribute. For this example, assume the value is 32001.
5. Go to the Billing work area, click Query by Example, then query for the value.

Attribute	Value
Transaction Number	32001

6. Click the **32001** link in the Transaction Number column.
7. On the Edit Transaction page, in the General Information area, examine the value.

Attribute	Value
Tax	125 This is the value of the tax on the invoice. Notice that this attribute contains the same value that the sales order contains for the tax.

Modify Tax on Invoice

What if the tax on the invoice must be different than the tax on the sales order?

Continuing our example, let's say you need to change the value of the tax invoice from 125 to 130.

1. When you're on the Edit Transaction page, in the General Information area, click the **pencil** in the Tax attribute.
2. In the dialog that displays, click **Actions > Add Row**.
3. Select a tax rate or override the tax rate and manually enter your own tax amount.
   For example, select a tax rate, change the value in the Tax Amount attribute to 130, then click Save and Close. Receivables will change tax on the invoice from 125 to 130 but Order Management won't change the tax amount on the sales order.
4. On the Edit Transaction page, click **Save and Close**.

Don't Add Any Tax on Sales Orders

Set up your environment so Order Management doesn't add any tax on the sales order.

In some instances you might not want to include any tax on your sales order. For example, you sell to a nonprofit organization that doesn't pay sales tax, you sell to a wholesaler who doesn't pay sales tax because they charge tax to their retail customers, or you charge tax according to the delivery address but the tax authority at that location doesn't charge sales tax.

Summary of the Setup

1. Create the factor set.
2. Create the condition set.
3. Create the tax rule.
4. Test your set up.

Assume you already set up the US_TAX_REGIME tax regime, the STATE1 tax, and the STANDARD tax status code when you set up tax for the customer site. For details, see *Apply Tax According to Customer Site*.

Create the Factor Set

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Determining Factor Sets
2. On the Manage Tax Determining Factor Sets page, click **Create**.
3. On the Create Tax Determining Factor Set page, set the values.

Attribute	Value
Tax Determining Factor Set Code	FACTOR SET FOR NO TAX
Tax Determining Factor Set Name	FACTOR SET FOR NO TAX
Set Usage	Tax Rules
Tax Regime Code	US_TAX_REGIME

4. In the Associate Tax Determining Factors area, set the values, then click **Save and Close > Done**.

Attribute	Value
Determining Factor Class	Transaction Generic Classification
Determining Factor Name	Transaction Business Category

Create the Condition Set

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Condition Sets
2. On the Manage Tax Condition Sets page, click **Create**.

3. On the Create Tax Condition Sets page, set the values.

Attribute	Value
Tax Condition Set Code	CONDITION SET FOR NO TAX
Tax Condition Set Name	CONDITION SET FOR NO TAX
Tax Determining Factor Set Code	FACTOR SET FOR NO TAX

4. In the Tax Condition Set Details area, set the values, then click **Save and Close > Done**.

Attribute	Value
Tax Determining Factor Class	Transaction Generic Classification
Tax Determining Factor Name	Transaction Business Category
Operator	Equal To
Value or Start From Range	SALES_TRANSACTION

Create the Tax Rule

1. On the Setup page, go to the task.
    - Offering: Financials
    - Functional Area: Transaction Tax
    - Task: Manage Tax Rules
2. On the Manage Tax Rules page, click **Actions > Create**.
3. On the Create Determine Tax Applicability Rule page, set the values.

Attribute	Value
Configuration Owner	Global Configuration Owner
Tax Regime Code	US_TAX_REGIME
Tax	STATE1

Attribute	Value
Rule Code	Rule for No Tax
Rule Name	Rule for No Tax

4. In the Tax Determining Factor Set area, in the Code attribute, set the value, then click **Next**.

Attribute	Value
Code	FACTOR SET FOR NO TAX

5. On the Create Determine Tax Applicability Rule page, in the Tax Condition Set area, set the values, then click **Save and Next**.

Attribute	Value
Tax Condition Set Code	CONDITION SET FOR NO TAX
New Condition Set Order	1
Result	Not Applicable
Enabled	Contains a check mark

6. On the Edit Determine Tax Applicability Rule page, in the Rule Status and Order area, add a check mark to the Enabled attribute, then click **Submit > Done**.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Ship-to Address	2164 Broadway, Tempe, AZ, 85282

2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	1
Amount	2,500

3. Click **Actions > Reprice Order**.
4. Click the **total** at the top of the sales order, then verify that there's no tax.

   The tax line should contain a value of 0 or there should be no tax line at all.

Attribute	Value
Total List Price	2,500.00
Discount	0.00
Total Net Price	2,500.00
Shipping	0.00
Total Credit	0.00
Pay Now	2,500.00

Related Topics

- Apply Tax According to Customer Site

Set Up Tax Profile for Legal Entity

If you create a sales order but it doesn't calculate tax, you might need to set up the tax profile.

Assume your tax doesn't get calculated when you create a sales order with the Business Unit set to Vision Operations.

Try this.

1. Go to the Setup and Maintenance work area.
2. On the Setup page, click **Tasks > Search**.
3. Search for and open the Manage Legal Entity Tax Profiles task.
4. On the Manage Legal Entity Tax Profiles page, search for the value.

Attribute	Value
Legal Entity	Vision Operations

5. In the search results, click **Vision Operations**.
6. On the Legal Entity Tax Profile page, click **Edit**.
7. Click **Service Subscriptions**.

   If the Service Subscriptions tab contains a record that's similar to these values, then Vision Operations subscribes to the order-to-cash flow.

Attribute	Value
Party Name	Vision Operations
Party Type	First party legal entity
Business Process Flow	Procure-to-pay and order-to-cash

8. Click **Controls and Defaults**.
9. On the Controls and Defaults tab, in the Tax Transaction area, set the value, then click **Save and Close**.

If Vision Operations	Then Make Sure the Use Legal Entity Tax Subscription Option
Subscribes to the order-to-cash flow.	Doesn't contain a check mark.
Doesn't subscribe to the order-to-cash flow.	Contains a check mark.

Get Tax Details Through SQL

Use SQL to get tax details for a sales order.

For example, get tax details for sales order 657486.

```
select dfl.source_order_number, dfl.fulfill_line_id, zr.tax, zr.tax_rate_code, zr.tax_regime_code,
 zr.percentage_rate
from doo_fulfill_lines_all dfl,
doo_order_charges doo,
doo_order_charge_components docc,
doo_order_tax_details dotd,
zx_rates_b zr
where dfl.fulfill_line_id=doo.parent_entity_id
and doo.order_charge_id=docc.order_charge_id
and docc.order_charge_component_id=dotd.order_charge_component_id
and dotd.tax_rate_id=zr.tax_rate_id
and dfl.source_order_number=657486
```

Related Topics

- Use SQL to Query Order Management Data

Troubleshoot Tax Set Up

Troubleshoot problems that happen with your tax setup for Order Management.

Trouble	Shoot
My sales order doesn't include any tax.	A variety of incorrect setups might cause this problem. Here are some set ups you can examine. • Make sure you set up the tax profile. For details, see *Set Up Tax Profile for Legal Entity*. • Make sure you set up a tax to apply, by default. Most implementations apply tax according to customer site. For details, see *Apply Tax According to Customer Site*.
I need to change the tax rate on my sales order starting on a specific date. For example, my local taxing authority changes their tax rate from 7% to 7.5% on October 9, 2020. I use the Manage Configuration Owner Tax Options page to add a new Tax Classification Code with a rate of 7.5%, however, I have a sales order that I already submitted with the 7% rate.	If your sales order. • Hasn't passed the Awaiting Shipping status, then you can revise it, change the Tax Classification Code to 7.5%, then resubmit it. • Already passed the Awaiting Shipping status, then you must return the order and create a new one with the 7.5% rate. I have no idea here. Is this correct? The tax amount on a sales order is only an estimate, and Order Management doesn't send it to Receivables for invoicing. The tax that Receivables uses depends on the tax effective dates, how you set up your tax classification code, and other attributes that affect tax.

Tax Determinants

Troubleshoot problems with tax determinants. To set them, you create a sales order in the Order Management work area, add an order line, click the Actions down arrow on the order line, then click Edit Tax Determinants.

Trouble	Shoot
The Product Type attribute in the Edit Tax Determinants dialog has a value that doesn't match the item. For example, the item is the AS54888 Desktop Computer,	1. Go to the Product Information Management work area, click **Tasks > Manage Items**, search for, then open your item for editing.

Trouble	Shoot
which is a Goods, but the Product Type attribute contains Services.	2. Click **Specifications > Inventory**, then set the Inventory Item attribute to Yes. If the item is a service, such as 3 Year Warranty, but the Product Type attribute in the Order Management work area displays Goods, then set the Inventory Item attribute to No.
I set the Tax Classification Code to a different value and click OK, but the exclusive tax on the sales order doesn't change.	Assume you set the Business Unit to Vision Operations on the sales order. Try this. 1. Enable the feature. o Go to the Setup and Maintenance work area, then select Financials next to Setup. o Click **Change Feature Opt In**. o On the Opt In Financials page, in the Financials row, click the **pencil** in the Features column. o On the Edit Features page, on the Tax Calculation on Receipt Accounting Distributions row, add a check mark to the Enable option, then click **Done > Done**. 2. On the Setup Financials page, go to the task. o Offering: Financials o Functional Area: Transaction Tax o Task: Manage Configuration Owner Tax Options. 3. On the Configuration Owner Transaction Tax Options page, search the Configuration Owner attribute for Vision Operations. 4. In the search results, in the row that has Application Name equal to Tax and Event Class equal to Purchase Transaction, set the Regime Determination Set to Standard Tax Classification Codes. For details, see *Configure Tax Calculation and Accounting*.
I see that the Tax Classification Code attribute contains a value, then submit the order. Sometime later, I create a return for the order, but now the Tax Classification Code is empty.	Order Management uses different lookups for the Tax Classification Code. It uses. • ZX_OUTPUT_CLASSIFICATIONS for sales orders • ZX_INPUT_CLASSIFICATIONS for return orders Try this. 1. Go to the Setup and Maintenance work area, then go to the task. o Offering: Financials o Functional Area: Transaction Tax o Task: Manage Tax Lookup Codes 2. On the Manage Tax Lookup Codes page, search the Lookup Type attribute for ZX_OUTPUT_CLASSIFICATIONS. 3. In the Financials Generic Lookup Type area, examine the lookup codes. 4. Search the Lookup Type attribute for ZX_INPUT_CLASSIFICATIONS for return orders. 5. In the Financials Generic Lookup Type area, make sure the lookup codes contain the same values that you examined for ZX_OUTPUT_CLASSIFICATIONS. If they don't, add them.
I need to calculate tax for an order line that has a value of zero in the Amount attribute on the line, or I need to calculate tax according to some value other than what I have in the Amount attribute. For example, the Amount attribute contains $35, but I need to assess tax on a value of $30. How do I do this?	Try this. 1. Enable the opt-in feature Specify Business Unit for Selling Profit Center for Goods and Services Tax. You have to enable this feature to get the Edit Tax Determinants dialog to display the Assessable Value attribute. For details, see *Set Up Business Units for Selling Profit Centers*.

Trouble	Shoot
	2. Create your order line, open the Edit Tax Determinants dialog, enter 30 in the Assessable Value attribute, then click Submit. Order Management will send a value of 30 as the Assessable Value to Oracle Receivables, and Receivables will calculate tax according to the assessable value 30 instead of the value 35 from the Amount attribute.

Related Topics

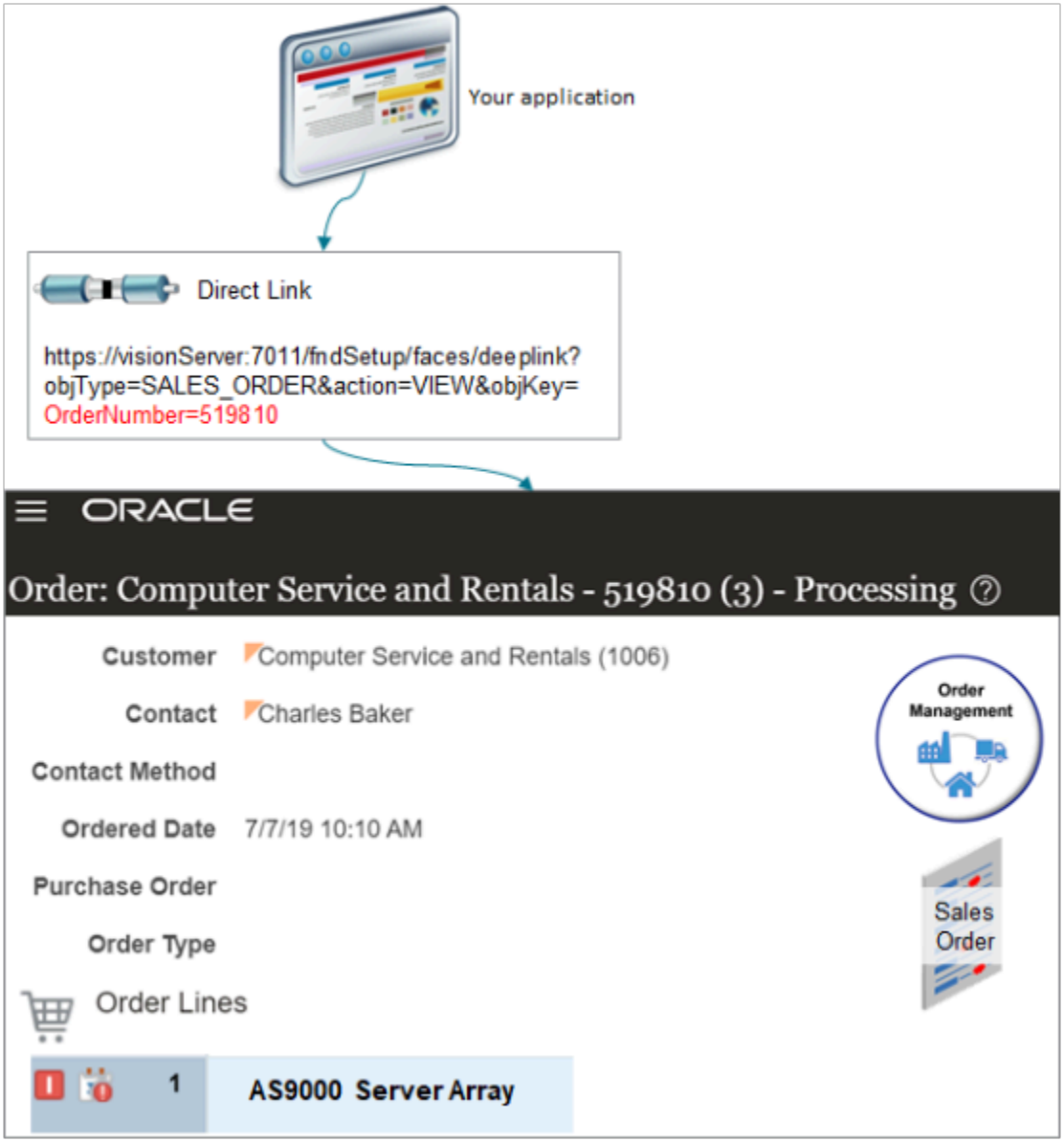
- Set Up Tax Profile for Legal Entity
- Apply Tax According to Customer Site
- Set Up Business Units for Selling Profit Centers
- Configure Tax Calculation and Accounting

More

Create Direct Links to Order Management Pages

Create a direct link from your application or an Oracle Application to a page in the Order Management work area.

For example, assume you have a legacy application named VisionSCM that your field support team uses during maintenance calls. Your service team is on site at your Computer Service and Rentals customer to service the AS9000 Server Array, and needs to get quickly from VisionSCM to the sales order in the Order Management work area so they can view order details while they're servicing the equipment. The service team also needs to send an email to a parts supplier where having a direct link in the email would expedite getting the correct part.



Here's the generic format for you direct link.

```
https://host:port/fndSetup/faces/deeplink?objType=SALES_ORDER&action=actionVIEW&objKey=objectKey
```

where

Code	Description
`host`	The name of the server that hosts your implementation of Order Management. For example, assume the Vision Operations organization uses a server named visionServer.
`port`	The port on your server that you use to host your implementation of Order Management. Assume Vision Operations uses port 7011 on visionServer.

Code	Description
`action`	The action to take when doing the direct link. Use one of these values. • VIEW • FULFILLMENT_VIEW • CREATE
`VIEW&objKey`	A combination that identifies the type of view you're linking to and a value that uniquely identifies that object. Here are the values you can use for VIEW. • HeaderId • OrderNumber • SourceTransactionNumber • SourceTransactionId • SourceTransactionSystem • ChangeVersionNumber • DraftOrderFlag The value for `objKey` uniquely identifies the object you're linking to.

Link to the Overview Page

Set action to NONE or don't include it in your link. Set objType to SALES_ORDER. For example:

```
https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER
```

Link to Sales Order

Set objType to SALES_ORDER.

Set action to VIEW.

Link According To	Example
HeaderId	Link to the sales order that has a HeaderId of 300100181471495. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=HeaderId=300100181471495` HeaderId is different from the sales order number, such as 519810.
OrderNumber	Link to the sales order that has an OrderNumber of 519810. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=OrderNumber=519810` If OrderNumber isn't unique, then use SourceOrderSystem to identify the source order.

Link According To	Example
SourceTransactionNumber	Link to the sales order that you imported for source transaction 507252 from the GPR source system. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=SourceTransactionNumber=507252;SourceTransactionSystem=GPR`
SourceTransactionId	Link to the sales order that you imported for source transaction Id 300100177124844 from the GPR source system. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=SourceTransactionId=300100177124844;SourceTransactionSystem=`
ChangeVersionNumber	Link to version 2 of the sales order that you imported for source transaction Id 300100177124844 from the GPR source system. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=SourceTransactionId=300100177124844;SourceTransactionSystem=`
DraftOrderFlag	Link to the draft version of sales order 504121. `http:// visionServer:7011/fscmUI/faces/deeplink?objType=SALES_ORDER&action=VIEW&objKey=OrderNumber=504121;DraftOrderFlag=true`

For HeaderId, OrderNumber, SourceTransactionNumber, and SourceTransactionId.

- If you already submitted the sales order to fulfillment, then the link displays the sales order in the same view that you see when you click **Tasks > Manage Orders** in the Order Management work area.

  `Order: Computer Service and Rentals - 519810 - Processing`

- If the sales order is in Draft status, then the link displays the draft order.

Link to Sales Order in Fulfillment View

Set objType to SALES_ORDER.

Set action to FULFILLMENT_VIEW.

Link According To	Example
HeaderId	Link to the sales order that has a HeaderId of 300100181471495. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=FULFILLMENT_VIEW&objKey=HeaderId=300100181471495`
OrderNumber	Link to the sales order that has an OrderNumber of 519810. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=FULFILLMENT_VIEW&objKey=OrderNumber=519810` If OrderNumber isn't unique, then use SourceOrderSystem to identify the source order.
SourceTransactionNumber	Link to the sales order that you imported for source transaction 507252 from the GPR source system.

Link According To	Example
	`https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=FULFILLMENT_VIEW&objKey=SourceTransactionNumber=507252;SourceTransactionSystem=GPR`
SourceTransactionId	Link to the sales order that you imported for source transaction Id 300100177124844 from the GPR source system. `https://visionServer:7011/fndSetup/faces/deeplink?objType=SALES_ORDER&action=FULFILLMENT_VIEW&objKey=SourceTransactionId=300100177124844;SourceTransactionSystem=GPR`

The link displays the sales order in the same view that you see when you access the sales order through **Tasks > Manage Fulfillment Lines** in the Order Management work area.

Link to Create Order Page

Set objType to SALES_ORDER.

Set action to CREATE.

For example:

```
https://visionServer:80/fndSetup/faces/deeplink?objType=SALES_ORDER&action=CREATE
```

The link displays the same Create Order page that you see when you click **Create Order** in the Order Management work area.

Return to Calling Application

How do I get back to where I once belonged?

The direct link feature displays a return icon above the order total. You click it to return to the application that called the direct link.

Assume you call a direct link from VisionSCM. The visionServer server hosts VisionSCM on port 7011.

VisionSCM must include the `returnApp` parameter and `returnAppParams` parameter in the `deeplink` URL so the return icon knows how to get back to VisionSCM. Here's your format.

```
https://server:port/fndSetup/faces/deeplink?
objType=SALES_ORDER&action=VIEW&objKey=OrderNumber=519810&returnApp=applicationName&returnAppParams=parameterName1=V
% parameterName2=Value%parameterName3=Value
```

where

Code	Description
returnApp	Identifies the name of the application that called the direct link.
returnAppParams	List of parameters that you can include in the call. You must use a % (percent symbol) to separate each parameter and its value.

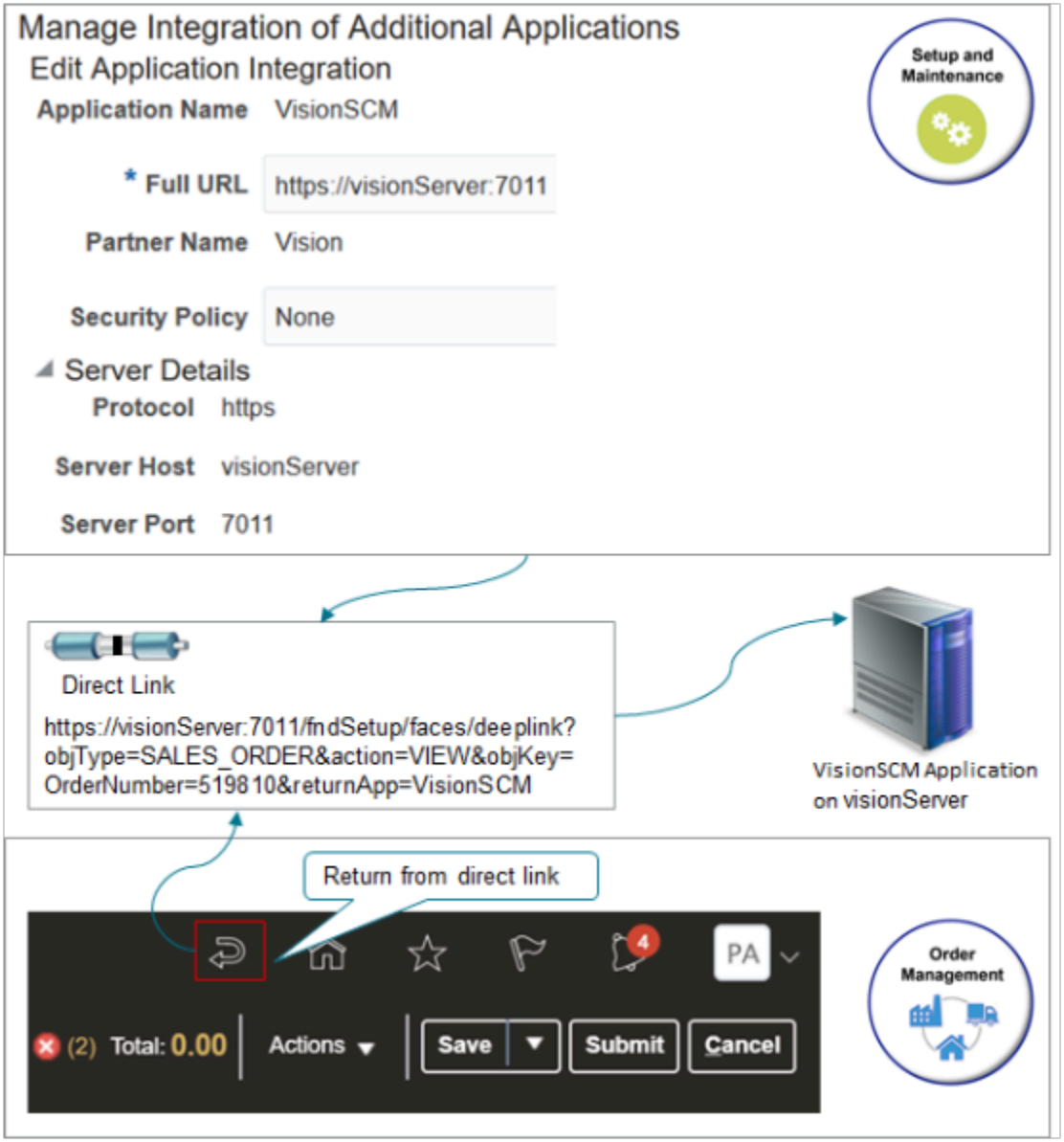
Here's the URL you use to return to VisionSCM.

```
https://visionServer:7011/fndSetup/faces/deeplink?
objType=SALES_ORDER&action=VIEW&objKey=OrderNumber=519810&returnApp=VisionSCM&returnAppParams=abc=123%26xyz=456
```

where

Code	Description
returnApp	Identifies the calling application as VisionSCM
returnAppParams	Sets the value of parameter abc to 123 and the value of parameter 26xyz to 456.

You must add the calling application in the Setup and Maintenance work area. For example:



Try it.

1. Go to the Setup and Maintenance work area.
2. Click **Tasks > Manage Setup Content**.
3. On the Manage Setup Content page, click **Manage Integration of Additional Applications**.
4. On the Manage Integration of Additional Applications page, click **Actions > Create**.
5. On the Create Application Integration page, set the values, click **Apply**, then click **Save and Close**.

Attribute	Value
Application Name	VisionSCM
Full URL	https://visionServer:7011

Attribute	Value

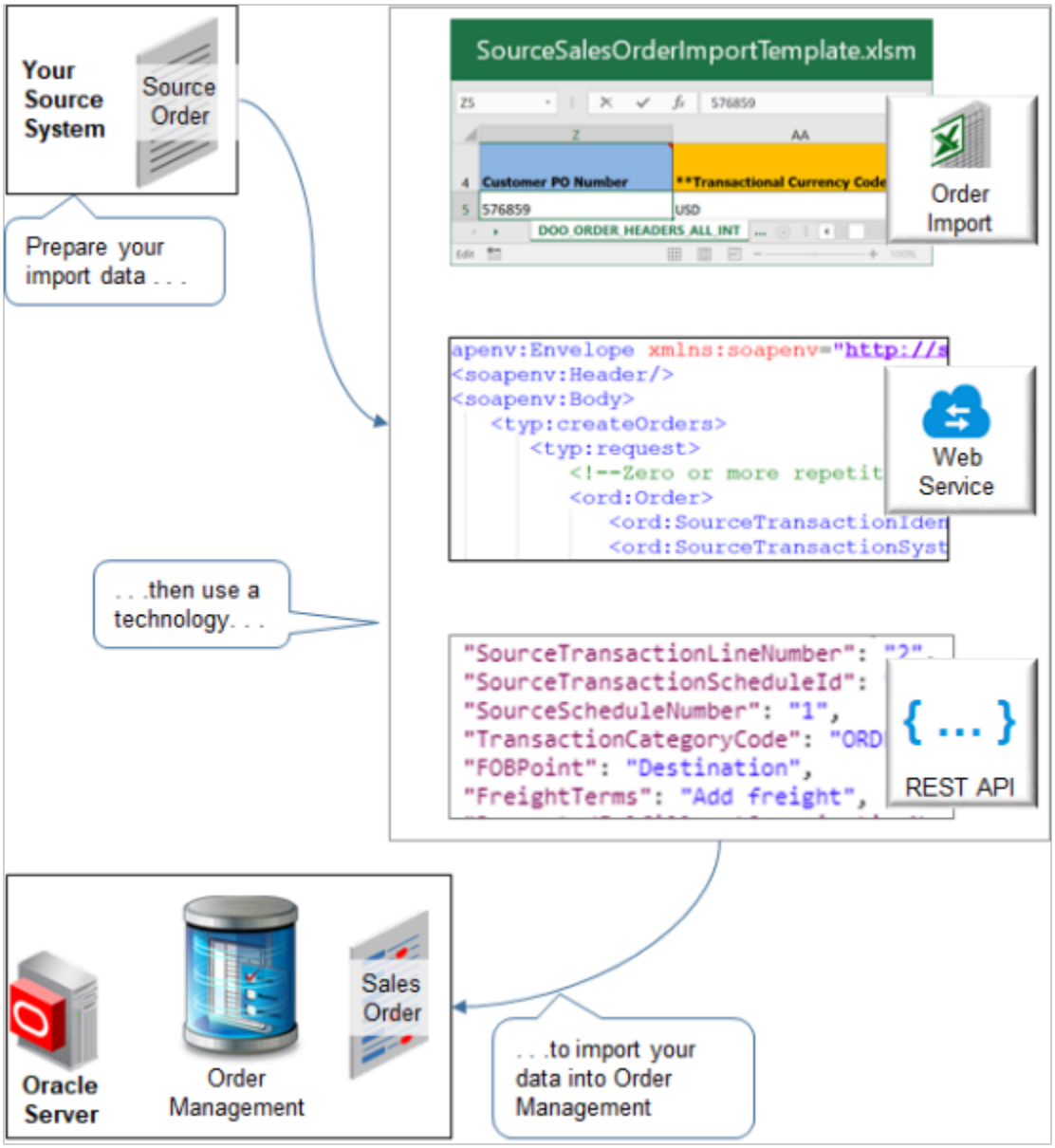
5 Import and Transform

Import

Overview

Overview of Importing Orders Into Order Management

Use the Order Import Template, a web service, or REST API to import orders from your source system into Oracle Order Management.



Use the Order Import Template to reduce errors and simplify your import. This template contains a structure that the Oracle database requires. It includes a tab for each database table, and it displays tabs in a specific sequence. Columns on each tab represent the table columns that Oracle requires, and the template specifies the data type that Oracle requires for each database column.

What You Need	Where You Can Get It
Learn how to use the Order Import Template.	*Convert Source Data Into CSV File*
Copy of the Order Import Template and details about tables that the template references.	Go to *File-Based Data Import (FBDI) for SCM*. In the Order Management chapter, click **Import Sales Orders**.

What You Need	Where You Can Get It
Example templates.	Go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Payloads and Files attachment.
Details about how to use a web service to create an integration that imports source orders.	*Guidelines for Using Web Services to Integrate Order Management*.
Details about the salesOrdersForOrderHub REST API.	For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.
More technical details.	*Master Note Importing Data using FBDI (Doc ID 2665940.1)*

Compare the Import Technologies

What You're Doing	Order Management Work Area	File Based Data Import	REST API	Web Service	ADF Desktop Integration (ADFDI)
Update an extensible flexfield on the order header while the sales order and the order lines are closed.	No	No	No	No	No
Update the Request Date attribute on the order line.	Yes	Yes	Yes	Yes	No
Update the Order Type attribute on the order header.	Yes	Yes	Yes	Yes	No
Update the header of an order that isn't Canceled or Closed.	Yes	No	Yes	No	No
Update the Primary Salesperson on the order header.	Yes	Yes, but only if you haven't submitted the order.	Yes	No	No
Update pricing on an order line.*	Yes	Yes	Yes	Yes	No
Update a sales order that's in Awaiting Billing status.	No	No	No	No	No

* You can update pricing only if pricing isn't frozen. For details see, *Freeze Price on Sales Orders*.

Import a Large Number of Source Orders

We recommend that you use file-based data import (FBDI) instead of web services or REST API to import a large number of source orders. FBDI is more resilient and is easier to retry imports that fail without having to resend the source order from your source system.

Consider an example.

- You process about 4,000 orders each day.
- You average about 50 order lines in each source order.
- You process about 200,000 order lines each day.
- You expect to import about 330 source orders each hour from 8AM to 8 PM.
- You currently need to import from four source systems, and you plan to expand to eight, possibly doubling overall processing to 8,000 source orders a day.
- You must ship your sales orders the day after you import them.
- Your orchestration process includes typical fulfillment tasks, such as schedule, reserve, ship, bill, and so on.

FBDI is the best technology to handle this kind of volume.

If you must use a web service or REST API to import your source orders, then include only one source order in each payload request.

REST API only supports the synchronous operation, so if you include more than one source order in each payload, and if the request runs for more than about 5 minutes, then you might encounter a time out error that you can't recover from.

If you must use the Order Import web service, then use the asynchronous operation. Don't use the synchronous operation.

Differences in Attribute Names and Values

The import technologies use slightly different names and values for some attributes. Order Management processes the attributes in the same way regardless of these nomenclature differences.

Attribute Names

Format	Order Import Template	Web Services and REST API
Capitalization of the attribute name.	Headline, such as Payment Method Code.	Camel case, such as PaymentMethodCode.
Name of an attribute that contains a Boolean value.	Doesn't give an indication, for example Freeze Pricing.	Typically includes the text `Flag`, such as FreezePriceFlag.
Name of an attribute that contains an identifier.	Typically includes the word Identifier, such as Source Transaction Identifier.	Typically includes the text Id, such as SourceTransactionId.

Boolean Values

Technology	True	False
Order Import Template	Y	N
Web Services	Y or true	N or false
REST API	true	false

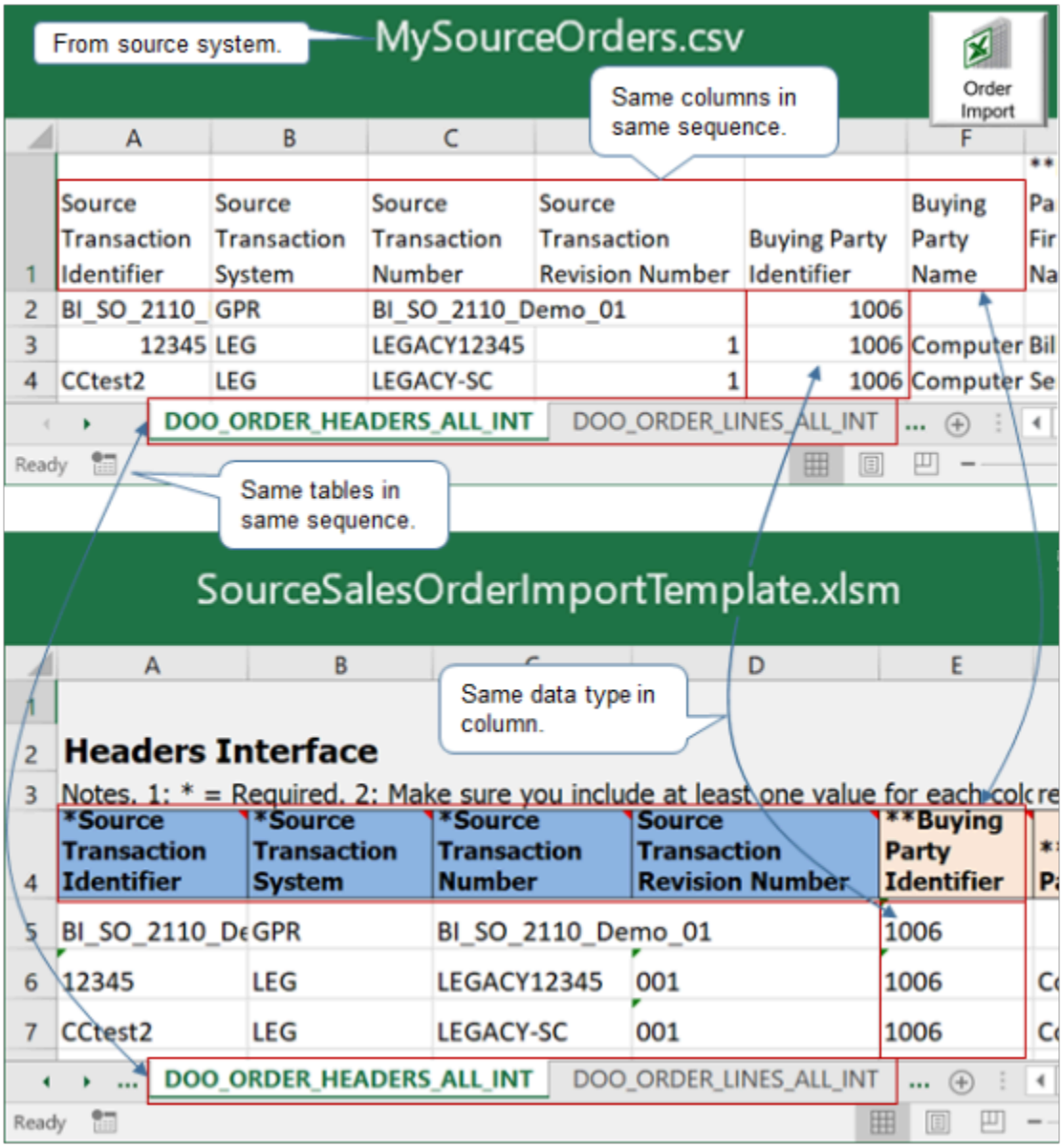
Related Topics

- Import Orders Into Order Management
- Guidelines for Using Web Services to Integrate Order Management
- Import Different Kinds of Data
- Convert Source Data Into CSV File
- Data Security

Guidelines for Using File-Based Data Import

Convert Source Data Into CSV File

You must create a CSV file that includes order data from your source system. Use these guidelines to make sure the structure that the CSV file contains mirrors the structure that the Order Import Template contains.



Note

- Create a CSV file, such as MySourceOrders.csv. To make sure you include the same structure that the template uses, we recommend that you open the template file, save it as a CSV file, structure your data so it mirrors the CSV file, and then copy your source data into the CSV file
- Include the same table names and sequence them in the same sequential order. The tab sequence that the template uses determines sequential order.
- Include the same columns in each table, and arrange columns in the same sequential order inside each table.
- Use the same data type for each column.
- To get the template file, see *Overview of Importing Orders Into Order Management*.

Example

View an example of the CSV structure.

1. Open the Order Import Template in Microsoft Excel.
2. Click the **DOO_ORDER_HEADERS_ALL_INT** tab.
3. Click **File > Save As > Excel Workbook**.
4. Save the file as a csv file, then use a text editor to examine the output.

```
BI_SO_2110_Demo_01,GPR,BI_SO_2110_Demo_01,,1006,,,,,,,,1560,,,,,,,,,,,576859,USD,,,,,10/10/2016
 17:12,204,,,,,,,204,,,,,,,,,,
12345,LEG,LEGACY12345,1,1006,Computer Service and Rentals,Bill,Gates, ,Jr,Mr,,1006-
Party,1560, ,Bill,Williams,Bob, ,Mr.,ORGANIZATION,,1560-Contact,1000,Preferred Sold To Contact Point
 Orig Sys Reference,,USD,US Dollar,Spot,1.23,1/15/2014 10:10,1/29/2016 10:10, ,Vision Operations,,,,Order
 12345,12345, ,,,N,NONE,No Reason,1/15/2014 10:10,N,N,N,,N
CCtest2,LEG,LEGACY-SC,1,1006,Computer Service and Rentals,,,,,,1006,1006-Party,1560,Piere
 Legrand,,,,,,,CONTACT-1560,1560-Contact,,,,USD, ,,,,1/29/2016 10:10, ,Vision Operations,,,,order with
 header sales credits,cc,204,,,N,,,,N,N,N,,
```

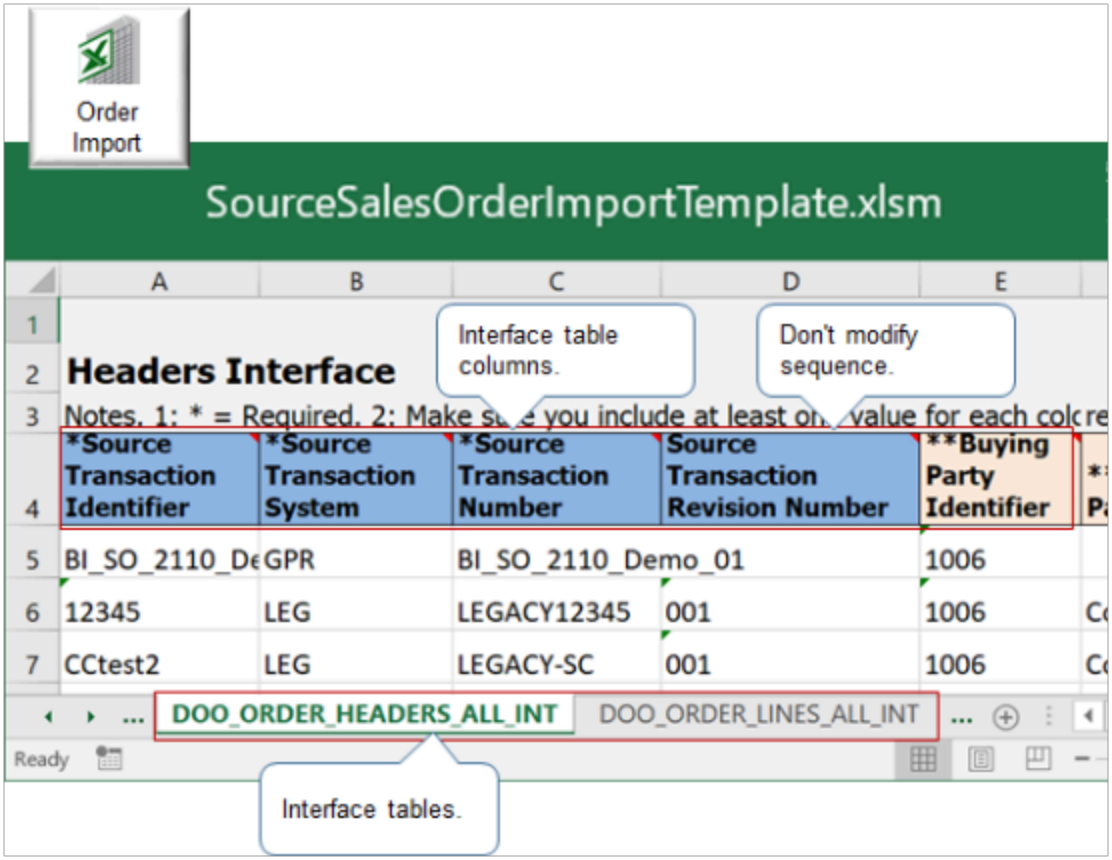
Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- How Order Management Transforms Source Orders Into Sales Orders
- Data Security

Use the Import Sales Order Template

Apply guidelines when you use the import template.

Reflect the Structure of the Oracle Database



To help visualize the structure, go to *File-Based Data Import (FBDI) for Oracle SCM*, look in the Order Management chapter, click **Import Sales Orders**, then click **SourceSalesOrderImportTemplate.xlsm**. Examine the file that you download.

Enter your data so it reflects the structure of the Oracle database. The Source Sales Order Import Template uses a separate spreadsheet in an Excel workbook to represent each interface table.

- A row near the top of each spreadsheet contains column headers.
- Each column header represents the name of an interface table column.
- The spreadsheet displays columns in a specific sequence.
- You must not modify the sequence. If you do, import will fail.
- Hide columns you don't need, but you must not delete them. If you delete a column, import will fail.

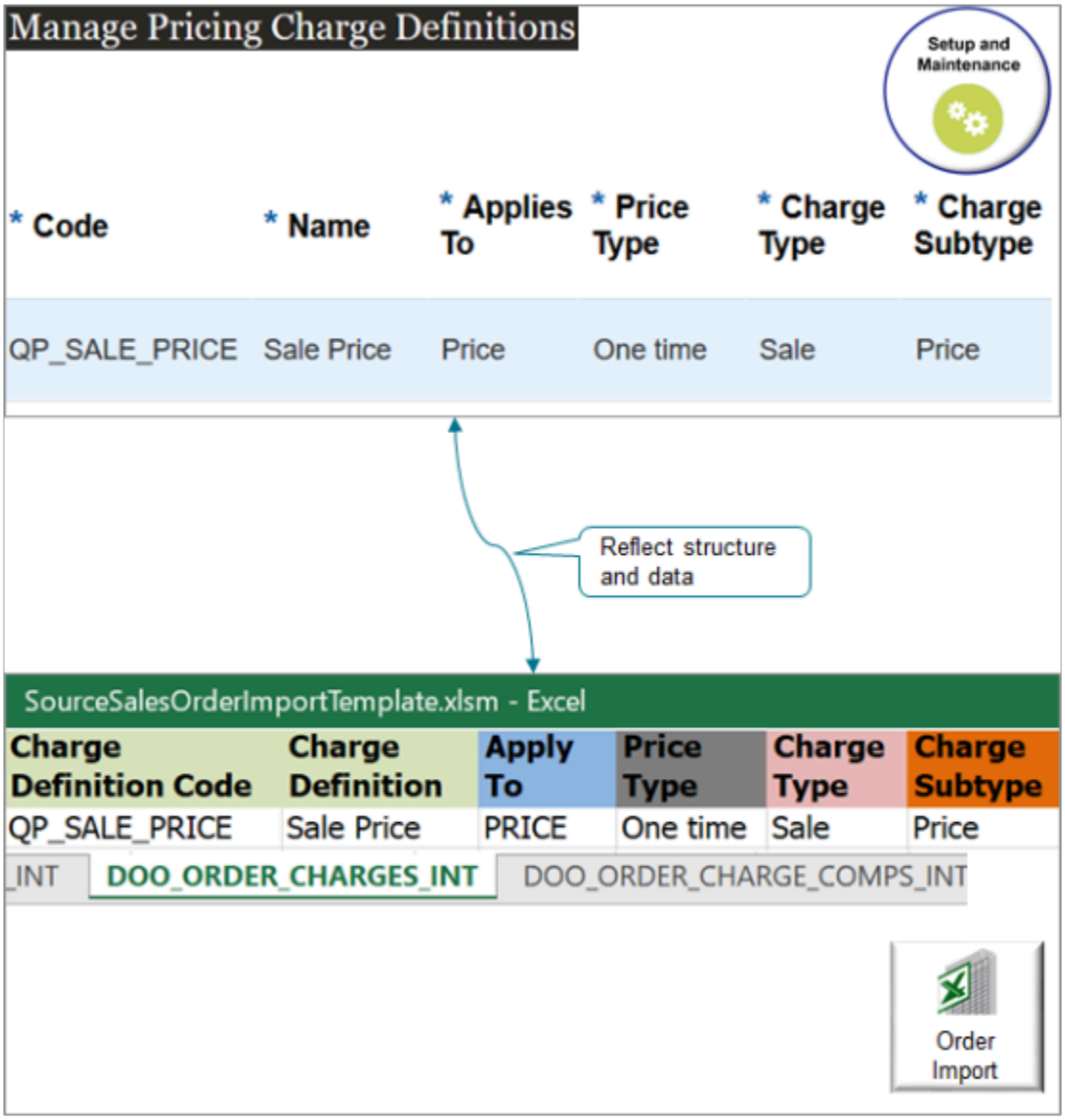
Source Sales Order Import Template includes a tab for each of these interface tables. The _INT suffix indicates its an interface table. You must include data in each required table.

Table Name	Details This Table Must Contain	Required
DOO_ORDER_HEADERS_ALL_INT	Order header data.	Yes
DOO_ORDER_LINES_ALL_INT	Order line data.	Yes

Table Name	Details This Table Must Contain	Required
DOO_ORDER_ADDRESSES_INT	Address for the sales order.	Yes
DOO_ORDER_TXN_ATTRIBUTES_INT	Attributes that might be associated with an order line.	Yes
DOO_ORDER_SALES_CREDITS_INT	Sales credits for the sales representative that the source order references.	No
DOO_ORDER_PAYMENTS_INT	Payment details for the order line.	No
DOO_ORDER_LOT_SERIALS_INT	Lot serial details for the order line.	No
DOO_ORDER_DOC_REFERENCES_INT	References to documents that Order Management imports with the source order.	No
DOO_ORDER_CHARGES_INT	Charges for the sales order.	Required only if you don't calculate pricing in Oracle Pricing.
DOO_ORDER_CHARGE_COMPS_INT	Charge components for the sales order.	Required only if you don't calculate pricing in Oracle Pricing.
DOO_ORDER_BILLING_PLANS_INT	Billing plans for the sales order.	No
DOO_ORDER_MANUAL_PRICE_ADJ_INT	Manual price adjustments for the sales order.	No
DOO_ORDER_HDRS_ALL_EFF_B_INT	Flexfield details for the order header.	No
DOO_ORDER_LINES_ALL_EFF_B_INT	Flexfield details for the order line.	No
DOO_PROJECTS_INT	Projects for the sales order.	No

Example

Here's an example where the columns and values on the DOO_ORDER_CHARGES_INT worksheet reflect data from the Manage Pricing Charge Definitions page in the Setup and Maintenance work area.

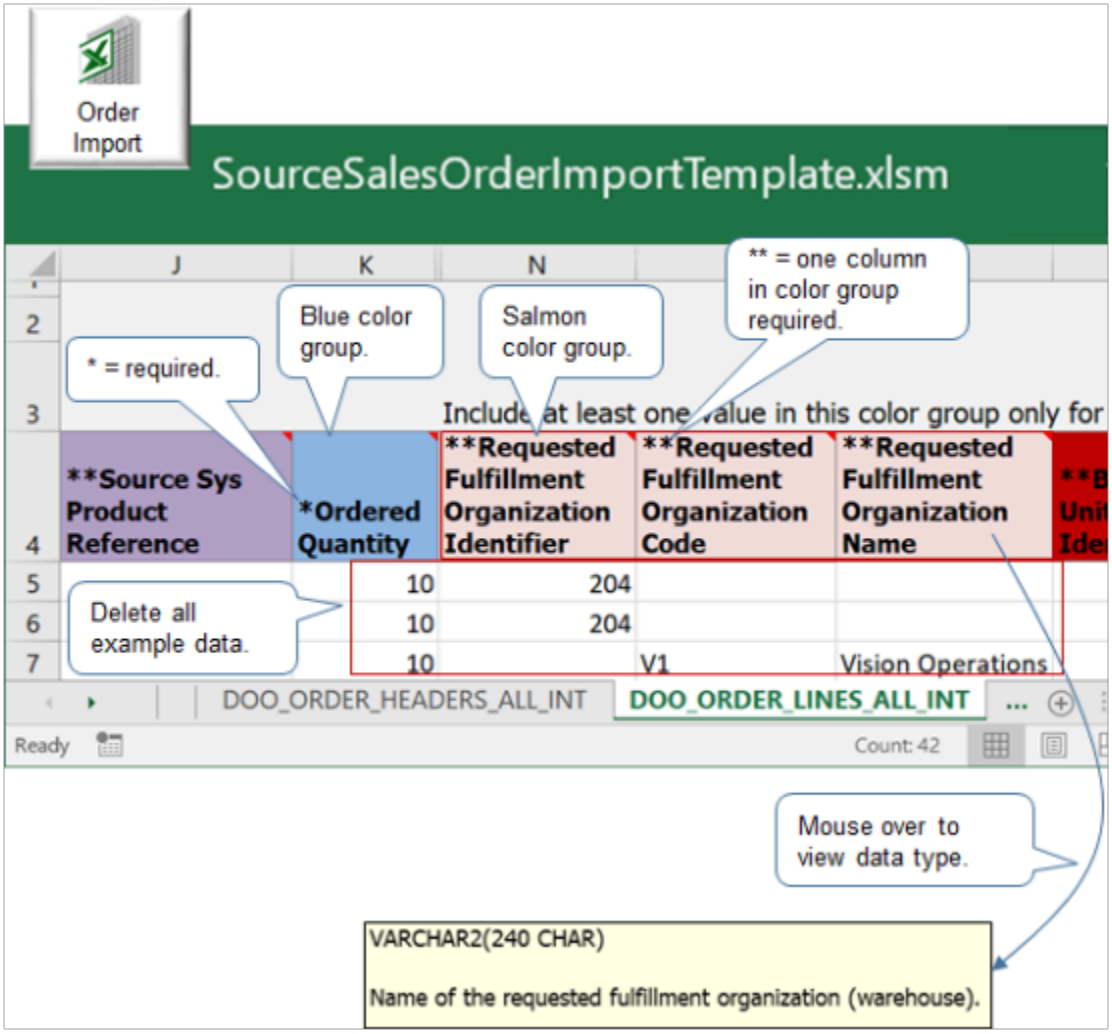


For example:

Table Name	Details This Table Must Contain	Attribute Value
Code	Charge Definition Code	QP_SALE_PRICE
Name	Charge Definition	Sale Price
Applies To	Apply To	PRICE
Price Type	Price Type	One Time
Charge Type	Charge Type	Sale

Table Name	Details This Table Must Contain	Attribute Value
Charge Subtype	Charge Subtype	Price

Enter Data in Required Columns



Note

- Source Sales Order Import Template uses an asterisk (*) to indicate required columns. For example, the Source Transaction Schedule Identifier column on the DOO_ORDER_LINES_ALL_INT tab is required.
- The template uses an asterisk (*) to indicate each required column.
- The template uses a double asterisk (**) to indicate the color group requires a value in at least one of these columns. For example, if a color group includes columns **a, **b, and c, then you must include a value in a or b, or a and b.
- Required columns aren't always contiguous. Carefully examine column headings in each spreadsheet to make sure you include data for each required column.

- To locate required headings on each spreadsheet, press **CTRL + F**, enter ~* (a tilde and an asterisk), and then click **Find All**.

Use Color to Determine What's Required

- The template uses a blue background for column headings, by default. It uses other colors to group some columns.
- For example, the DOO_ORDER_HEADERS_ALL_INT sheet uses a tan color to group columns Buying Party Identifier, Buying Party Name, and Buying Party Number. You must enter a value in at least one column of this color group.
- Some sheets include more than one color group. You must enter at least one value for each color group.
- Color groups might not display contiguously. Carefully examine the color group on each spreadsheet to make sure you include at least one value for each color group.
- Some color groups are conditional. For example, you must enter a value for at least one column in the color group that represents these columns.
    - Requested Fulfillment Organization Identifier
    - Requested Fulfillment Organization Code
    - Requested Fulfillment Organization Name

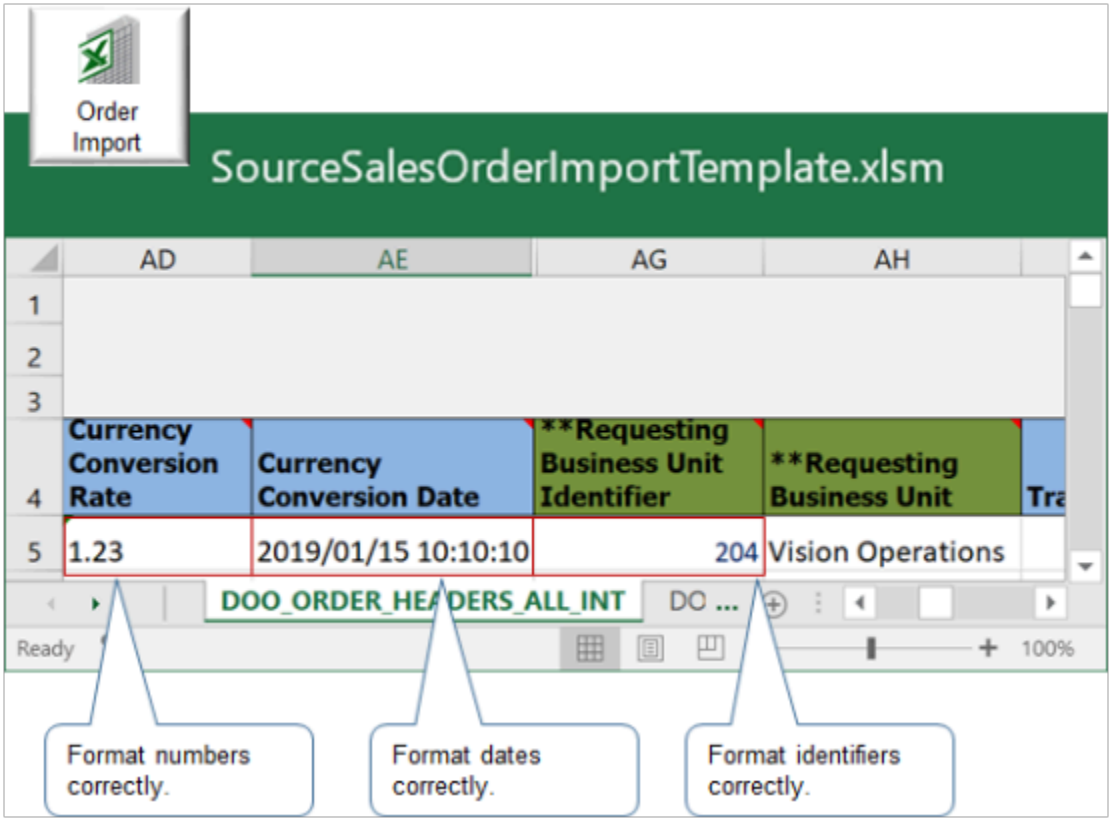
  The DOO_ORDER_LINES_ALL_INT sheet contains these columns and uses them only for rows that references a return order. The sheet includes an instructional note immediately above each conditional color group.

Use the Correct Data Type

- Use the format that the database column uses. For example, data in column Source Transaction Identifier of tab DOO_ORDER_LINES_ALL_INT must use format VARCHAR2(50).
- Use example data and descriptive text to help determine the type of data to include.
- Source Sales Order Import Template comes predefined with example data already populated, and some column headers include descriptive text.
- For example, click the DOO_ORDER_LINES_ALL_INT tab, and notice that row seven in column Requested Fulfillment Organization Name includes a value of Vision Operations.
- Position your mouse over column header Requested Fulfillment Organization Name, and then read the descriptive text that indicates this field must contain only VARCHAR2(240 CHAR) data. The value Vision Operations meets this requirement.
- Make sure you remove example data before you import data.

For details, see Oracle Datatypes in *Oracle Server Concepts Manual*.

Format Amounts and Dates



Note

- You must use a comma (,) for the thousands separator and a period (.) for the decimal separator in number columns. For example, use 1,500.25. Don't use 1500,25.
- Use a whole number when required. If a column must use a whole number, then the control file that validates data in this column includes only whole numbers. If it doesn't, then the import fails.
- You must use format `YYYY/MM/DD HH:MM:SS` for each date field. For example, `1/15/2019 10:10:10 AM` is January 15, year 2019, 10 hours, 10 minutes, and 10 seconds after 00:00:00 AM.

Format Internal Identifiers

If your data includes an internal identifier, then make sure you map it correctly.

- An internal identifier is an identifier that Order Management uses to create a reference between a lookup that you specify for an item, customer, or reference data, to details from the Product Information Manager work area, Trading Community Architecture, or collection data. This data resides in the Order Orchestration and Planning Repository. For example, if your order uses Payment Terms, then you must set up this value in Oracle, then set up collections processing.
- You must map your source data value to an Order Management value for each internal identifier column.
- Read the descriptive text in the spreadsheet for help with columns that require an internal identifier.
- The template includes more than one column that specifies details for internal identifiers. It uses the Identifier suffix to identify these columns, such as Source Transaction Identifier or Buying Party Identifier.
- If your source data includes an identifier column that doesn't require setup in Product Information Management, Trading Community Architecture, or data collection, then you can use the implementation pages

in your implementation project to get the identifier and other attributes that you must map for the internal identifier.

Import Customer Items

Make sure you correctly set up the rank for each customer item relationship before you import orders. For details, see *Import Customer Items Into Order Management*.

Other Things to Consider

If the value that you import for any attribute starts with a space character and you want to keep the space, then you must enclose the value with double quotation marks (" ").

Assume your attribute contains the text MyValue, and the value has three space characters before MyValue. Here's how you import that:

```
"  MyValue"
```

where

- There are three space characters between the opening quotation mark and MyValue.

If you don't do this, your import will still work but the imported value won't have any leading spaces.

Sequence Your Sales Orders

If you import more than one sales order, then Order Management creates them in a sequence according to the Source Transaction Identifier attribute and the Source Transaction System attribute.

Assume you have these values on the DOO_ORDER_HEADERS_ALL_INT worksheet.

Source Transaction Identifier	Source Transaction System
00001	LEG
00002	LEG
00003	GPR
00006	GPR
00005	GPR
00004	LEG

Here's the sequence that Order Management will use when it creates sales orders for these transactions.

Source Transaction Identifier	Sales Order
00001	50003
00002	50004
00003	50005
00006	50008

Source Transaction Identifier	Sales Order
00005	50007
00004	50006

where

- 50002 is the most recent sales order that exists in Oracle data when you start the import, so the import starts with the next number that's available, which is 50003.
- Order Management creates sales orders in a sequence according to Source Transaction Identifier regardless of the value in Source Transaction System.

The sales order numbers in this example assume that no other transactions occur during the import. Assume you create sales order 50004 in the Order Management work area while you're importing, and you create 50004 right after the import creates order 50003 for source transaction 00001 but before the import creates the sales order for transaction 00002. In this case, the import would use the next available order number for transaction 00002, which is order number 50005.

You can't modify this sequence.

Cancel Order Lines

Follow these guidelines when you use file-based data import to cancel an order line.

- Use the same template populated with the same data that you used when you imported the source order.
- Modify values only in the DOO_ORDER_LINES_ALL_INT worksheet. Don't edit data in any of the other worksheets.
- Set these values in the DOO_ORDER_LINES_ALL_INT worksheet on the order line that you need to cancel.

Attribute	Value
Ordered Quantity	0 If you set this value to 0, then the import will set the order line status to Cancelled.
Request Cancel Date	Specify the date when you want Order Management to cancel the line.
Cancel Reason Code	Enter a code.
Cancel Reason	Enter a reason.
Operation Mode	Cancel

Here's how to determine what values you can use for the reason code and the reason:

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups

2. On the Manage Order Lookups page, search for the value.

Attribute	Value
Lookup Type	DOO_RETURN_REASON

3. In the Lookup Codes section, examine the values in the:
    - Lookup Code attribute. You can use any one of these values in the Cancel Reason Code attribute in your worksheet.
    - Meaning attribute. You can use any one of these values in the Cancel Reason attribute in your worksheet.
    - If you don't see a value that meets your needs, then add one.

If you:

- Created the original order in Order Management, then set the Source Transaction Identifier to the SOURCE_ORDER_ID.
- Didn't create the original order in Order Management, then set the Source Transaction Identifier to SOURCE_ORDER_NUMBER.

Make sure you specify the same value for the:

- Source Transaction Identifier attribute in DOO_ORDER_LINES_ALL_INT and in DOO_ORDER_HEADERS_ALL_INT
- Source Transaction System attribute in DOO_ORDER_LINES_ALL_INT and in DOO_ORDER_HEADERS_ALL_INT

You can cancel more than one line at the same time. Here's what that might look like.

Product Number	Ordered Quantity	Request Cancel Date	Cancel Reason Code	Cancel Reason	Operation Mode
AS54888	0	2023/04/01	LATE	Late Delivery	Cancel
CN82441	0	2023/04/01	NO_REASON	No Reason	Cancel
WR001	0	2023/04/01	DEFECTIVE	Item is Defective	Cancel
CM62202	0	2023/04/01	MISSING_PARTS	Parts are Missing	Cancel
CM76962	0	2023/04/01	DISCONTINUED	Discontinued Item	Cancel
CM74237	0	2023/04/01	PRICE_CHANGE	Price Change	Cancel
CN62101	0	2023/04/01	QTY_CHANGE	Quantity Change	Cancel

You can cancel as many lines as you need to cancel. There's no limit on the number of lines that you can cancel.

For more, see *Import Return Orders* and *Cancel Backorder During Import*.

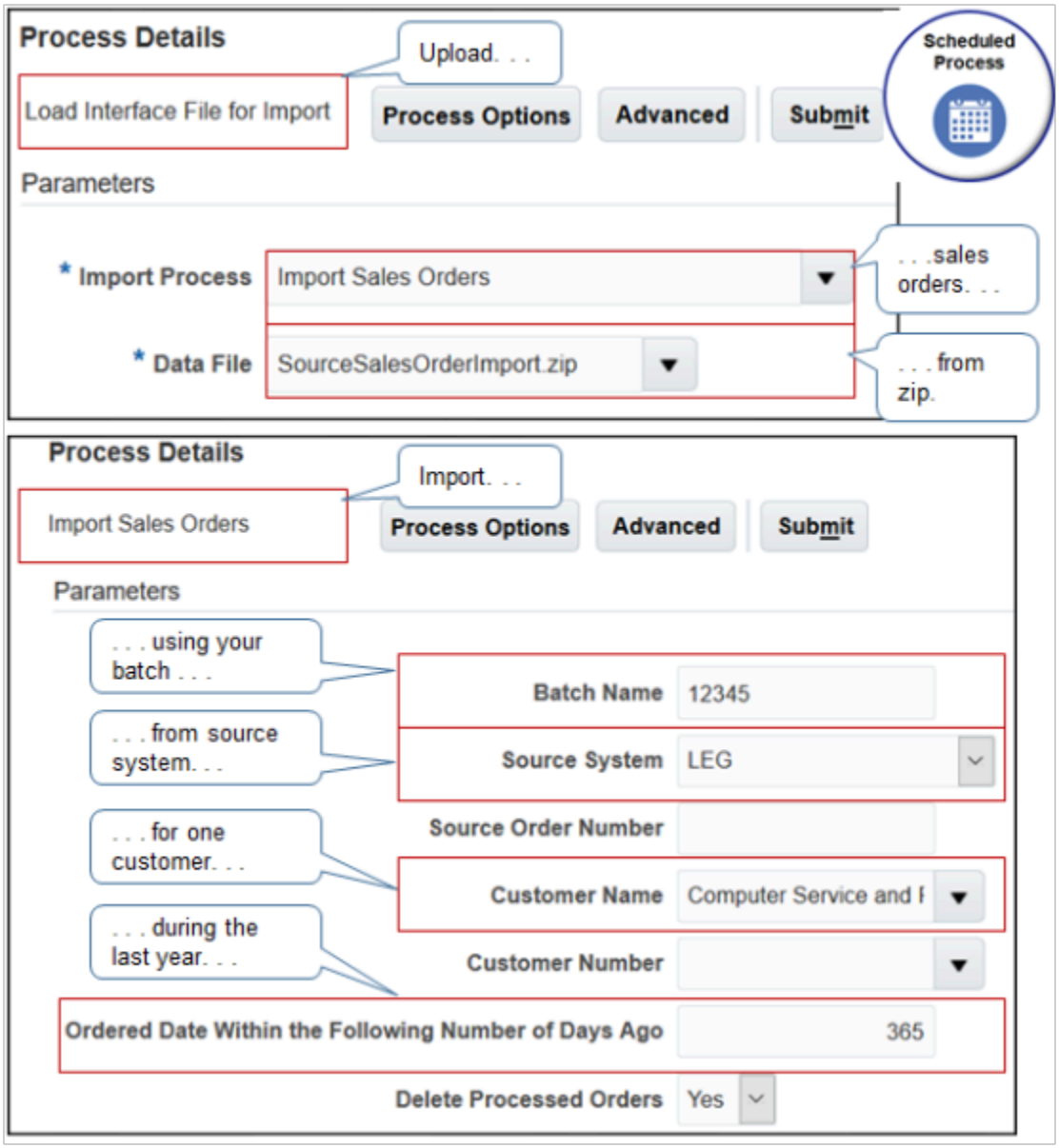
Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- How Order Management Transforms Source Orders Into Sales Orders
- Data Security

Upload and Import Your Data

Apply guidelines when you upload and import your data.

Use scheduled processes.



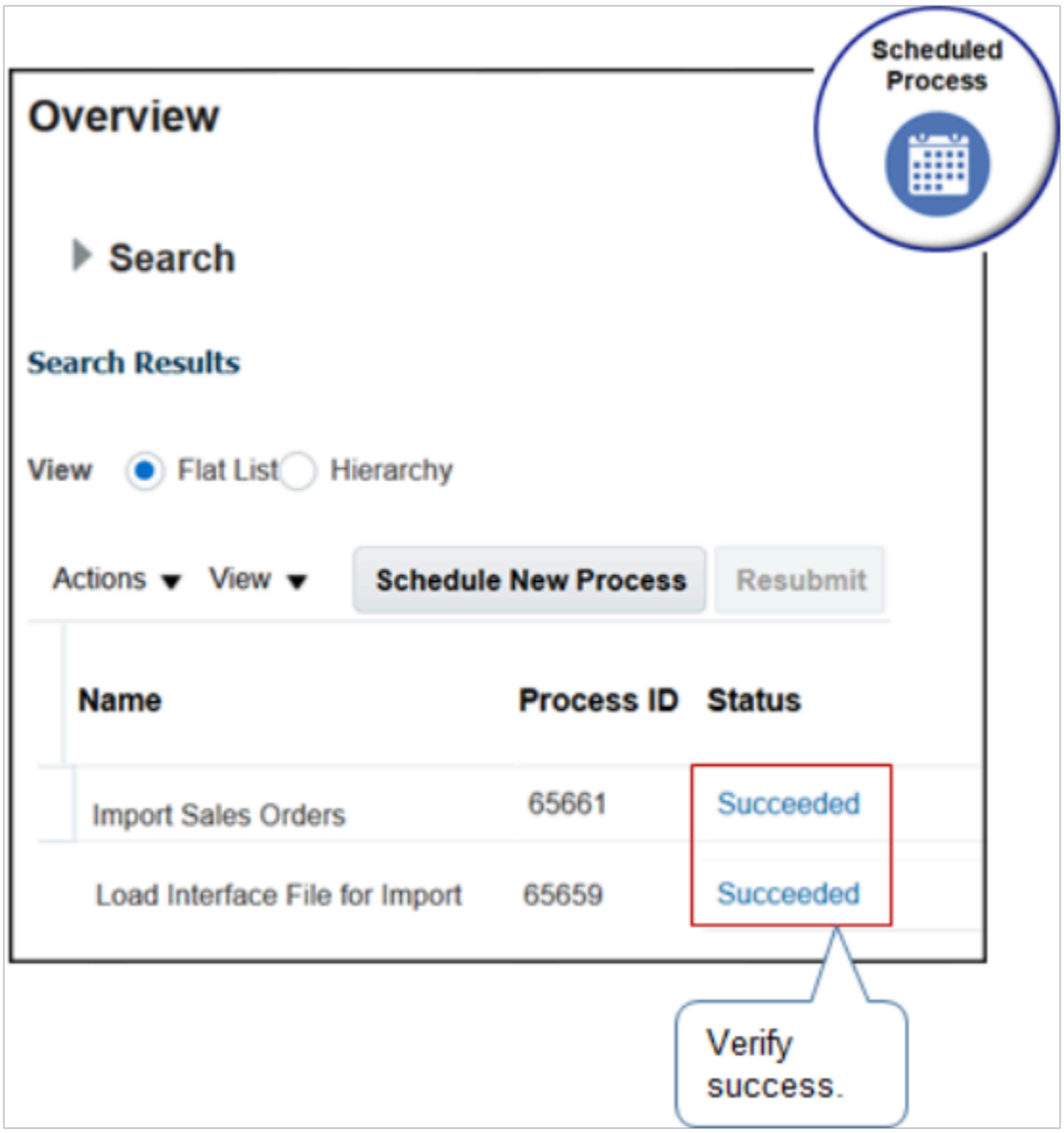
Note

- Use the *Load Interface File for Import* scheduled process to upload your data to the server.

  Use the *Import Sales Orders* scheduled process to import data into Order Management.

  - Specify the zip file that you created from the template.
  - Specify parameters as necessary. For example, filter according to source system, customer, and time frame.

Verify that your scheduled processes succeeded.



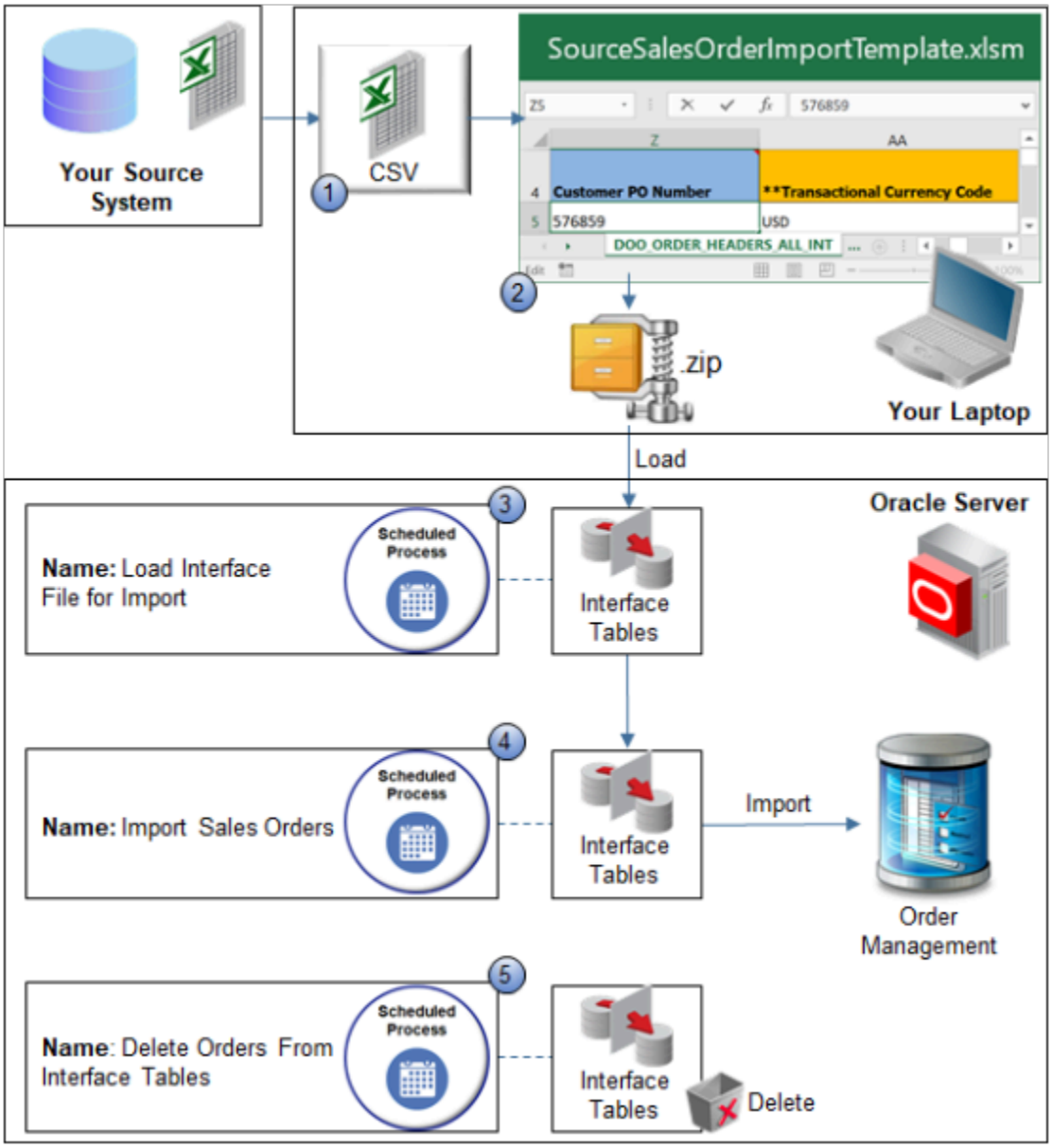
Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- How Order Management Transforms Source Orders Into Sales Orders
- Data Security

Procedures

Import Orders Into Order Management

Use files to import source orders from a source system into Order Management.



Note

1. Convert source data.
2. Create import file.
3. Upload source data.
4. Import source data.
5. Delete imported orders from interface tables.

Order import typically uses this same flow when you integrate Order Management with a source system, except you import orders from a spreadsheet. For details, see *How Order-to-Cash Works with Order Capture Systems*.

Convert Source Data

Use the template to help make sure your converted data uses the same structure that the Oracle database requires. This topic describes one way to convert your source data. The details of your conversion might require a different way.

If you can't use Order Import Template for some reason, and if you aren't familiar with doing data conversion, then consult Oracle resources about how to use an open interface when importing data. Learn about other ways to import your data, such as through a web service. For details, see *Web Services That You Can Use to Integrate Order Management*.

Convert your source data into a CSV file.

1. Download the Order Import Template.
    - Go to *File-Based Data Import for Oracle Supply Chain Management Cloud*.
    - Expand **Order Management**, then click **Import Sales Orders**.
    - In the Import Sales Orders area, click **SourceSalesOrderImportTemplate.xlsm**.
2. Use a spreadsheet editor that can read a CSV file, such as Microsoft Excel, to open Order Import Template, and then familiarize yourself with the structure that it uses.
3. Use a data manipulation tool to structure your source data so it mirrors the structure that the Order Import Template contains, and then save this data to a CSV file.

    You can use SQL (Structured Query Language), ODI (Oracle Data Integrator), or some other tool to convert your source data into a CSV file.

Create Import File

1. Prepare the Order Import Template.
    - Use a spreadsheet editor that can read a CSV file, such as Excel, to open Order Import Template.
    - Delete the example data from Order Import Template. This template comes with example data that helps you determine the type of data that you must include. For example, row four of tab DOO_ORDER_HEADERS_ALL_INT includes example data. Make sure you delete all example data from all tables in the spreadsheet, even from tables you don't need.
2. Copy and paste your source data into each of the tables in Order Import Template.
    - Use a spreadsheet editor to open the CSV file that contains your source data.
    - Copy your order line data to the clipboard.
    - In the Order Import Template, click the **DOO_ORDER_LINES_ALL_INT** tab.
    - Click cell **A5**, then paste your data.
    - Examine the results to make sure you correctly pasted the data. For example, make sure the Product Description column contains VARCHAR data, and that the Ordered Quantity column contains NUMBER data.
    - Continue copying data for each table until you finish copying all your source data into Order Import Template.

    Save your work after each copy.
3. Create the import file.
    - Click the **Create CSV** tab, then click **Generate CSV File**.

      If Generate CSV File isn't active, then click **Developer** in the menu bar, then click **Macros**. In the Macro dialog, select **GenCSV**, then click **Run**.
    - Wait for the macro in Excel to finish.

      When the macro finishes, Excel displays a dialog that allows you to save a zip file.

- In the save dialog, select a location to save your zip file, then click **Save**.

  The macro creates a zip file that includes a separate file for each table that the template contains.
- Optional. In subsequent save dialog boxes, save each individual CSV file.

  If you must import data for an individual table, then save the data as an individual CSV file. Otherwise, click **Cancel**.

Upload Source Data

1. Make sure you have the privileges that you need to administer Order Management.
2. Set up the Number of Processes for Order Import parameter.

   For details, see *Manage Order Management Parameters*.
3. Go to the Scheduled Processes work area.
4. On the Scheduled Process page, click **Schedule New Process**.
5. In the Schedule New Process dialog, set the value, then click **OK**.

Attribute	Value
Name	See *Load Interface File for Import*.

6. In the Process Details dialog, set the parameters.

Parameter	Description
Import Process	Select Import Sales Orders.
Data File	a. In the Data File attribute, click the **down arrow**. b. Scroll down and click **Upload a New File**. c. In the Upload File dialog, click **Select File.** d. In your Windows Explorer window, locate and select the zip file that you created when you used the Order Import Template, then click **Open**. e. In the Upload File dialog, click **OK.** f. In the Process Details dialog, make sure the Data File attribute displays the name of the zip file before you continue.

7. Click **Submit**.
8. In the Confirmation dialog, note the value of attribute Process ID, then click **OK > Close**.
9. Click **Actions > Refresh**.
10. Use the Process ID that you noted earlier to locate your scheduled process, then make sure the Status attribute for this process displays Succeeded.

    The Succeeded status indicates that the scheduled process successfully uploaded your source data. If the upload fails on any row, then the status displays Error. If the search results doesn't display your process, then click **Refresh** until it does.
11. Correct errors, if necessary.

    - If the scheduled process ends in an error, then click the **Error** status in the search results for your scheduled process and examine the log and output files to get details about data that caused the error.
    - Use Excel to open the Order Import Template that contains your source data, and then correct the source data that causes the error.

- In the Order Import Template, click **Generate CSV File**, then run the scheduled process again.

Repeat this step until the scheduled process successfully uploads your source data.

Import Source Data

1. Run the *Import Sales Orders* scheduled process. Set the parameters in the Process Details dialog.

Parameter	Description
Batch Name	As an option, enter a value that you can use to identify the batch of sales orders that you're importing.
Delete Processed Orders	Set a value. - **Yes**. Delete each sales order that you successfully import from the interface tables. - **No**. Don't delete them.

Set other, optional parameters to filter the data that the scheduled process will look at. To avoid a conflict in the data to import, specify only one of these optional parameter:

Parameter	Filter the Import so it Only Imports Records:
Source System	From one source system. Enter the value that you set in the Source Transaction System column on the DOO_ORDER_LINES_ALL_INT tab of the Order Import Template. For example, enter `LEG`.
Source Order Number	For one source order. Enter the value that you set in the Source Transaction Identifier column on the DOO_ORDER_LINES_ALL_INT tab of the Order Import Template. For example, enter `12345`.
Customer Name	For one customer. Select the value that you set in the Buying Party Name column on the DOO_ORDER_HEADERS_ALL_INT tab of the Order Import Template. For example, select Computer Service and Rentals.
Customer Number	For one customer. Select the value that you set in the Buying Party Identifier column on the DOO_ORDER_HEADERS_ALL_INT tab of the Order Import Template.

Parameter	Filter the Import so it Only Imports Records:
	For example, select 1006.
Ordered Date Within the Following Number of Days Ago	For sales orders according to the Ordered Date attribute. For example, if today is January 15, and if you set this parameter to 5, then the import will only import sales orders that have an Ordered Date that happens between January 11 and January 15.
Order Interface Status	If Order Management fails to import a source order, then it sets the status for the order to an error status. Use this parameter to specify whether to process the order in a subsequent import. Set the value. o **All statuses**. Import all orders regardless of status. o **Error**. Import orders that are in error status. Ignore all other orders. o **No status**. Import orders that have no status. Ignore all other orders.

2. Make sure the Status is Succeeded attribute for the *Import Sales Orders* scheduled process.

   The Succeeded status indicates that the scheduled process successfully imported your source data. If the import fails on any row, then the status displays Error.
3. Verify your import.
   - Go to the Order Management work area, then click **Tasks > Manage Order Orchestration Messages**.
   - Query for records that the import includes.
   - Examine errors in the spreadsheet and fix source data in the Order Import Template that causes them.
   - Submit the modified data and make sure the scheduled process successfully imports all rows.
   - Navigate to work area Order Management and query for one or more of the orders you imported.
   - Make sure the work area displays the sales order, and that the order data is identical to the source data.

Troubleshoot your import.

- See *Troubleshoot Problems With Order Import*.
- Adjust the Number of Processes for Order Import order management parameter depending on how it impacts performance. For details, see *Manage Order Management Parameters*.
- If you must import binary data, then you must convert your binary data to text before you do the import, and you must use the base64 encoding scheme when you do the conversion. For details, see the I Can't Import Binary Data section in *Troubleshoot Problems With Order Import*.

Delete Imported Orders From Interface Tables

Order Management uses interface tables when you import data from your source system so it can handle import errors and to retain a data backup during import. As an option, run the *Delete Orders from Interface Tables* scheduled process to remove data and save storage space.

- Run this scheduled process after you successfully finish running the Import Sales Order scheduled process. You might want to wait a few days after you finish the import before you run Delete Orders from Interface Tables. This way, you can see what data you imported if you need to troubleshoot the import. For details, see .
- Order Management uses interface tables when you use File Based Date Import or REST API to import data from your source system. It uses these tables to help it manage any errors that happen during your import and to keep a backup copy of your data.
- This scheduled process deletes data from interface tables for the order header, the order line, and any entities that are part of the sales order, including charges, charge components, document references, extensible flexfields on the order header and the order line, lots, serials, manual price adjustments, payments, and sales credits.
- This scheduled process doesn't delete child entities that aren't part of a sales order or order line. For example, it doesn't delete a sales credit that isn't part of a sales order.

Use these parameters to filter the data that the scheduled process will look at.

Parameter	Description
Batch Name	Provide a unique name for each batch. If you have a lot of data, then we recommend that you run this scheduled process in batches. This keeps each instance relatively small, helps to reduce the time that each run needs to finish, and can help to focus troubleshooting efforts. Assume you have source system *x* and source system *y*. Run batch *a* for *x*, and then run batch *b* for *y*.
Source System	Specify the source system that provided the data that you imported.
Orders Uploaded Prior to the Following Number of Days Ago	Specify the number of days that the scheduled process should consider, starting with the current date. For example, if today is March 15, and if you set this parameter to 5, then the scheduled process will delete all orders that you imported before March 11. It won't delete any orders that you imported from March 15 to March 11.
Order Interface Status	Select a status. For example, if you select Error, then the scheduled process will only delete rows that are in error from the interface table. You can select: • All Statuses • Error • Imported • No Status • Warning In most cases, we recommend that you select All Statuses unless you have a specific business need to filter the data you want to delete.

All parameters are optional, but you have to specify at least one of them.

No special combinations are required.

Include the Batch Name

Make sure you include a value in the BatchName attribute in your import payload.

Make sure you specify the Batch Name parameter when you run the *Import Sales Orders* scheduled process.

If you don't specify a batch name, then the scheduled process will start a separate instance for each unique source system that you include in your import. For example, if your import includes 5 unique source systems, then you'll have 5 instances of the scheduled process running at the same time, and this might degrade performance.

This behavior might also mess up how you expect the Number of Processes for Order Import order management parameter will perform. Assume during testing, to achieve optimal performance, you conclude to set Number of Processes for Order Import to 10. If you don't specify a batch name, and if you import 2 unique source systems, then the scheduled process will run only two concurrent instances instead of 10. This means it will take much longer for those 2 instances to finish when compared to running 10 concurrent instances, even though you determined that your system can run 10 concurrent instances without affecting performance.

Related Topics

- Overview of Importing Orders Into Order Management
- Roadmap for Setting Up Order-to-Cash
- How Order-to-Cash Works with Order Capture Systems
- Guidelines for Using Scheduled Processes in Order Management

Example of Importing Orders into Order Management

Use this example to get experience using file-based data import (FBDI).

1. Download the SourceSalesOrderImportTemplate.xlsm file.
    - Go to *File-Based Data Import (FBDI) for Oracle SCM*.
    - In the Order Management chapter, click **Import Sales Orders**.
    - In the File Links area, click **SourceSalesOrderImportTemplate.xlsm**.

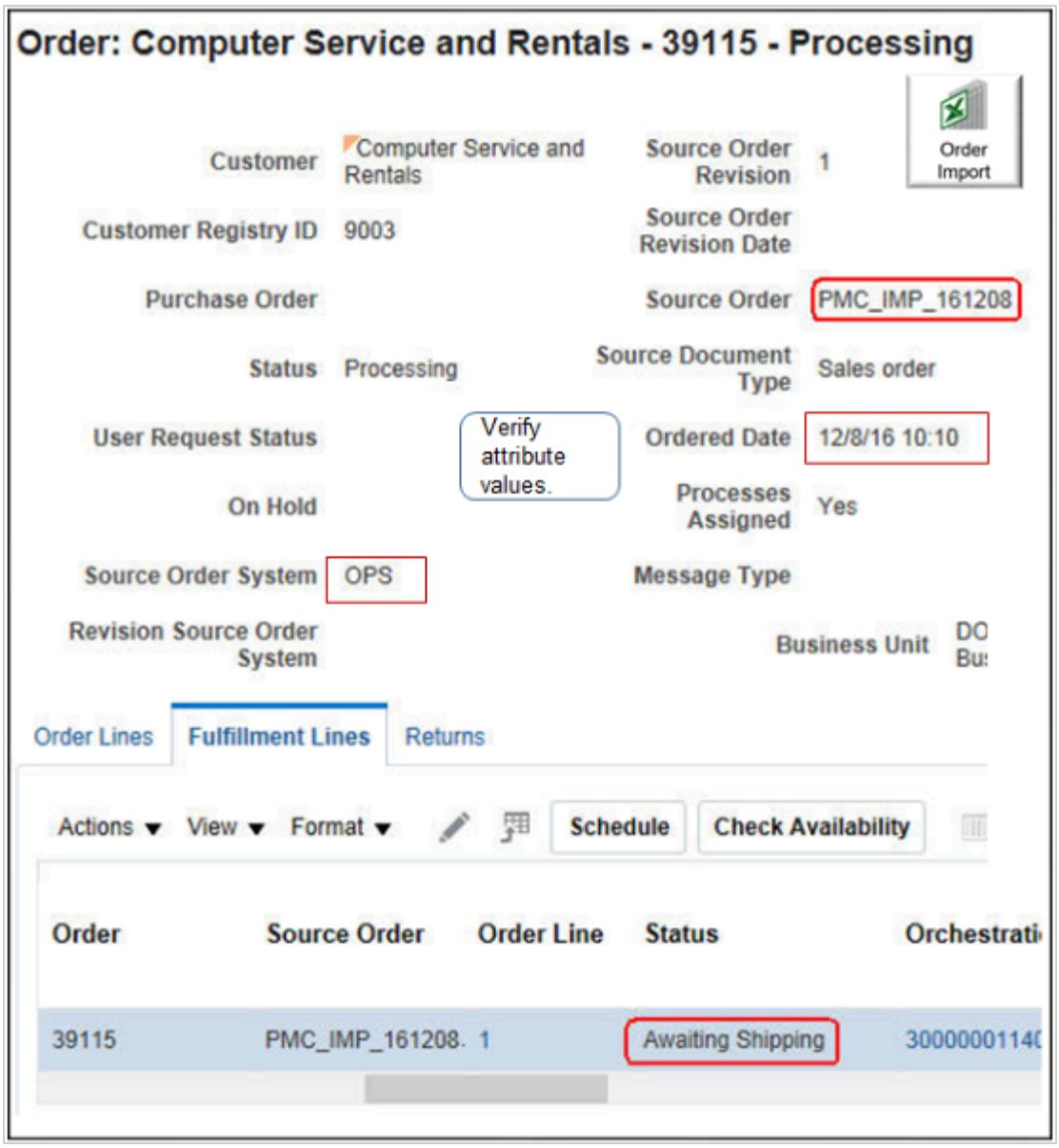
      The page downloads a copy of the file to your computer.
    - Use Excel to open the file.

    This file contains example data that you can use for this example.
2. In the file you downloaded, click **Generate CSV File**, then save the output file as SourceSalesOrderImport.zip.
3. Run the *Load Interface File for Import* scheduled process.
4. Run the *Imported Sales Orders* scheduled process.

5. Go to the Order Management work area, use a fulfillment view to open a sales order that the example file contains, then verify your imported data.

   Examine the values in the Source Order, Ordered Date, Source Order System, and Status attributes. Make sure they match the example data you imported.

   Here's an example.



   Your data might or might not match the data in this screen print, depending on the example data that's available in SourceSalesOrderImportTemplate.xlsm.

Example Sales Order

Here's an example of a sales order that you can import. To get more experience, copy these values into the template, import them, then verify your results.

Assume all other attributes are empty.

DOO_ORDER_HEADERS_ALL_INT

Attribute	Value
SourceTransactionIdentifier	PMC_IMP_161208
SourceTransactionSystem	OPS
SourceTransactionNumber	PMC_IMP_161208
SourceTransactionRevisionNumber	1
BuyingPartyIdentifier	300000001469001
BuyingPartyName	Computer Service and Rentals
BuyingPartyType	ORGANIZATION
TransactionalCurrencyCode	USD
TransactionalCurrencyName	US Dollar
TransactionOn	2016/12/08 10:10:10
RequestingBusinessUnit	DOO CSPS Business Unit 1
Comments	Testing Order Import
BatchName	10810
RequestingLegalEntity	IFU USA Ltd.
PartialShipAllowedFlag	N
PricedOn	2016/12/08 10:10:10
FreezePriceFlag	N
FreezeShippingChargeFlag	N
FreezeTaxFlag	N

Attribute	Value
CreateCustomerInformationFlag	N

DOO_ORDER_LINES_ALL_INT

Attribute	Value
SourceTransactionIdentifier	PMC_IMP_161208
SourceTransactionSystem	OPS
SourceTransactionLineIdentifier	1
SourceTransactionScheduleIdentifier	1011
SourceTransactionScheduleNumber	11
SourceTransactionLineNumber	1
ProductNumber	AS54888
OrderedQuantity	3
OrderedUOMCode	`zzy`
OrderedUOM	Ea
RequestedFulfillmentOrganizationCode	DOOCSPS1
BusinessUnitName	DOO CSPS Business Unit 1
RequestingBusinessUnitName	DOO CSPS Business Unit 1
SubstitutionAllowedFlag	N
RequestedShipDate	2016/12/09
PaymentTerm	30 Net
TransactionCategoryCode	ORDER
PartialShipAllowedFlag	N

Attribute	Value
ContractStartDate	2014/09/26
ContractEndDate	2019/09/26

DOO_ORDER_ADDRESSES_INT

Attribute	Value in Record 1	Value in Record 2
SourceTransactionIdentifier	PMC_IMP_161208	PMC_IMP_161208
SourceTransactionSystem	OPS	OPS
SourceTransactionLineIdentifier	1	1
SourceTransactionScheduleIdentifier	1011	1011
AddressUseType	SHIP_TO	BILL_TO
PartyName	Computer Service and Rentals	Not applicable
CustomerName	Not applicable	Computer Service and Rentals
PartySiteIdentifier	300000001469004	Not applicable
AccountSiteIdentifier	Not applicable	300000001469016
AddressLine1	2164 Broadway	777 Oracle Blvd
City	TEMPE	COLORADO SPRING
PostalCode	85282	80921
State	AZ	CO
Country	United States	United States
PartyType	ORGANIZATION	ORGANIZATION

Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- How Order Management Transforms Source Orders Into Sales Orders

Automatically Import Source Orders into Order Management

You can use the Order Import Template to manually import orders from your source system into Order Management. This topic describes how to do it automatically.

You use the ERP Integration Service to upload your completed import template to the server that hosts Oracle WebCenter Content, and to run a scheduled process that imports the uploaded file to interface tables, processes them, and imports each interface record as a sales order. For background details about this service, see *Using the Oracle ERP Cloud Adapter with Oracle Integration*.

Summary of the Set Up

1. Prepare and import your payload.
2. Load interface tables.

Prepare and Import Your Payload

1. Prepare the payload that you will import.

   Use the same format that you apply when you use the Order Import Template. For details, see *Overview of Importing Orders Into Order Management*.
2. Save your payload as a zip file. For this example, use my_file.zip.

   This is the file you create when you click Generate CSV File in the template. For details, see *Import Orders Into Order Management*.
3. Use your favorite utility to encode my_file.zip to the BASE64 encoding format.
4. Prepare the erpIntegrationService payload.

   Here's an example payload.

```
<soapenv:Envelope xmlns:erp="http://xmlns.oracle.com/apps/financials/commonModules/shared/model/
erpIntegrationService/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">
 <soapenv:Header>
 <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <wsu:Timestamp wsu:Id="TS-F2664B61E2C11459A614776888804392">
 <wsu:Created>2016-10-28T21:08:00.439Z</wsu:Created>
 <wsu:Expires>2016-10-28T21:09:00.439Z</wsu:Expires>
 </wsu:Timestamp>
 <wsse:UsernameToken wsu:Id="UsernameToken-3DCAF6FF7D37834B0714678327222631">
 <wsse:Username>XXXXXXX</wsse:Username>
 <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">ZZZZZZZZ</wsse:Password>
 </wsse:UsernameToken>
 </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
 <typ:uploadFileToUcm>
 <typ:document>
```

```
 <erp:Content>UEsDBAoAAAAAAHywWkkAAAAAAAAAAAAAAAAAKAAAATkVXIEVSUk9SL1BLAwQUAAAACABqsFpJ46UaP4sCAAC8BgAAGQAAAE5FVy
+0/wFlz2BDSdIggtZk6RZ1qzaF7qEvkwu3YBVsZps09K
+fSfgRUrRJ3SJFwnfffb7v7jh8yUkObOet2A5SnoOxz1Imvdo8HyVK5R5CMkogI9LS3splcbFF1QOCOg6N6khV5m3UwaKxJErBiniGSJ5LzZUhqs
+FtBmKKKguwbyg2IHY3gu4P0bdo7Ct6+MQyjFfMJSAziaKzsT1KCt4GoEDrYaFBZIdUd00CpNPd8ZDcSKnirIeaRlkAklaaWyQ7KNQA5GLsI2+jo
+tc3mfoB9sq0LWztZdzRF69l77EX6lCslr7Ve5RceCHNQGvLcqM6rbXEcGO6y9lq4i7H2L10l6vJYvXBdqfT6jiZzvDEuezxNFRLAbpPceBge2La
BFYq7Rn/avo6YDoE+pA0OcnwoiVEBHzkqTvyZamUGTVUPuojz0haom+EimfuIiNUM/
r65tf1HeY6qAqF/ynTqMagHfNDaEeuFFww1MuM67fJrCdizrDBjKY4S1nERgrFvGYsu2/
pVm9YKbulyRbMNtxrbJcEAkTd0EZEeUoeE7uZ8l1sru7Thx386twx4/
p7dV87qMupf86jDVxbza6LYF6a6JeKWhop7SLZsHjsiPQy8j7VYAoW1Nj1SUjjD4TRTlb8hiCm+XnH9cffTTo/
FP0Og4u8PFn45k9dp2XJOv4BUW7b9cKslMO3TY8nvloCHROogTAlypB20e98zlQFg8tVzVKwYWPBu1d7c9q56OzEneG5ksV/
AZQSwMEFAAAAAgAdLBaSaddoJiUAQAAcQQAABoAAABORVcgRVJST1IvSU5WX1Jlc3BvbnNlLnR4dKVUwU6DQBC9m/
gPG86WQT1JthxqMGrUmlaNnsx2GSnNskt2t7T9e4EWS1tqNF4I+2bevNk3DBRl7ocyR6EyJMtUSOMXUN
+ZWpv5AIZPMWXGLSJGscxVOobyBXDDASc4PSGEVIVukUWooYUMVLRaH2vkhs2FbaAS/
SoRriIMyvgYdY6awhY9yDVWJzIO7gfD3sWV53k+eZW4zJBbjAguOWY2UZJwNo
+n1iczljNXMBm7T3MhnlUiLeqwTjsjqYn7soj4pHxulAuNUqOtHaFliWhDJWqlqXpOOIZaK/2IxrC49rKINl5WSOEg4wJdrlJg0ReYnAOWPAPOhr
cLzzuH98WFcjaSXSGOZ5OjsNVT3VFlX2UOhOXenpuuW/2Lorwyl0K7erW2wmHZiV8E4fAtHdy8fn
+FoNBxR2Al2c9cDIYVfvl1l2HeOzKLDon/K1/TGm2vBjHliKQYz9pM5FI7QDm8I2yt2RI
+U2Xxds4ly7yfqr7IUjji4swew1xSFvb2m0Fr99aH
+xQTfUEsBAh8ACgAAAAAfLBaSQAAAAAAAAAAAAAAAAoAJAAAAAAAAAAQAAAAAAAAE5FVyBFUlJPUi8KACAAAAAAAAEAGAAqIGJSsy/
SASogYlKzL9IBIGmkTLMv0gFQSwECHwAUAAAACABqsFpJ46UaP4sCAAC8BgAAGQAkAAAAAAAAACAAAAAoAAAATkVXIEVSUk9SL0lOVl9SZXF1ZXN
SAVBLAQIfABQAAAAIAHSwWkmnXaCYlAEAAHEEAAAaACQAAAAAAAAAIAAAAOoCAABORVcgRVJST1IvSU5WX1Jlc3BvbnNlLnR4dAoAIAAAAAAAAQA
SAVBLBQYAAAAAAwADADMBAAC2BAAAAAA=</erp:Content>
 <erp:FileName>E:\WIP\errors.zip</erp:FileName>
 <!--Optional:-->
 <erp:ContentType>zip</erp:ContentType>
 <!--Optional:-->
 <erp:DocumentTitle>PCTitle002</erp:DocumentTitle>
 <!--Optional:-->
 <erp:DocumentAuthor>Diane Cho</erp:DocumentAuthor>
 <!--Optional:-->
 <erp:DocumentSecurityGroup>FAFusionImportExport</erp:DocumentSecurityGroup>
 <!--Optional:-->
 <erp:DocumentAccount>scm$/sourceSalesOrder$/import$</erp:DocumentAccount>
 <!--Optional:-->
 <erp:DocumentName>PCName002</erp:DocumentName>
 <!--Optional:-->
 <erp:DocumentId>002</erp:DocumentId>
 </typ:document>
 </typ:uploadFileToUcm>
 </soapenv:Body>
</soapenv:Envelope>
```

Note

Code	Value
`UsernameToken`	Add this token while you're creating the payload in SOAP UI. a. Open your payload in SOAP UI. b. Click **Auth**. c. In the Authorization area, enter your user name and password. d. Position your cursor in the payload where you need to add the token. In this example, position it immediately after the **Timestamp** tag. e. Right-click in the payload, then click **Add WSS Username Token**. SOAP UI adds the entire UsernameToken tag to your payload. For details, see *How to Use SoapUI to Invoke a Web Service (Doc ID 2234114.1)*.
`erp:content`	Insert the BASE64 content that you encoded earlier in this procedure into **erp:content**.

Code	Value
	`erp:content` stores the details of the file that you're importing in BASE64. This BASE64 content represents the sales order data that you created through file-based data import.
`FileName`	If your upload fails, examine the errors.zip file. It contains details about why the upload failed.
`DocumentTitle`	Use PCTitle002.
`DocumentAuthor`	Enter your name. You can use it later when you verify the upload. For this example, assume your name is Diane Cho.
`erp:DocumentAccount`	Use `scm$/sourceSalesOrder$/import$` because it identifies the account you use when you run the scheduled process that imports sales orders later in this procedure.
`DocumentName`	Enter any alphanumeric value. Use a maximum value of 30 characters. This example uses PCName002.

Learn about the attributes in this payload. For details, go to *SOAP Web Services for Financials*, then expand **Business Object Services > ERP Object Descriptive Flexfield Update Service > Document Information**.

5. Use this URL to access the WSDL for the service.

   `https://servername/fscmService/ErpIntegrationService?WSDL`

6. Use the uploadFiletoUcm operation to upload and submit your payload.

   go to *SOAP Web Services for Financials*, then search for uploadFiletoUcm.

7. Notice the document Id that the submit returns. You use it when you verify the scheduled process.
8. Verify the import.
   - Sign into Oracle Applications, then go to the File Import and Export work area.
   - On the Overview page, enter Diane Cho in the Owner attribute, then click **Search**.
   - Examine the search results and verify that it contains the errors.zip file and the sourceSalesOrder file.

File	Account	Owner
errors.zip	scm/sourceSalesOrder/import	Diane Cho
sourceSalesOrder	scm/sourceSalesOrder/import	Diane Cho

File	Account	Owner

The sourceSalesOrder file is now on the server. You will reference next when you run the scheduled process.

Load Interface Tables

1. Make a SOAP call.
   Here's an example payload you can use when you use the submitESSJobRequest operation in your SOAP UI call.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">
 <soapenv:Header>
 <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
 <wsu:Timestamp wsu:Id="TS-F2664B61E2C11459A614776898797693">
 <wsu:Created>2016-10-28T21:24:39.768Z</wsu:Created>
 <wsu:Expires>2016-10-28T21:25:39.768Z</wsu:Expires>
 </wsu:Timestamp>
 <wsse:UsernameToken wsu:Id="UsernameToken-3DCAF6FF7D37834B0714678327222631">
 <wsse:Username>XXXXXXX</wsse:Username>
 <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">ZZZZZZZ</wsse:Password>
 </wsse:UsernameToken>
 </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
 <typ:submitESSJobRequest>
 <typ:jobPackageName>/oracle/apps/ess/financials/commonModules/shared/common/interfaceLoader</
typ:jobPackageName>
 <typ:jobDefinitionName>InterfaceLoaderController</typ:jobDefinitionName>
 <typ:paramList>72</typ:paramList>
 <typ:paramList>24131</typ:paramList>
 </typ:submitESSJobRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

   Note
   - `typ:paramList>72` identifies the import process you use with the uploadFiletoUcm operation. The value 72 indicates that you imported a sales order. To determine this value, see *How to Find Payload Parameter for Load Interface File For Import Web Service (Doc ID 2071025.1)*.
   - `<typ:paramList>24131` identifies the file you import when you use with the uploadFiletoUcm operation.
2. Use the getESSJobStatus operation in your SOAP UI call to get the status of your scheduled process.
   Here's an example payload.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/financials/commonModules/shared/model/erpIntegrationService/types/">
 <soapenv:Header>
 <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd">
```

```
 <wsu:Timestamp wsu:Id="TS-F2664B61E2C11459A614776903991084">
 <wsu:Created>2016-10-28T21:33:19.108Z</wsu:Created>
 <wsu:Expires>2016-10-28T21:34:19.108Z</wsu:Expires>
 </wsu:Timestamp>
 <wsse:UsernameToken wsu:Id="UsernameToken-3DCAF6FF7D37834B0714678327222631">
 <wsse:Username>XXXXXXX</wsse:Username>
 <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">ZZZZZZZ</wsse:Password>
 </wsse:UsernameToken>
 </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
 <typ:getESSJobStatus>
 <typ:requestId>55562</typ:requestId>
 </typ:getESSJobStatus>
 </soapenv:Body>
</soapenv:Envelope>
```

`typ:requestId>55562` identifies the scheduled process that runs when you submit the submitESSJobRequest operation.

Related Topics

- Import Orders Into Order Management
- Overview of Importing Orders Into Order Management

Use Web Services to Import Orders

Use an order import service to receive a request for details about an order line from your source system, and then transform it into a structure that Order Management can use.

Use the Order Import web service to capture sales orders from your source system. For details, see *Web Services That You Can Use to Integrate Order Management*. Use example payloads. For details, see *Example Web Service Payloads That Integrate Order Management*.

Here are some other web services that you can also use.

Web Service	Description

Get Order Shipping Details	• Communicate shipping details to the source system that requests it.
	  ○ Current schedules
	  ○ Status of each schedule
	  ○ Schedule details, such as warehouse and shipping method
	• Organizes the reply according to shipments instead of fulfillment lines.
	• Include details about fulfillment lines in the reply. More than one fulfillment line in Order Management might reference a single order line in the source order.
	• Return details about the order lines of a sales order, or for only a subset of these lines.

Apply a Hold	Route a request for a hold from your source system to a hold task. Process more than one hold in a request, each hold for one sales order, or more than one order line in the sales order.
Release a Hold	Route a request to release a hold from our source system to the hold task.

Web Service	Description
Check Availability	Send a request to Order Management to determine the quantity that's available on a date in the source system. Order Management sends a reply to the source system that includes these details.

Update Split Fulfillment Lines

You can't use the Order Import web service to update the quantity on a split fulfillment line.

Assume you use Integration Cloud Service to integrate your implementation. The original order line contains a quantity of 100. Global Order Promising analyzes the supply chain and determines the best way to fulfill your order is to split the order into two fulfillment lines, where line 1.1 contains a quantity of 75 and line 1.2 contains a quantity of 25. Promising sends the request to Integration Cloud Service, who then calls the Order Import web service to make the change in Order Management, and Order Management successfully splits the line.

The supply chain is constantly changing. The next day, Global Order Promising analyzes the supply chain again and determines to use a quantity of 60 on line 1.1 and 40 on line 2.2. So you now must change the quantity on line 1.1 from 75 to 60, and the quantity on line 1.2 from 60 to 40.

You use the Order Import service to change the values.

- source line number = 1, fulfillment line number = 1, quantity = 60
- source line number = 1, fulfillment line number = 2, quantity = 40

But you encounter a null pointer error because the import can't locate the fulfillment lines.

You use the Order Import service again to change the values but this time specify different source lines so they match the fulfillment lines.

- source line number = 1, fulfillment line number = 1, quantity = 60
- source line number = 2, fulfillment line number = 2, quantity = 40

The import updates the quantity on line 1 to 60, cancels line 1.2, and creates a new line 2 with a quantity of 40.

Some time later, you receive a request to update only fulfillment line 1.1. You use the Order Import service to successfully update line 1.1, but the import automatically cancels line 2.

You can only do these kinds of updates to split fulfillment lines in the Order Management work area or through REST API. You can't use the Order Import service to do them.

If you must update a split fulfillment line, then see *Use FBDI and REST API to Import Large Volumes of Sales Orders*, and see the Display Line Numbers subtopic in *Import Different Kinds of Data*.

Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- Web Services That You Can Use to Integrate Order Management
- Example Web Service Payloads That Integrate Order Management
- How Order Management Transforms Source Orders Into Sales Orders

Specify the Orchestration Process When You Import

Improve performance and decrease the time it takes to create a sales order. You can specify the orchestration process name when you import a sales order through file-based data import (FBDI) or REST API instead of having to use a rule to assign the process.

Use this feature when you know which orchestration process you want to use and don't have complex business logic that requires you to use an assignment rule to select from different orchestration processes according to attribute values from the sales order.

- If you specify the orchestration process name for any line in the sales order, then Order Management won't call the assignment rule for any line in the order. So you must specify the orchestration process name for each line in the order.
- You must use only an assignment rule to assign the orchestration process for all order lines, or assign each line in your import payload when you import. If you use an assignment rule to assign an orchestration process to a sales order, then you must use that same rule to add a new line to the same order. You can't use the rule to assign the process and then at some later time import a new line where you specify the orchestration process in your import payload.

File-Based Data Import

Use the SourceSalesOrderImportTemplate.xlsm template.

Use the Process Name column in the DOO_ORDER_LINES_ALL worksheet sheet.

To get the template, go to *File-Based Data Import (FBDI) for Oracle SCM*. Locate the Order Management chapter, then click **Import Sales Orders**.

For details, see *Overview of Importing Orders Into Order Management*.

REST API

Use the salesOrdersForOrderHubRequests resource.

Here's an example payload. It assigns the fulfillment line to the ManualSchedulingProcess orchestration process.

```
"lines":[
{
"SourceTransactionLineId":"1",
"SourceTransactionLineNumber":"1",
"SourceTransactionScheduleId":"1",
"SourceScheduleNumber":"1",
"TransactionCategoryCode":"ORDER",
…
"OrchestrationProcessName": "ManualSchedulingProcess", // <= Use this code to specify the orchestration
 process name
…
}
]
```

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Sales Orders for Order Hub**.

If you use REST API, then you must enable the Import and Fulfill Large Volumes of Sales Orders opt-in feature. For details, see *Import and Fulfill Large Volumes of Sales Orders*.

Order Management Extension

If you import through REST API, then use the OrchestrationProcessName attribute in your extension to specify the orchestration process name on the order line, and use the On-Save or On Start of Submission Request extension point. For example:

```
line.setAttribute("OrchestrationProcessName","ManualSchedulingProcess");
```

For details, see *Overview of Creating Order Management Extensions*.

Import a Large Volume of Sales Orders

Import and Fulfill Large Volumes of Sales Orders

Improve the performance of your order-to-cash flow when you need to fulfill a bunch of sales orders, including orders that have a wide range of order lines.

Use this feature to fulfill a large number of sales orders through Oracle Global Order Promising, Oracle Shipping, and Oracle Accounts Receivable.

Use this feature so you can:

- Improve performance and reduce the time it takes to import a large volume of sales orders. The *Import Sales Orders* scheduled process helps you do this.
- Use pricing algorithms to efficiently price your sales orders.
- Use the Import and Fulfill Large Volumes of Sales Order feature and Global Order Promising's High Volume Order Promising feature together to promise a large volume of sales orders.
- Confirm shipment for a large volume of sales orders. Reduce the need to manually move the shipped order line to the next step.
- Invoice a large volume of sales order so that you can reduce delays in getting paid from invoices. Reduce the need to manually move the invoiced order line to the next task.

Realize these benefits.

- Receive, import, and fulfill a large volume of sales orders.
- Minimize the need for an Order Manager to manually intervene and take corrective action on the sales order.

This topic describes the behavior that Order Management applies when you use the Import and Fulfill Large Volumes of Sales Order feature.

How It Works

Import

You use the Import Sales Orders scheduled process to import your sales orders.

Ship

If you haven't enabled the Defer Sending Inventory Updates To Integrated Applications option, and if Oracle Shipping confirms the shipment, then:

1. Oracle Shipping runs the *Manage Shipment Interface* scheduled process to add shipment details to an interface table, and then runs the *Process Responses from Order Fulfillment* scheduled process.

   Process Responses from Order Fulfillment sends the shipment details that are in the interface tables to Order Management.
2. Order Management updates shipment details on the fulfillment line and moves the shipped line to the next fulfillment task.
3. The Manage Shipment Interface scheduled process runs the Process Responses from Order Fulfillment scheduled process. This flow doesn't run the *Send Shipment Advice* scheduled process to reduce the total number of scheduled processes that are running.
4. If the scheduled process can't process a shipment from the interface table, and if Oracle Shipping already confirmed the shipment, then the order line in Order Management will have an error message. See the troubleshooting section later in this topic.

For details about Defer Sending Inventory Updates To Integrated Applications option, see *Fix an Order Status That Doesn't Update to Shipped*.

Invoice

If Oracle Receivables has billed the lines, then:

1. Oracle Receivables runs the Notify Feeder System of Receivables Transactions scheduled process, adds billing details to the interface table, then runs the Process Responses from Order Fulfillment scheduled process. For details, see *Get Things Moving Again in Accounts Receivable*.
2. Process Responses from Order Fulfillment sends billing data from the interface tables to Order Management.
3. Order Management updates the invoice details on the fulfillment line and moves the billed line to the next fulfillment task.
4. The Import Receivables Transactions Using AutoInvoice scheduled process creates invoices in Oracle Receivables from sales order data, and then runs the Notify Feeder System of Receivables Transactions only one time. It does this to reduce the total number of scheduled processes that are running.
5. If the scheduled process can't process a billing from the interface table, and if Oracle Receivables already billed the order, then the order line in Order Management will have an error message. See the troubleshooting section later in this topic.

Guidelines

Import

- If you use File Based Data Import to import your order, then Order Management won't run pretransformation, transformation, or posttransformation rules to validate, modify, or set default values for attributes on the order header or the order line. Instead, we recommend that you use an order management extension to do your transformations. For details, see *Overview of Creating Order Management Extensions*.
- You can't create a customer during import. If you need to create a customer, then you must use Customer Management to do it during your set up.
- If it takes longer than 5 minutes to import a large sales order through REST API, then you might encounter an error that indicates the import timed out, such as a 504 Gateway Timeout error. This doesn't mean the import failed. The import will continue to run in the background. You can use a GET request with the salesOrdersForOrderHub REST API to get an update on the order's status.

Orchestrate

We recommend that you use the predefined DOO_ScheduleShipInvoice orchestration process. It doesn't have branching conditions or a reservation task so it provides an efficient way to fulfill your sales order. Use it to schedule, ship, and invoice each of your sales orders.

- To avoid performance problems, don't use an assignment rule to assign the orchestration process to the order line. Instead, specify the orchestration process name in your import payload or use an order management extension to specify the orchestration process. Make sure you specify the orchestration process for all lines in the sales order.
- Don't override the fulfillment line's status in the orchestration process after you ship or bill the line. Order Management will set the status to Shipped or Billed, and you can't change that value.
- Don't set up a custom rule to filter the lines that orchestration schedules, ships, or invoices. Order Management will use its predefined rules to pick the lines that orchestration can schedule, ship, or invoice.
- Don't add a parallel step or a subprocess step to your orchestration process.

Summary of the Set Up

1. Set up parameters and profiles.
2. Update your pricing algorithms.

Set Up Parameters and Profiles

1. Go to the Setup and Maintenance work area.
2. Use the Manage Order Management Parameters task to set the parameters.

Parameter	Value
Use Pricing Algorithms to Calculate Totals for Sales Order	Yes
Number of Processes for Order Import	40 Adjust this value during your performance testing, as necessary.

   See *Manage Order Management Parameters*.
3. Use the Manage Order Profiles task.
   - Enable the Respond Immediately on Start of Submission Request order profile so Order Management can start the import right away.
   - Create a new profile option.

Attribute	Value
Profile Option Code	FOM_HVOP_SHIP_NUM_PROCESS
Profile Display Name	Number of Child Scheduled Processes for High Volume Sales Orders
Application	Order Management
Module	Order Management

Attribute	Value
Description	We're using this profile option to specify the number of child processes to run for each instance of the Process Responses from Order Fulfillment scheduled process.
SQL Validation	Leave empty.
Start Date	Set it to yesterday.

- Create another profile option.

Attribute	Value
Profile Option Code	FOM_HVOP_SHIP_BATCH_SIZE
Profile Display Name	Number of Responses to Process for High Volume Sales Orders
Application	Order Management
Module	Order Management
Description	We're using this profile option to specify the total number responses to process for each instance of the Process Responses from Order Fulfillment scheduled process.
SQL Validation	Leave empty.
Start Date	Set it to yesterday.

- Use the Profile Values area to set the values for each of your new profile options.

Attribute	Value
Profile Level	Site
Profile Value	Yes

See *Use Order Profiles to Control Order Management Behavior*.

4. Use the Manage Administrator Profile Values task.
   - Set the value for the new profile option that you just created.

Attribute	Value
Profile Display Name	Number of Child Scheduled Processes for High Volume Sales Orders
Profile Level	Site

Attribute	Value
Profile Value	10

- Set the value for the other new profile option that you just created.

Attribute	Value
Profile Display Name	Number of Responses to Process for High Volume Sales Orders
Profile Level	Site
Profile Value	500

- Adjust these values during your performance testing, as necessary.

Update Your Pricing Algorithms

Make sure you have the latest version of the Price Sales Transactions pricing algorithm. Do the Pre-Update and Post-Update steps that are described in the Pricing section of Oracle Fusion Cloud SCM: Performing Your Quarterly Update. For details, see *Promote Pricing Algorithms Into the Latest Update*.

Process Responses from Order Fulfillment

If Order Management can't update the shipment details or invoice details on the order line because of a system error, then the order line might not move to the next fulfilment task in the orchestration process even though Shipping successfully shipped and Receivables successfully invoiced the line.

A system error might happen for a variety of reasons:

- The fulfillment line in Order Management isn't on a wait step.
- Order Management is still processing an earlier response from the fulfilment system on the line.
- The scheduled process processes more than one shipment or billing detail for the same line in quick succession.
- The server is slow and times out, there's a network connectivity problem, and so on.

As a result, you might notice:

- Shipping shipped your fulfillment line but the order line in Order Management doesn't contain the latest shipping details.
- Accounts Receivable billed your fulfillment line but the order line in Order Management doesn't contain the latest billing details.

To fix this problem, we recommend that you set up the *Process Responses from Order Fulfillment* scheduled process to run on a schedule at regular intervals. This scheduled process will automatically update the line with the latest details from Oracle Shipping and Oracle Receivables. For example, it will automatically update:

- Shipping details on the fulfillment line in Order Management after Shipping confirmed the shipment but the line remains in Awaiting Shipping status.
- Invoice details on the fulfillment line in Order Management after Receivables billed the invoice but the line remains in Awaiting Billing status.

You can also run it manually while troubleshooting. If you do, set these parameters to filter the data that the scheduled process will look at.

Parameter	Description
Task Type	Set a value: • **Invoice**. Update billing details. • **Shipment**. Update shipping details. Note • You can only process a response that you receive from Oracle Shipping or Oracle Accounts Receivable. You can't use this scheduled process to process a response from any other fulfillment system, such as through your own task type. For details, see *Create Your Own Task Type*. • If you set up this scheduled process to run on a schedule, and if you need to process shipments and invoices, then create one instance of the process for shipments, and another instance for invoices. For details, see *Fulfillment Tasks*.
Record Set	Set it to Process Records That Failed. Don't set it to Process Records That Are Not Yet Processed. That value is for Oracle internal use only.
From Date To Date	Filter the data that you want to process according to when Order Management received the fulfillment response.
Incremental From Load Request To Load Request Number of Processes Batch Size	Leave these parameters empty. They are for Oracle internal use only.

Note

- You can use this scheduled process only to process responses that you receive from Oracle shipping or Oracle Accounts Receivables. You can't use it with any other fulfillment system.
- Shipping and Accounts Receivable store their details in the DOO_FS_FULFILL_LINES_INT table. If the scheduled process encounters an error, then it marks each record that's in error in DOO_FS_FULFILL_LINES_INT.
- For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

Use FBDI and REST API to Import Large Volumes of Sales Orders

Use the Import and Fulfill Large Volumes of Sales Orders opt-in feature with file based data import (FBDI) when you have a large volume of sales orders to import. Using them together improves efficiency and performance.

You can import:

- Standard items, models, kits, covered items, and coverage items.
- Items that you price in Pricing Administration or that you price in your source system.
- Referenced and unreferenced returns.
- Change orders and revisions. You can import only the order line or the attributes that you changed. You don't have to import the entire order or all the attributes.

Import different kinds of entities.

- Order headers, orders lines, and addresses
- Charges and charge components
- Manual price adjustments
- Extensible flexfields on the order leader and order line
- Sales credits and billing plans
- Lot serial details
- Payments
- Transaction attributes

Try it.

1. Enable the Order Management feature.
    - Go to the Setup and Maintenance work area.
    - Select the Order Management offering.
    - Click **Change Feature Opt In**.
    - On the Opt In page, in the row that has Order Management in the Name column, click the **pencil**.
    - On the Edit Features page, add a check mark to the Import and Fulfill Large Volumes of Sales Order feature.

      You can't opt out of this feature after you opt in and submit a sales order. At this point, you're committed to using the feature.
    - Click **Done**.
2. Enable the Order Promising feature.
    - Select the Supply Chain Planning offering.
    - Click **Change Feature Opt In**.
    - On the Opt In page, in the row that has Global Order Promising in the Name column, click the **pencil**.
    - On the Edit Features page, add a check mark to the High Volume Order Promising feature.
    - Click **Done**.
3. Use file-based data import to import your source orders.

The import will automatically route your import requests through REST API.

For details, see *Overview of Importing Orders Into Order Management*.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Guidelines

Apply these guidelines when you use the Import and Fulfill Large Volumes of Sales Order Opt-in Feature.

Functionality	Import Without the Feature	Import With the Feature
Apply Business Rules	The import will apply your pretransformation, product transformation, and posttransformation rules on your import data.	The import won't apply these rules. However, you can create an order management extension to implement the logic that you might normally get from these rule. If you do create an extension, then set it up to run on the On Save extension point or on the On Start of Submission Request extension point.
Create a Customer	You can create a customer during the import.	You can't create a customer during the import. You can't use the Create Customer Information Flag attribute to create a customer. If you must create a customer when you import, then we recommend that you don't enable the opt-in but instead import through file-based data import without REST API, or import only through REST API. For details about this attribute, see *Import Customer Data for Your Sales Orders*.
Set Default Values	The import attempts to set default values for attributes that contain customer details just like the Order Management work area does when you create a sales order. Assume you import only the Sold-to Party but not the Sold-to Contact, Ship-to Site, or Contact Point. REST API will use the value in Sold-to Party to set the default value for Sold-to Contact, Ship-to Site, and Contact Point according to your setup in Customer Data Management.	The import won't set default values for attributes that contain customer details. For details about Customer Data Management and the party model, see *Overview of Displaying Customer Details on Sales Orders*.
Cascade Values	The import cascades the values for attributes that contain billing and fulfillment details from the order header to the order lines just like the Order Management work area does when you create a sales order, such as ship-to attributes and bill-to attributes. The import also cascades attributes from the root line of a configured item to the child lines of that item, including values from the segment of an extensible flexfield.	The import won't cascade any of these values. Your import template must contain all of them.
Use Case Sensitive Values	The attribute values in your import don't have to use the same upper case and lower case values that the Oracle database uses.	The attribute values in your import must use the same upper case and lower case values that the Oracle database uses. For example, if the Product Number attribute in the Oracle database contains the value

Functionality	Import Without the Feature	Import With the Feature
		AS54888, then the Product Number attribute on the DOO_ORDER_LINES_ALL_INT worksheet of your import template must also contain AS54888. Your import template must not contain as54888, aS54888, As54888, and so on.
Cross-Reference Attributes	You can import your own custom values.	You can't import your own custom values. If Order Management cross-references an attribute, then your import must use the same value that the Order Orchestration and Planning Data Repository contains for that attribute. Assume you want to use a truck to ship the order line. The Order Orchestration and Planning Data Repository contains the value FEI TRUCK for the ShippingServiceLevel attribute, so the Shipping Service Level attribute on the DOO_ORDER_LINES_ALL_INT worksheet of your import template must also contain FEI TRUCK. For details, see *Integrate Names and Codes Between Source Systems and Order Management*.
Revise Your Sales Order	You must reimport the entire sales order, including the changes for the revision.	You only have to import the values that you're changing and a few other values that identify the order. Assume you need to update the charge component on an order line. Your payload only has to include the revised value for the charge component and a few other attributes that identify parts of the sales order, such as the line that contains the charge, and values that identify the order header.
Report Errors	Your import stops when it encounters the first error, and reports only that error.	REST API continues to process your import even after it encounters the first error, then reports all the errors that it found.

Use REST API to Import Large Volumes of Sales Orders

Take advantage of REST API's superior performance when you need to import a large volume of sales orders.

Try it.

1. Use the Sales Orders for Order Hub Requests REST resource to stage your source orders into interface tables.

   For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click **Sales Orders for Order Hub**.

2. Run these scheduled processes.
    - *Load Interface File for Import*
    - *Import Sales Orders*
    - *Delete Orders from Interface Tables*

    Each process creates a separate data file after it finishes the import. You can use them to identify successful imports, and also to identify orders that failed import.

    For details, see *Import Orders Into Order Management*.

Use Order Management Extensions to Import Large Volumes of Sales Orders

Use an order management extension when you have a large volume of them to import.

Assume you need to process 100,000 order lines each hour and over 1,000,000 order lines each day. We recommend that you use an order management extension instead of a transformation rule. An extension provides superior performance when processing a high volume of lines.

For details about using extensions, see *Overview of Creating Order Management Extensions*.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Guidelines

Make sure your extension:

- Creates the customer before you import the order.
- Identifies the item.
    - Sets the value for the ProductIdentifier attribute or the ProductNumber attribute on the order line. Make sure the value uniquely identifies the item.
    - Make sure the Product Information Management work area contains the item that these attributes reference.
- Sets the Ordered Quantity attribute on the order line to a positive, numeric value.
- Adds the new order line on the On Save or the On Start of Submission Request event. Don't add the line on the End of Submission event.
- Adds the new line to a sales order that isn't closed or canceled. If the sales order is closed or canceled, then you can't add a new line to that order.
- Adds a new line that references an existing order line only when a transformation rule hasn't already transformed the existing line.
- Doesn't add a configured item to a sales order. You can't use an extension to add a configured item to a sales order.
- Doesn't add a new order line to a sales order during the End of Submission event.

A transformation rule automatically does a number of actions. However, if you use an extension, then you must code your extension to do these actions:

- Set the default quantity of the order line that you're adding according to the quantity on the source line.
- You must cascade or manipulate the quantity according to the source line, or specify a quantity according to some other attribute.

- If you modify or remove the source line, then your extension must modify or remove the new line that it created for the source line.

Note

- You must enable the Import and Fulfill Large Volumes of Sales Orders opt-in feature. For details, see *Use FBDI and REST API to Import Large Volumes of Sales Orders*.
- As an option, you can enable the Respond Immediately on Start of Submission Request order profile. If you do, the Sales Orders For Order Hub REST service will respond when it starts to submit the sales order. If you don't enable it, then REST responds after it finishes submitting the sales order.
- REST API can't call an oracle business rule, but you can use it with all three extension points.
- You can't use extensions and transformation rules to add new lines at the same time. You must use extensions or transformation rules.
- Transformation rules and extensions set the price of the new line to 0.
- You can't edit the new line in the Order Management work area.

Examples

Assume the examples in this subtopic use these values.

ProductIdentifier	ProductNumber
149	AS54888
2157	AS92888

Add More Than Order One Line for a Source Line

Assume you need to import item AS54888 and transform it into two separate order lines. One line will have the AS92888 item with a quantity of 15, and the other line will have the AS54888 item with a quantity of 20.

```
import oracle.apps.scm.doo.common.extensions.CreateLineParams;

if( !"CREATE_MUL_LINES".equals( header.getAttribute("CustomerPONumber" ) ) ) return;

def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {
 def line = lines.next();

//If the source line is closed, cancelled, or already transformed, then you can't add it to the sales
 order.
// See if the source line is closed.
 def isClosed = line.isClosed()

 // See if the source line is canceled.
 def isCanceled = line.isCanceled()

 // See if the source line is already transformed.
 def isTransformed = line.isTransformed();

 if (isClosed || isCanceled || isTransformed) { // Skip if the line is closed, canceled, or already
 transformed.
 continue;
 }//Get the name of the item that's on the order line.
 def productName = line.getAttribute("ProductName");
```

```
 if (isClosed || isCanceled || isTransformed || productName != 'AS54888') { // Skip this section if the line
 is closed, canceled, already transformed, or the item on the line isn't AS54888.

 continue;
 }

 def orderedQuantity = line.getAttribute("OrderedQuantity");

 //// Add the first transformed line AS92888
 def createLineParams = new CreateLineParams();
 createLineParams.setProductNumber("AS92888");
 //createLineParams.setProductIdentifier(2157);
 createLineParams.setOrderedUOM("Each")
 createLineParams.setOrderedQuantity(15);

 def transformedLines = line.getTransformedLines("AS92888");
 //def transformedLines = line.getTransformedLines(2157);
 if (transformedLines.size() == 0) { // If you haven't already added the line, then add it now.
 line.createNewLine(createLineParams);

 }

 //// Add the second transformed line AS54888
 createLineParams.setProductNumber("AS54888");
 //createLineParams.setProductIdentifier(149);
 createLineParams.setOrderedUOM("Each")
 createLineParams.setOrderedQuantity(20);

 transformedLines = line.getTransformedLines("AS54888");
 //transformedLines = line.getTransformedLines(149);
 if (transformedLines.size() == 0) { // add new line if it is not created before
 line.createNewLine(createLineParams);
 }
}
```

Add a Line According to Conditions

You can specify to add a new order line only if the source line meets a condition. Assume you want to add a line only if the import payload contains the AS54888 item and its quantity is 100.

```
import oracle.apps.scm.doo.common.extensions.CreateLineParams;

if( !"CREATE_LINE_IF_AS54888&QTY100".equals( header.getAttribute("CustomerPONumber" ) ) ) return;

def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {
 def line = lines.next();

 def isClosed = line.isClosed()
 def isCanceled = line.isCanceled()
 def isTransformed = line.isTransformed();

 if (isClosed || isCanceled || isTransformed) { // Skip if the line is closed or canceled, or if the line is
 already transformed.
 continue;
 }

 def orderedQuantity = line.getAttribute("OrderedQuantity"); // get the original line's quantity
 def productNumber = line.getAttribute("ProductNumber"); // get the original line's product number

 if (productNumber == "AS54888" && orderedQuantity == 100) {

 //def transformedLines = line.getTransformedLines("AS92888");
 def transformedLines = line.getTransformedLines(2157);
```

```
 if (transformedLines.size() == 0) { // If you haven't already created the transformed line, then create it
 now.
 def createLineParams = new CreateLineParams();
 //createLineParams.setProductNumber("AS92888");
 createLineParams.setProductIdentifier(2157);
 createLineParams.setOrderedUOM("Each")
 createLineParams.setOrderedQuantity(orderedQuantity);

 line.createNewLine(createLineParams);
 }
 }
}
```

Cancel Order Lines That You Transformed

If you modify or remove the source line, then your extension must modify or remove the new line that it created for the source line. Canceling the source line won't automatically cancel the transformed lines.

This example removes the transformed lines any time you remove the source lines or canceled lines.

```
import oracle.apps.scm.doo.common.extensions.CreateLineParams;
import oracle.apps.scm.doo.common.extensions.Line;

if( !"CANCEL_TRANSFORMED_LINES".equals( header.getAttribute("CustomerPONumber" ) ) ) return;

def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {

 def line = lines.next();

 def isClosed = line.isClosed();
 def isCanceled = line.isCanceled();
 def isTransformed = line.isTransformed();
 def orderedQuantity = line.getAttribute("OrderedQuantity");

 if (orderedQuantity == 0 && !isTransformed) { //Source line is canceled
 List<Line> transfomrLines = line.getTransformedLines(); //Get all the transformed lines for the current
 line.
 for (Line transformedLine : transfomrLines) {
 def orderedQuantityTL = transformedLine.getAttribute("OrderedQuantity");
 if (orderedQuantityTL != 0 && transformedLine.isOpen()) {
 transformedLine.setAttribute("OrderedQuantity", 0);
 }
 }
 }
}
```

Example That Fails

Assume you create an extension and receive an error.

```
Request failed because order management extension CREATE_LINE_CLOSED_CANCELED attempted to add new line
 that references a closed or canceled line. Happens during event On Save. You can use an order management
 extension to add a new order line that references an existing order line only if the existing line isnt
 closed or canceled.
```

Here's an example of an extension that fails because it attempts to add a new line to a sales order that's closed, canceled, or already transformed.

```
"import oracle.apps.scm.doo.common.extensions.CreateLineParams;
if( !"CREATE_LINE_CLOSED_CANCELED".equals( header.getAttribute("CustomerPONumber" ) ) ) return;
def createLineParams = new CreateLineParams();
createLineParams.setProductNumber("AS54888");
createLineParams.setOrderedUOM("Each")
```

```
createLineParams.setOrderedQuantity(50);
def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();
 def isTransformed = line.isTransformed();
 def transformedLines = line.getTransformedLines();
 if (isTransformed || transformedLines.size() > 0) {
 continue;
 }
 line.createNewLine(createLineParams);
}"
```

To avoid this problem, add this snippet near the beginning of your code.

```
// See if the source line is closed.
 def isClosed = line.isClosed()

 // See if the source line is canceled.
 def isCanceled = line.isCanceled()

 // See if the source line is already transformed.
 def isTransformed = line.isTransformed();

 if (isClosed || isCanceled || isTransformed) { // Skip if the line is closed, canceled, or already
 transformed.
 continue;
 }
```

Import Different Kinds of Data

Import Different Kinds of Data

Apply guidelines when you import different kinds of data.

New Customers

If you use file-based data import (FBDI) to import a customer, contact, or address that doesn't already exist in Trading Community Architecture, then you must manually assign privileges to the role for the user who runs the *Load Interface File for Import* scheduled process.

Privilege	Description
HZ_ENTER_TRADING_COMMUNITY_PERSON_PRIV	Enter Trading Community Person
HZ_UPDATE_TRADING_COMMUNITY_PERSON_PRIV	Update Trading Community Person
HZ_ENTER_TRADING_COMMUNITY_ORGANIZATION_INFORMATION_PRIV	Enter Trading Community Organization Information
HZ_UPDATE_TRADING_COMMUNITY_ORGANIZATION_PRIV	Update Trading Community Organization

Learn how to add a privilege. For details, see *Data Security*.

Draft Sales Orders

In some instances its important to import your sales order in draft status to avoid a problem in downstream fulfillment. For example, assume you import a sales order and need to modify the quantity to avoid having the order get stuck in awaiting shipping status because your inventory levels are fluctuating. You can import it as a draft, modify the quantity in the Order Management work area so it meets your downstream fulfillment requirements, then submit it.

If you don't want to submit your imported order to fulfillment, but instead want to import it as a draft sales order:

Import Technology	Description
FBDI	Set the **`Submit Flag`** attribute to N on the DOO_ORDER_HEADERS_ALL_INT worksheet of the order import template.
REST API	Set the SubmittedFlag attribute to false in the Sales Orders For Order Hub REST API resource. False is the default value in REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.
Web Service	Set the SubmittedFlag attribute to N in your web service payload.

Import a Revision on a Draft

If you use FBDI, REST API, or a web service to import a draft sales order, then do another import that changes the draft, then Order Management assigns a new order number to the draft.

You must use the new order number in any subsequent updates or processing. For example, assume you use FBDI to import source order 57486, and Order Management saves 57486 as a draft. You then use FBDI to import a change to a date attribute on order 57486. Order Management will create a new order number, such as 57490, then save your change in 57490. Order Management deletes all data on order 57486, including any holds that you applied to it. You can't use order 57486 anymore. You must use order 57490.

Revisions

If you import a draft order, modify it in the Order Management work area, submit it, then import a revision of the order in a subsequent import, then the imported revision must include the current attribute values. If you want to change attribute values, then the imported revision must include the new values.

For example, assume you set up an order management extension that sets the default value for the Shipping Method attribute. You import a value of Air for Shipping Method, you change the Shipping Method to Rail in the Order Management work area, submit the order, then import an order revision in a subsequent import. If you want to keep Rail, then your subsequent import must include a value of Rail for Shipping Method. If the subsequent import doesn't include any value for Shipping Method, then Order Management will set Shipping Method to some other value, depending on your pretransformation rule.

RevisionSourceOrderSystem Attribute

You can use the RevisionSourceOrderSystem attribute to identify the source system that sends a revision of the sales order. You can use RevisionSourceOrderSystem in a processing constraint to control the changes that you allow when you import a revision from the source system.

RevisionSourceOrderSystem doesn't affect any other processing that Order Management does when you create a draft of the revision or submit it. For example, Order Management doesn't create a new revision according to how you set RevisionSourceOrderSystem.

Assume you.

1. Import revision 1 for order 45768, with a quantity of 10 on line 1, and RevisionSourceOrderSystem in the import payload for this order contains EastSystem.
2. Import revision 2 for order 45768 from source system EastSystem, RevisionSourceOrderSystem in the import payload contains EastSystem, the import adds line 2 with a quantity of 5, and you keep it in draft status.
3. Import an update for revision 1 from another system, WestSystem, RevisionSourceOrderSystem in the import payload contains WestSystem, the import updates the quantity on line 1 to 7, the quantity on line 2 to 5, and you submit the order. Order Management will submit this update and overwrite the revision 2 that it received from EastSystem. The submitted order will contain a quantity of 7 on line 1 and a quantity of 5 on line 2. Order Management won't create a third revision.

Closed Order Lines

You can import an order line that's already closed, but you need to modify the orchestration process so it doesn't attempt to process the closed line.

Assume you need to import all of your closed order lines.

1. Make a duplicate of the predefined DOO_BillOnlyGenericProcess orchestration process.

   For details, see *Set Up Orchestration Processes*.
2. Change the name of your duplicate to Immediately Close Imported Sales Orders.
3. Revise the line-selection criteria on the Create Invoice step.
   - Click **Click for Rule** In the Line Selection Criteria column on the Create Invoice step.
   - In the Line-Selection Criteria Set dialog, in the If area, change the first condition to.

     ```
     If 1 is 2
     ```

     1 will never equal 2, so this condition instructs the orchestration process to skip invoicing for the order line and immediately close the order line.
   - Delete all the other conditions in the If area.
4. Release and deploy your new orchestration process.
5. Use the Manage Process Assignment Rules for Sales Orders task to create an assignment rule that assigns your new orchestration process according to your requirement. Assume you use the status value Closed in your import payload to indicate a closed line. Here's your rule.

   ```
   If Status Code (Order Fulfill Line) is equal to Closed, then Process Name is set to Immediately Close
    Imported Sales Orders
   ```

   For details, see *Overview of Using Business Rules With Order Management*.
6. Import your sales orders.

   Your custom orchestration process will immediately close the sales orders that meet your criteria.

Configured Items

Note

- Your import must maintain a relationship between the parent configuration model and its child configure options.
- A configuration model can have one or more child configure options. A child configure option can also have one or more children.

Assume you have a model that contains four lines. Line 1 is the root parent and line 2 is a child of line 1. Line 2 is also a model and lines 3 and 4 are children of line 2. Here's what the hierarchy looks like.

```
Line 1
 Line 2
 Line 3
 Line4
```

where

- SourceTransactionLineId is the parent
- ParentSourceTransactionLineId identifies the parent in a child entity.

Here's a payload that correctly maintains the hierarchy.

```
"lines": [{
"SourceTransactionLineId": "1",
"SourceTransactionLineNumber": "1",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"OrderedUOMCode": "Ea",
"OrderedQuantity": 1,
"ProductNumber": "PTO54222",
"RequestedFulfillmentOrganizationId": 204
},
{
"SourceTransactionLineId": "2",
"SourceTransactionLineNumber": "2",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"OrderedUOMCode": "Ea",
"OrderedQuantity": 1,
"ProductNumber": "OP44136",
"RequestedFulfillmentOrganizationId": 204,
"ParentSourceTransactionLineId": "1"
},
{
"SourceTransactionLineId": "3",
"SourceTransactionLineNumber": "3",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"OrderedUOMCode": "Ea",
"OrderedQuantity": 1,
"ProductNumber": "KB18761",
"RequestedFulfillmentOrganizationId": 204,
"ParentSourceTransactionLineId": "2"
},
{
"SourceTransactionLineId": "4",
"SourceTransactionLineNumber": "4",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"OrderedUOMCode": "Ea",
```

```
"OrderedQuantity": 1,
"ProductNumber": "KB18759",
"RequestedFulfillmentOrganizationId": 204,
"ParentSourceTransactionLineId": "2"
}
]
```

where

- `"SourceTransactionLineId": "1"` specifies that line 1 is the root parent.
- `"ParentSourceTransactionLineId": "1"` in the SourceTransactionLineId 2 entity specifies that line 1 is the parent of line 2.
- `"ParentSourceTransactionLineId": "2"` in the SourceTransactionLineId 3 entity specifies that line 2 is the parent of line 3.
- `"ParentSourceTransactionLineId": "2"` in the SourceTransactionLineId 4 entity specifies that line 2 is the parent of line 4.

Now let's say you wanted to add a fifth line, and you use this code.

```
"SourceTransactionLineId": "5",
"SourceTransactionLineNumber": "5",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"OrderedUOMCode": "Ea",
"OrderedQuantity": 1,
"ProductNumber": "KB18760",
"RequestedFulfillmentOrganizationId": 204,
"ParentSourceTransactionLineId": "9"
}
]
```

This import would fail because ParentSourceTransactionLineId has a value of 9, but there's no SourceTransactionLineId in the payload that contains 9.

Shipment Sets or Kits

You must set the PartialShipAllowedFlag attribute to N on the line that contains the set or kit.

Coverages and Subscriptions

Make sure each coverage item in your import payload references a unique fulfillment line that contains the covered item you want to cover. If you split a fulfillment line, then each of these lines reference the same source line. Assume you split line 1 into lines 1-1 and 1-2. Lines 1-1 and 1-2 both reference the same source line, the source line isn't unique, so you can't use the source line to add coverage. The lines must be unique.

Here's part of an example payload that includes coverage.

```
<ns2:DocumentReference>
 <ns2:DocumentReferenceType>COVERAGE_COVERED_ASSOCIATION</ns2:DocumentReferenceType>
 <ns2:DocumentIdentifier>300100546398344</ns2:DocumentIdentifier>
 <ns2:DocumentSubLineIdentifier>300100546398346l</ns2:DocumentSubLineIdentifier>
 </ns2:DocumentReference>
<ns2:DocumentReference>
 <ns2:DocumentReferenceType>SOURCE_COVERAGE_COVERED_ASSOC</ns2:DocumentReferenceType>
 <ns2:DocumentIdentifier>jm_PDSC_PTO_IC_12_08_01</ns2:DocumentIdentifier>
 <ns2:DocumentAdditionalIdentifier>GPR</ns2:DocumentAdditionalIdentifier>
 <ns2:DocumentNumber>jm_PDSC_PTO_IC_12_08_01</ns2:DocumentNumber>
 <ns2:DocumentLineIdentifier>104</ns2:DocumentLineIdentifier>
 <ns2:DocumentAdditionalLineIdentifier>101</ns2:DocumentAdditionalLineIdentifier>
 <ns2:DocumentSubLineIdentifier>101</ns2:DocumentSubLineIdentifier>
```

```
</ns2:DocumentReference>
```

Inventories

Lots and Serials

If you use the Order Import Service web service, and if you want to import lot and serial details for an order line, then the LotSerial entity in your import payload must include a value in at least one of these attributes for the order line.

- LotNumber
- SerialNumberFrom
- SerialNumberTo
- ItemRevisionNumber
- Locator
- LocatorIdentifier

If you don't provide a value for any of these attributes in the LotSerial entity.

- The import won't create a record for the lot or serial in the DOO_LOT_SERIAL_NUMBERS table.
- And if you provide a value for the SourceTransactionLotIdentifier attribute, then the import will display an error.

Subinventories

If your import includes a value in the SubInventoryCode attribute, then you must make sure:

- The Requested Fulfillment Organization that you specify on the order line to fulfill the item contains the subinventory that you specify on the order line.
- The End Date for the subinventory happens in the future or is empty before you collect the subinventory.
- You collect the subinventory.

For example, assume you need to use the Finished Goods subinventory to fulfill the order line from the Vision Manufacturing organization. The Subinventory Code for Finished Goods Subinventory is FGI. Here's how you can make sure the import will work.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Manufacturing and Supply Chain Materials Management
    - Functional Area: Inventory Management
    - Task: Manage Subinventories and Locators
2. In the dialog that displays, select the Vision Manufacturing organization.
3. On the Manage Subinventories page, search for the values.

Attribute	Value
Subinventory	FGI If the search results don't contain the Finished Goods subinventory, then you need to add it, or use a different subinventory on the order line.
Description	Finished Goods

Attribute	Value

4. In the search results, verify the value.

Attribute	Value
End Date	Happens in the future or is empty. If the date happens in the past, then you need to change its value so its empty or happens in the future.

5. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Collect Order Reference Data
6. On the page that displays, click **Reference Data**.
7. Move Subinventories to the Selected Entities window, then click **Submit**.

    This will collect the Finished Goods subinventory so its available in the Order Orchestration and Planning Data Repository.

    Learn how to use this page. For details, see *Collect Planning Data for Order Management*.
8. Notice the value that displays in the dialog, then click **OK**.

    For this example, assume the value is 107610.
9. Go to the Scheduled Processes work area, then verify the values.

Attribute	Value
Name	Collection Job Set
Process ID	107610
Status	Succeeded

Prefix Line Numbers in Your Source Orders

The value that you import in the Source Order Line Number must be alphanumeric. You must not use a numeric value. We recommend that you prefix the value in the Source Order Line Number attribute with a text value, and that you use meaningful text. For example, use LEG to identify a legacy system, such as LEG_576849.

Display Line Numbers

Make sure your payload has all the data that Order Management needs to accurately display the fulfillment line number in the Order Management work area.

Order Management data stores the Display Line Number as a concatenation of the FulfillLineId attribute plus the FulfillLineNumber attribute, such as 300100562534985 1-1. The salesOrdersForOrderHub Rest API returns the fulfillment line in the same concatenated form that you see in the Order Management work area.

Next, we'll look at some examples. You must make sure your salesOrdersForOrderHub REST API payload represents the fulfillment lines just like the examples represent them.

Example of a Split Fulfillment Line

Assume you have an order line that you split into two fulfillment lines. Here's an example of how your salesOrdersForOrderHub REST API payload must represent these lines.

```
{
 "lines" : {
 "items" : [
 {
 "FulfillLineId" : 300100562534985,
 "DisplayLineNumber" : "1-1",
 "FulfillLineNumber" : "1"
 },
 {
 "FulfillLineId" : 300100562535469,
 "DisplayLineNumber" : "1-2",
 "FulfillLineNumber" : "2"
 }
 ]
 }
}
```

Notice that a single `items` entity contains one group of FulfillLineId, DisplayLineNumber, and FulfillLineNumber attributes for each split fulfillment line, and that DisplayLineNumber has the same value that you see in the Order Management work area, such as 1-1.

Notice that a single `OrderLine` entity contains one DisplayLineNumber entity, and the DisplayLineNumber entity contains a group of FulfillLineId and FulfillLineNumber attributes for each split fulfillment line, but DisplayLineNumber doesn't have the 1-1 value.

Example of a Split Fulfillment Line with a Coverage

Assume you have an order line that you split into two fulfillment lines, and the line contains a coverage item. Here's an example of how your salesOrdersForOrderHub REST API payload must represent these lines.

```
{
 "lines": {
 "items": [
 {
 "FulfillLineId": 300100565241163,
 "DisplayLineNumber": "1:1-1",
 "FulfillLineNumber": "1"
 },
 {
 "FulfillLineId": 300100565241034,
 "DisplayLineNumber": "1:1-2",
 "FulfillLineNumber": "2"
 }
 ]
 }
```

```
}
```

For example, this payload sets the DisplayLineNumber attribute for fulfillment line 300100565241034 to `1:1-2`. The Order Management work area also displays the `1:1-2` value.

- The 1 before the colon represents the DisplayLineNumber of the covered line.
- The 1 after the colon represents the sequence number of the coverage. For example, `1:1` means you have another coverage for the same covered line, where 1 is the sequence number of the coverage.
- The dash means that Order Management split the order line into more than one fulfillment line. In this example, fulfillment line 1 and fulfillment line 2.
- The 2 after the dash represents the second split line, fulfillment line 2.

Example of a Split Fulfillment Line with a Configured Item

You use the DisplayLineNumber to represent a configured item's hierarchy. Here's an example.

Item	DisplayLineNumber
AS54888 Desktop Computer	1
Storage Option Class	1.1
1 Terabyte Hard Drive	1.1.1
2 Terabyte Hard Drive	1.1.2
Port Option Class	1.2
USB 3.0 Port	1.2.1
USB 2.0 Port	1.2.2

Note

- The DisplayLineNumber for the AS54888 root item is 1, so all child items of AS54888 will also start with 1.
- The DisplayLineNumber for the Storage Option Class item is 1.1, so all child items of Storage Option Class will also start with 1.1.
- The root item might not always be 1. For example, if you manually delete a line that has the AS54888, then add a new line with the AS54888 in a draft order, then Order Management changes the DisplayLineNumber for the AS54888 from 1 to 2, and it changes all child items of AS54888 so they also start with 2.

Assume you have an order line that includes a quantity of two for the 1 Terabyte Hard Drive item. There's only enough supply to fulfill a quantity of 1, so Order Management splits that order line into two fulfillment lines. Here's an example of how your salesOrdersForOrderHub REST API payload must represent these lines.

```
{
 "lines": {
 "items": [
 {
 "FulfillLineId": 300100563721562,
 "DisplayLineNumber": "1.1.1-1",
 "FulfillLineNumber": "1"
 },
 {
```

```
"FulfillLineId": 300100563723289,
"DisplayLineNumber": "1.1.1-2",
"FulfillLineNumber": "2"
}
]
}
}
```

This payload sets the DisplayLineNumber attribute for fulfillment line 300100563721562 to `1.1.1-1`. The Order Management work area also displays the `1.1.1-1` value.

- The first 1 represents the root line for the configured item.
- The second 1 represents the Storage option class.
- The 1 before the dash represents the 1 Terabyte Hard Drive configure option.
- The dash means Order Management split the fulfillment line.
- The 1 after the dash represents the first fulfillment line for the 1 Terabyte Hard Drive configure option.

Original Order Reference

There are specific guidelines for using the originalOrderReference entity. For details, see *Guidelines for Processing Return Orders*.

Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- Overview of Using Business Rules With Order Management
- How Order Management Transforms Source Orders Into Sales Orders
- Data Security

Export and Import Setup Data with Inventory Organizations

If you use an inventory organization when you set up Order Management, then you must import the setup data of the offerings that include your inventory organizations, and then import the setup data from Order Management to your implementation instance.

You do the import when you deploy your setup data from one application instance to another application instance.

Select an option when you export or import your setup data.

Option	Description
Create one configuration package.	This package contains the offerings that include the inventory organizations, and that also contains the Order Management offering.
Create more than one configuration package.	Use a separate configuration package for the offering that includes the inventory organizations, and use another package for the Order Management offering.

Import Customer Items Into Order Management

Import customer items into Order Management so you can use them throughout the order lifecycle.

- Use file-based data import, a web service, or REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click **Sales Orders for Order Hub**.
- You can't automatically determine the customer item during the import. Your import payload must include a value for the customer item.

Scenario

In this example, assume you must.

- Import a record that includes customer item DESK-COMP-10 and customer Computer Service and Rentals.
- Create a relationship between DESK-COMP-10 and inventory item AS54888 (Standard Desktop).
- Create a relationship between DESK-COMP-10 and inventory item AS54600 (Standard 9000-S Desktop).
- Rank AS54888 higher than AS54600 so Order Management uses AS54888 first to fulfill the item, then uses AS54600 only if AS54888 isn't available in inventory.

You can create a relationship between the AS54888 and DESK-COMP-10 when you import.

- The customer item relationship sets up a relationship between the customer item that you import, and the item in inventory that you use to fulfill the customer item.
- Order Management uses the customer item and sold-to customer details to identify the inventory item to use during order import.
- If you set up more than one customer item relationship for a customer item, then Order Management uses the customer item relationship that has the lowest rank.

Order import will fail in these situations.

- You don't set up a customer item relationship for the customer item in the Product Information Management work area.
- More than one customer item relationship specifies the lowest rank.

Summary of the Setup

1. Set up customer item relationships.
2. Import the customer item.

Set Up Customer Item Relationships

Set up customer item relationships so they support order import.

1. Make sure you have the privileges that you need to administer Product Information Management.
   If you don't sign in with these privileges, then the Product Information Management work area won't display your product details and you won't be able to do this procedure. For details, see *Privileges That You Need to Implement Order Management*.
2. Go to the Product Information Management work area, then click **Tasks > Manage Trading Partner Items**.
3. On the Manage Trading Partner Items page, click **Create Trading Partner Item** (the plus icon).
4. In the Create Trading Partner area, set the values.

Attribute	Value
Trading Partner Item	DESK-COMP-10

Attribute	Value
Type	Customer
Trading Partner	Computer Service and Rentals
Status	Active
Start Date End Date	Set the time frame when you want Order Management to use this partner item. The import validates these dates. If the dates in your import don't match the dates that you set on the Create Trading Partner Items page, then the import will fail.

5. In the Relationships area, click **Actions > Create**.
6. In the Create Customer Item Relationship dialog, set the values, then click **OK > Save**.

Attribute	Value
Organization	Vision Manufacturing Set it to the inventory organization that maintains inventory for the AS54888.
Item	AS54888
Rank	1 Enter any numeric value. o Enter a value. If you don't enter a value, then Order Management can't identify the inventory item to use during order import, and your import will fail. o Make sure the value that you enter is unique across the customer item relationships that you set up for this customer item. For example, if you set up a rank of 5 for AS54888, and a rank of 5 for AS54600, then the import will fail because Order Management can't determine which customer item relationship takes precedence.

7. In the Relationships area, click **Actions > Create**.
8. In the Create Customer Item Relationship dialog, set the values, then click **OK**.

Attribute	Value
Organization	Vision Manufacturing

Attribute	Value
Item	AS54600
Rank	2

9. Click **Save > Done**.

As an option, you can use file-based data import to import the relationship instead. If you do, set the values in the EGP_ITEM_RELATIONSHIPS_INTF worksheet when you import inventory items.

Attribute	Value
Item Relationship Type	CUSTOMER_ITEM_XREF
Item Number	AS54888
Trading Partner Item Number	DESK-COMP-10
Description	Customer's part number for the AS54888.

Note: Make sure you include a value in all other required attributes.

Import the Customer Item

If you're importing source orders through file-based data import, set the values on the DOO_ORDER_LINES_ALL_INT worksheet of the PriceListImportTemplate.xlsm file.

Attribute	Value
Product Number	AS54888
Customer Product Identifier	DESK-COMP-10

If you're importing through a web service or REST API, set the values in the payload.

Attribute	Value
ProductNumber	AS54888
CustomerItemId	DESK-COMP-10

Related Topics

- Customer Items in Order Management
- Import Orders Into Order Management
- Privileges That You Need to Implement Order Management
- Set Up Customer Items for Order Management

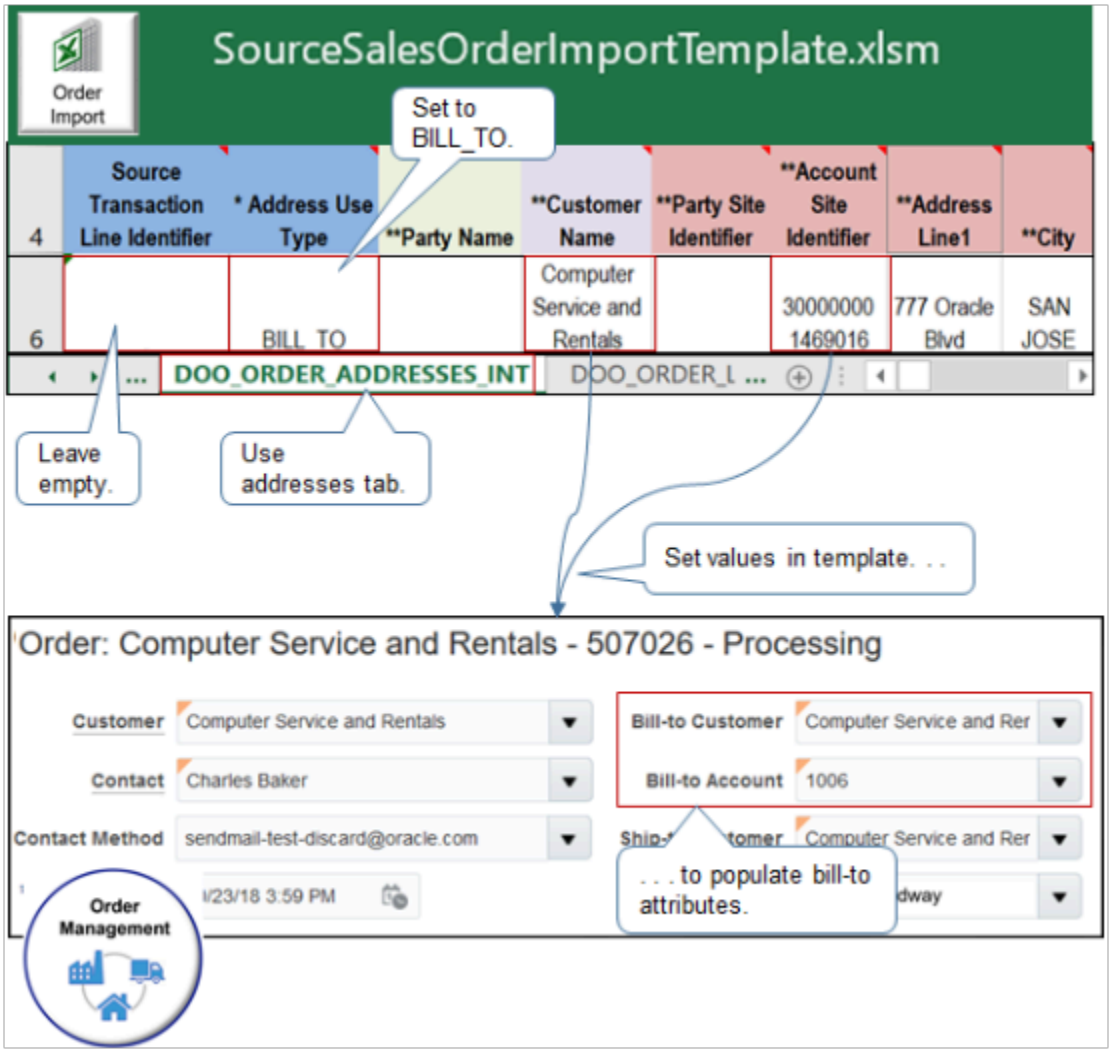
Import Addresses into Order Management

Use ship-to details and bill-to details to populate order header attributes and order line attributes when you use file-based data import.

Import Order Header Attributes

Populate bill-to attributes on the order header.

- Bill-to Customer
- Bill-to Account

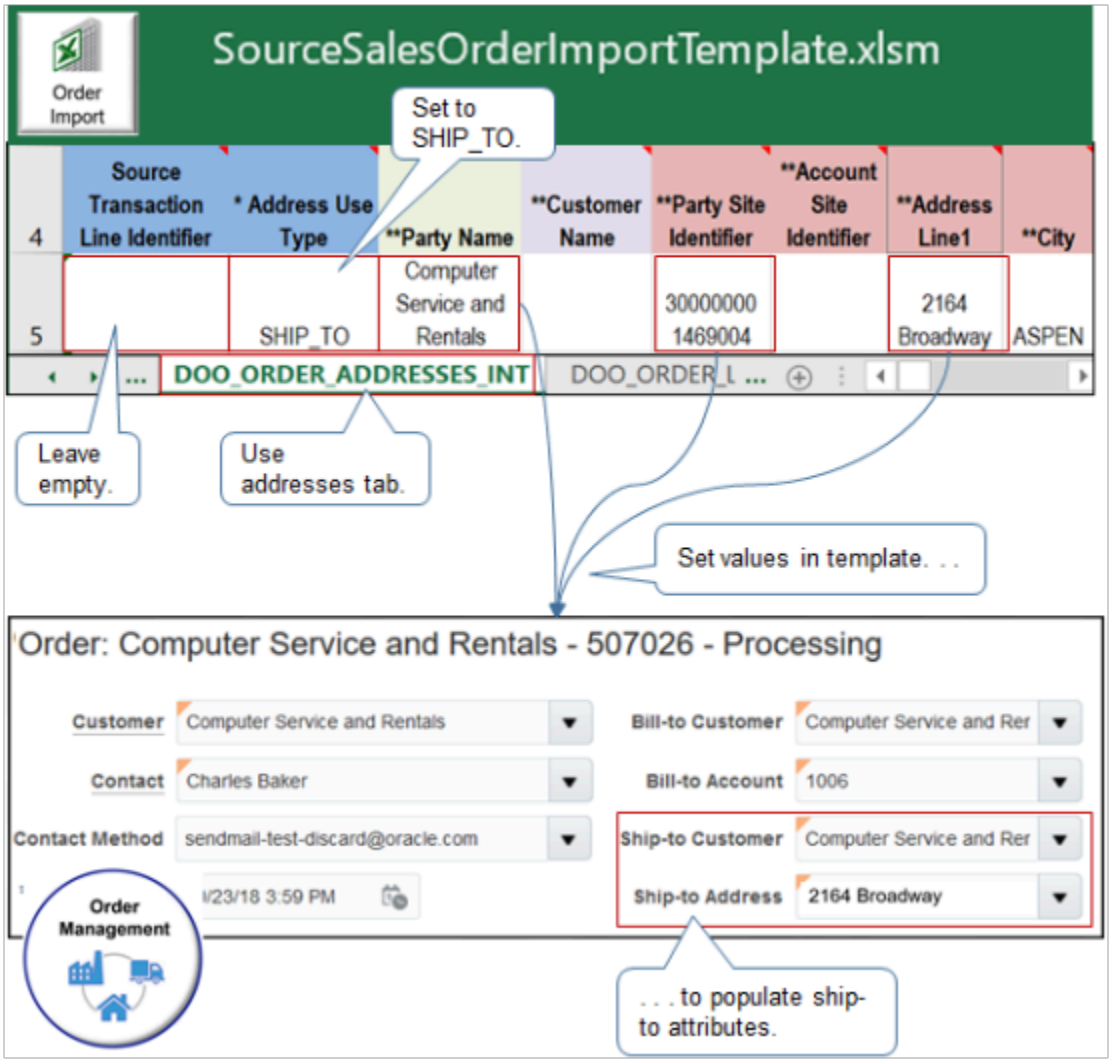


Use attributes on the DOO_ORDER_ADDRESSES_INT tab.

Attribute	Value
Address Use Type	BILL_TO
Customer Name	Name of customer you're importing, such as Computer Service and Rentals. Use this attribute only when Address Use Type contains BILL_TO.
Account Site Identifier	Value that uniquely identifies the account, such as 300000001469016. Use this attribute only when Address Use Type contains BILL_TO.
Source Transaction Line Identifier	Leave empty.

Populate ship-to attributes on the order header.

- Ship-to Customer
- Ship-to Address



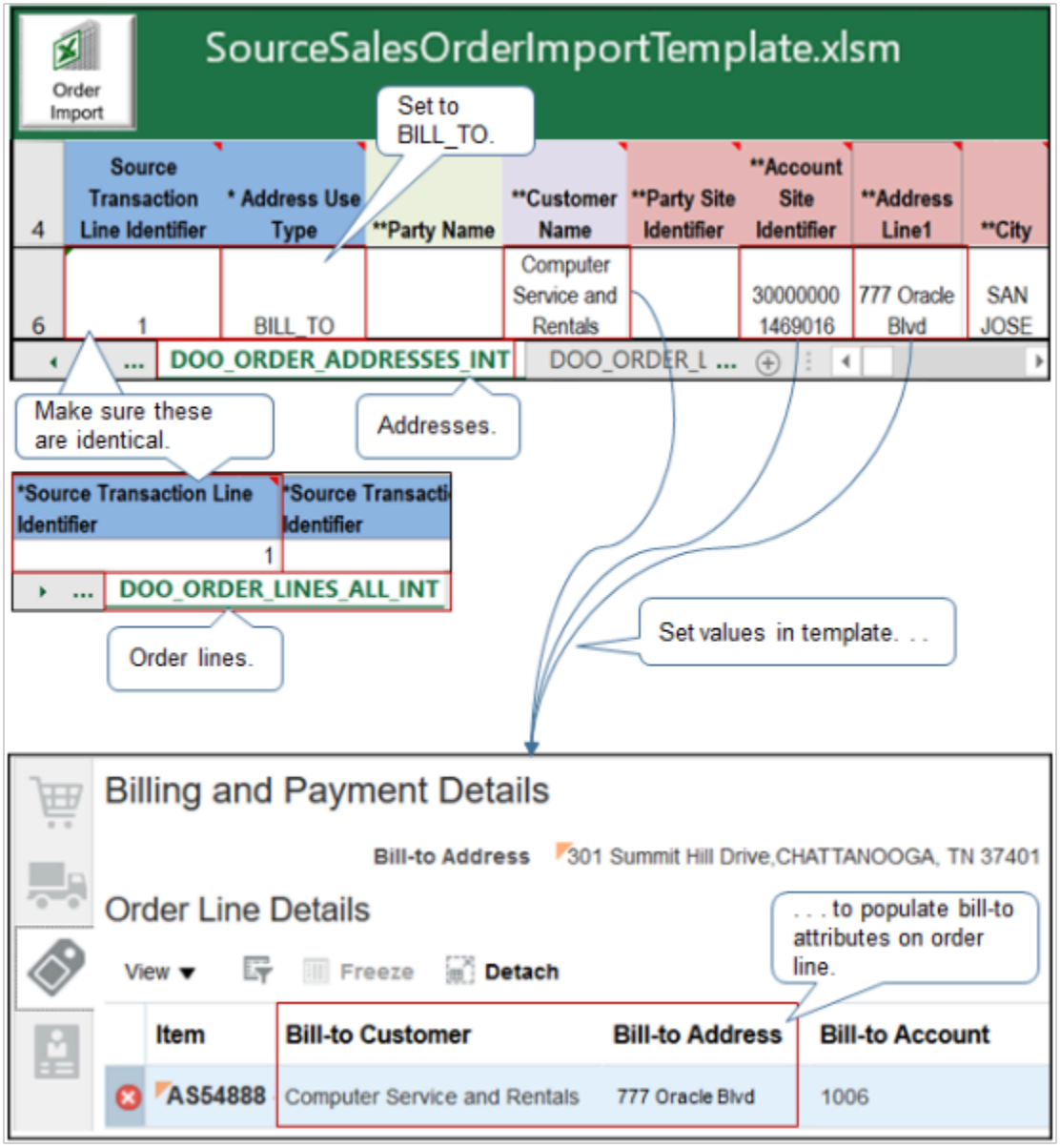
Use attributes on the DOO_ORDER_ADDRESSES_INT tab.

Attribute	Value
Address Use Type	SHIP_TO
Party Name	Name of the party you're importing, such as Computer Service and Rentals. Use this attribute only when Address Use Type contains SHIP_TO.
Party Site Identifier	Value that uniquely identifies the party site address, such as 300000001469004. Use this attribute only when Address Use Type contains SHIP_TO.
Source Transaction Line Identifier	Leave empty.

Import Order Line Attributes

Populate Bill-To Attributes on the Order Line

- Bill-to Customer
- Bill-to Address



Set the same values that you use to populate order header attributes, except set this attribute.

Attribute	Description
Source Transaction Line Identifier	Value that uniquely identifies the transaction line in the source system, such as 1.

Note

- Make sure you set Source Transaction Line Identifier on DOO_ORDER_LINES_ALL_INT and on DOO_ORDER_ADDRESSES_INT to the same value.
- The import uses Source Transaction Line Identifier to create a relationship between the address on DOO_ORDER_ADDRESSES_INT with the order line on DOO_ORDER_LINES_ALL_INT.

Set a value for Source Transaction Line Identifier only if your Ship-to Address on the order line must be different than the Ship-to Address on the order header.

If the Ship-to Address is the same for the order header and all order lines, then leave these attributes empty on the DOO_ORDER_ADDRESSES_INT tab.

- Source Transaction Line Identifier
- Source Transaction Schedule Identifier

Set the Address Use Type

If you set the Address Use Type attribute to BILL_TO or SHIP_TO, then the template uses these values to determine which row to use when it populates the Bill_to and Ship_to attributes on the order header. For example, if you set Address Use Type to SHIP_TO on row 5 of DOO_ORDER_ADDRESSES_INT, and if the Address Line1 attribute on row 5 contains 123 Main Street, then the import sets the Ship-to Address address on the order header to 123 Main Street, and it cascades this value from the order header to the Ship-to Address on all order lines.

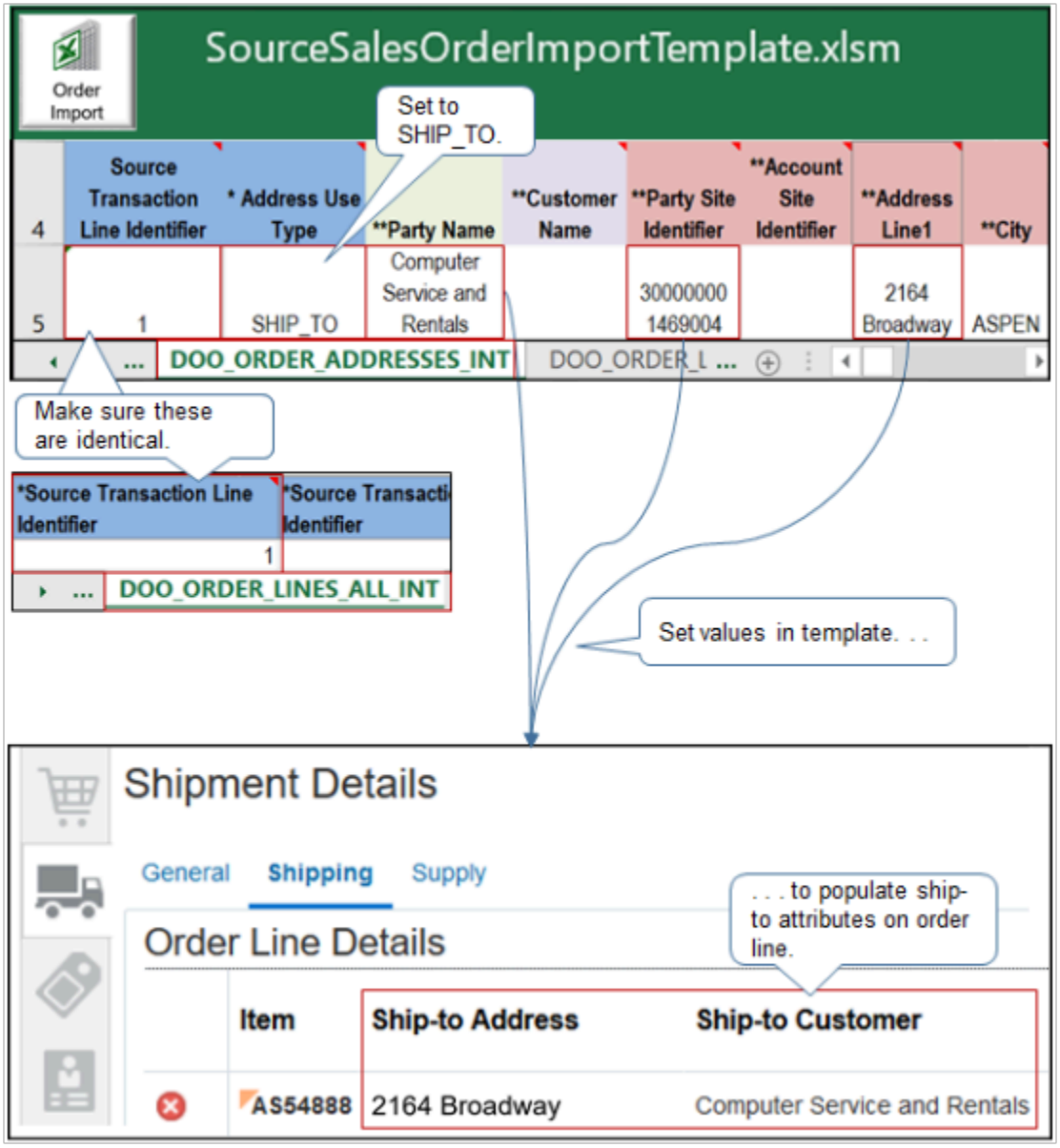
If you import a sales order and find that attributes on the order header are empty, its possible that you set the values for Source Transaction Line Identifier and Source Transaction Schedule Identifier on DOO_ORDER_ADDRESSES_INT but not for Address Use Type and other attributes that the import uses to populate values on the order header, such as Party Site Identifier and Address Line1.

You can also set Address Use Type to:

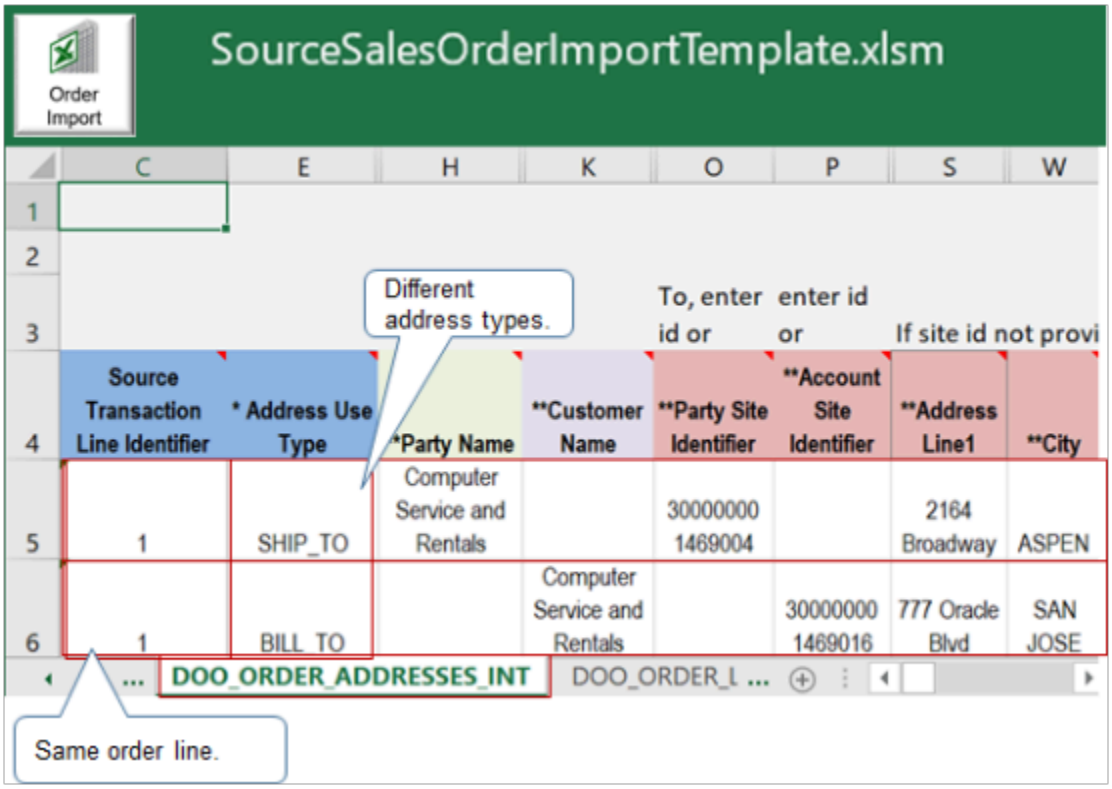
- DESTINATION_SHIPPING_TO. If you use this value, then the template uses the value in the Location Identifier attribute on the DOO_ORDER_ADDRESSES_INT worksheet to determine the shipping destination to use.
- FINAL_DISCHARGE_TO. If you use this value, then the template uses the value in the Location Identifier attribute on the DOO_ORDER_ADDRESSES_INT worksheet to determine the discharge location to use. Use this value to specify the location that you want to use when you calculate tax for the item on the invoice. For details, see the Manage Location of Final Discharge section in Oracle Financials Cloud, Implementing Tax.

Populate Ship-To Attributes on the Order Line

- Ship-to Customer
- Ship-to Address



Set Bill-to and Ship-to On the Same Order Line



Get Values for Identifier Attributes

Make sure you use the correct value for the Account Site Identifier attribute and the Party Site Identifier attribute. Use SQL to query the Order Management database to get these values.

Here's the SQL to run.

```
SELECT hzp.party_name
 ||
 ' '
 ||
 hzp.party_number ,
 hzp.party_id ,
 HZA.account_number ,
 HZA.account_name ,
 hza.CUST_ACCOUNT_ID ,
 HZA.status "Account Status" ,
 hzp.status "Party Status" ,
 hzps.status "Party Site Status" ,
 hzps.party_site_id "PARTY SITE ID - for SHIP_TO",
 hzcasa.status "Account Site Status" ,
 hzcsua.site_use_id "SITE USE ID - for BILL_TO" ,
 hzcasa.start_Date ,
 hzcasa.end_Date ,
 hzcsua.SITE_USE_CODE ,
 hzcasa.BILL_TO_FLAG ,
 hzcasa.SHIP_TO_FLAG ,
 hzcsua.PRIMARY_FLAG ,
 hzcsua.STATUS "Account Site USE Status" ,
 hzcsua.LOCATION ,
 hzl.ADDRESS1 ,
 hzl.ADDRESS2 ,
 hzl.ADDRESS3 ,
 hzl.ADDRESS4 ,
```

```
 hzl.CITY ,
 hzl.POSTAL_CODE ,
 hzl.STATE ,
 hzl.COUNTRY
FROM fusion.HZ_PARTIES HZP ,
 fusion.HZ_PARTY_SITES hzps ,
 fusion.HZ_CUST_ACCOUNTS HZA ,
 fusion.HZ_CUST_ACCT_SITES_ALL hzcasa,
 fusion.HZ_CUST_SITE_USES_ALL hzcsua ,
 fusion.hz_locations HZL
WHERE hzP.party_id = HZA.party_id (+)
 AND hza.CUST_ACCOUNT_ID = hzcasa.CUST_ACCOUNT_ID (+)
 AND hzcasa.party_site_id = hzps.party_site_id (+)
 AND hzcasa.cust_acct_site_id = hzcsua.cust_acct_site_id (+)
 AND hzps.location_id = hzl.location_id (+)
 -- and (
 -- hzcasa.start_Date <= sysdate
 -- OR hzcasa.start_Date IS NULL
 -- )
 -- AND
 -- (
 -- hzcasa.end_date >= sysdate
 -- OR hzcasa.end_date IS NULL
 -- )
 AND upper(hzP.party_name) LIKE upper('%GOO%IND%')
ORDER BY hzp.party_number ,
 hza.account_number,
 hzl.LOCATION_ID
```

Note

- Examine the SITE_USE_CODE column.
- For ship_to, use the value that `PARTY SITE ID - for SHIP_TO` returns.
- For bill_to, use the value that `SITE USE ID - for BILL_TO` returns.
- This SQL also returns inactive account details.

In your SQL, replace the placeholder with a value from your order import template.

Placeholder	Attribute on the DOO_ORDER_ADDRESSES_INT Tab
`party_name`	Party Name
`party_number`	Party Number
`party_id`	Party Identifier
`account_number`	Not applicable
`account_name`	Not applicable

Related Topics

- Use SQL to Query Order Management Data
- Overview of Importing Orders Into Order Management
- Roadmap for Setting Up Order-to-Cash
- How Order-to-Cash Works with Order Capture Systems

Use REST API to Update and Split Fulfillment Lines

Use the Sales Orders for Order Hub REST API to update or split fulfillment lines across sales orders without having to revise them.

You can update scheduling attributes, substitute items, or remove fulfillment lines from a shipment set. Oracle Order Management will then send your changes and split lines to downstream fulfillment systems.

Assume only some of the item's quantity on the order line is available on the requested date in the warehouse. The remaining quantity is available on a different date or in another warehouse, or a substitute item is available. You have different ways to reduce the delay when shipping the item to your customer. For example, you can:

- Split the line into lines *x* and *y*, ship the available quantity on the requested date on line *x*, and then ship the remaining quantity on the next available date on line *y*.
- Split the line into lines *x* and *y*, ship the available quantity from the warehouse on line *x*, and then ship the remaining quantity from another warehouse on the requested date on line *y*.
- Ship a substitute item.

You can use the UpdateSchedulingAttributeRequest operation in the Sales Orders for Order Hub REST API to update fulfillment lines across sales orders, and you can use the SplitFulfillmentLineRequest operation to split fulfillment lines across sales orders.

If you split or update scheduling attributes on the lines, then Order Management will compensate the fulfillment tasks, and then send the updated and split lines to your downstream fulfillment system.

Update

You can use UpdateSchedulingAttributeRequest to update these attributes on the fulfillment line:

- Scheduled Ship Date
- Scheduled Arrival date
- Warehouse
- Supplier
- Supplier Site
- Shipping Method, including carrier, mode of transport, and service level
- Demand Class
- Item
- Shipment Set

Try It

Assume you have these fulfillment lines.

Sales Order	Order Line	Fulfillment Line	Item	Original Item	Quantity	Warehouse	Requested Ship Date	Scheduled Ship Date	Shipment Set	Status	Override Schedule
530866	1	1-1	AS54888		10	M1	7/17/23	7/31/23		Awaiting Shipping	No
530866	2	2-1	OM-RT-Standard-Item1		12	M1	7/17/23	7/31/23	SS1	Awaiting Shipping	No
530866	3	3-1	OM-RT-Standard-Item4		16	M1	7/17/23	7/31/23	SS1	Awaiting Shipping	No
530868	1	1-1	AS92888		11	M1	7/17/23	7/31/23		Awaiting Shipping	No

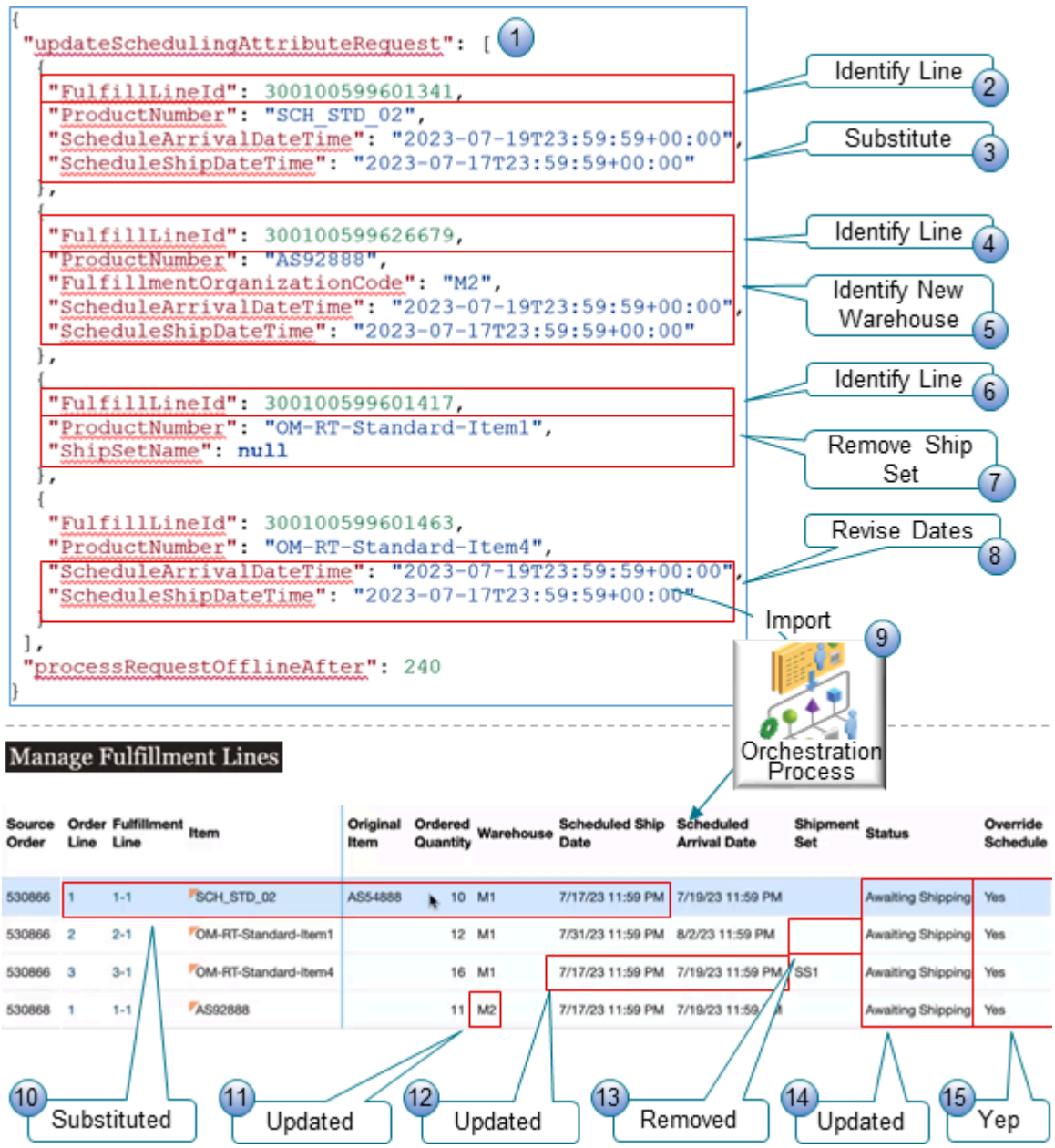
Note:

- All fulfillment lines have a Requested Ship Date of 7/17/23 but the Scheduled Ship Date is 7/31/23, so all lines are behind schedule. The Status is Awaiting Shipping.
- Sales order 530866 has three lines and order 530868 has one line.
- Lines 2-1 and 3-1 are part of a shipment set.
- All lines ship from the M1 warehouse.

To keep your customers happy, you want to:

- Substitute the AS54888 with another item that you know is ready to ship from the M1 warehouse, which is the SCH_STD_02 item.
- Remove the item on fulfillment line 2-1 from the shipment set so you can ship the item on fulfillment line 3-1 separately. Update the schedule dates of fulfillment line 3-1 to ship on the requested date.
- Change the warehouse from M1 to M2 for the AS92888 because M2's inventory is ready to ship.

Here's a payload that you can use to do this:



Notes

1. Use the updateSchedulingAttributeRequest operation to specify your changes. You can specify this operation anywhere in the Order entity in your payload.
2. Set FulfillLineID to the original fulfillment line that contains the AS54888.
3. Set ProductNumber to the SCH_STD_02 substitute item on the original fulfillment line, then set ScheduleArrivalDateTime and ScheduleShipDateTime to the requested date.
4. Set FulfillLineID to the original fulfillment line that contains the AS29888.
5. Set FulfillmentOrganizationCode to M2 and set ScheduleArrivalDateTime and ScheduleShipDateTime to the requested date.
6. Set FulfillLineID to the original fulfillment line that contains the OM-RT-Standard-Item1 item.
7. Set ShipSetName to `null` to indicate that you want to remove the OM-RT-Standard-Item1 from the ship set.

8. Update ScheduleArrivalDateTime and ScheduleShipDateTime on fulfillment line 3-1 to ship the OM-RT-Standard-Item4 item on the requested date.
9. Import your payload, then verify that you receive a Success status in the REST API response.

   The orchestration process on the fulfillment line uses the attribute that you updated to identify change for a fulfillment task, compensates the fulfillment task, and then sends the updated attribute values on the fulfillment lines to the fulfillment system. For example, if you change a scheduled date, substitute the item, or remove a fulfillment line from a shipment set, then Order Management compensates the Schedule, Reservation, and Shipping tasks, and then sends the changes to Oracle Global Order Promising and Oracle Inventory Management.

   Next, go to the Order Management work area, open the sales order in a fulfillment view, then verify your changes.

   The sales order's status is Processing. You know you didn't create a sales order revision because there's no revision number. You only updated the existing version.
10. The Item attribute on order line 1 contains the substitute SCH_STD_02 item, the Original Item attribute contains the AS54888 item, and the Scheduled Ship Date contains the updated requested date.
11. Line 1 for sales order 530868 has the M2 warehouse for your AS92888 item with the revised schedule date.
12. Line 3 for sales order 530866 has the revised dates that you set for the OM-RT-Standard-Item4.
13. The Shipment Set attribute is empty on line 2. This means you successfully removed OM-RT-Standard-Item1 from the shipment set.
14. The Status attribute contains Awaiting Shipping. All items except the item on line 2-1 are ready to ship on the requested ship date instead of after the requested ship date.
15. The Override Schedule attribute contains Yes. You can use this attribute to quickly determine whether you have overridden the original values from the fulfillment line.

Click *here* to see a nifty demonstration of this set up.

For details about REST API, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Sales Orders for Order Hub**.

Guidelines for Updating

- You must include the fulfillment lines and an updated value for at least one of the scheduling attributes on each line.
- As an option, you can include a reason when you update a scheduling attribute or substitute an item.
- If the item has a coverage, and if you want to substitute the covered item, then the substitute item's Enable Coverage Contract attribute in the Product Information Management work area must equal Yes.

Configured Items, Kits, and Shipment Sets

- You must include all of the items that are part of a configured item, kit, or shipment set, and you must use the same values in the scheduling attributes on each line that's part of the configured item, kit, or shipment set.
- If you want to remove a fulfillment line from a shipment set, then include an empty value such as " ", or include the value `null` in the ShipSetName attribute in the fulfillment line entity.
- You don't need to unschedule a fulfillment line to remove it from a shipment set. For example, you can remove a line that's in the Awaiting Shipping status from a shipment set. However, if you use the Remove from Shipment Set action in a fulfillment view in the Order Management work area, then you must continue to unschedule the fulfillment line before you remove it from the shipment set.
- You can't change the shipment set on a fulfillment line to another value. You can only change it to an empty value.

- You can update scheduling attributes when you remove a fulfillment line from a shipment set. For example, you can update the scheduled ship date, scheduled arrival date, warehouse, shipping method, and so on.

Substitutions

- If you want to substitute an item on the line, then specify the substitute item in one of these attributes: InventoryItemID, ProductNumber, or ProductDescription.
- You can update scheduling attributes when you substitute an item. For example, you can update the scheduled ship date, scheduled arrival date, warehouse, shipping method, and so on.
- You can substitute only a standard item. You can't substitute a configured item, kit, or the child item of a configured item or kit.
- If the line is in a shipment set, then you can't substitute the item on the fulfillment line. Instead, you can remove the fulfillment line from the shipment set, and then substitute the item.

Split

You can also use REST API to split a fulfillment line. Realize these benefits when you split a line:

- Provide more flexibility in how you split a fulfillment line or update scheduling attributes on the line.
- Improve efficiency and performance when you split a fulfillment line or update scheduling attributes on the line.
- Improve usability and efficiency when you need to split or update fulfillment lines in more than one sales order.
- Avoid revising sales orders.

Assume you have these fulfillment lines.

Sales Order	Order Line	Fulfillment Line	Item	Original Item	Quantity	Warehouse	Requested Ship Date	Scheduled Ship Date	Shipment Set	Status	Override Schedule
530883	1	1-1	OM-RT-Standard-Item3		15	M1	7/17/23	7/31/23		Awaiting Shipping	No
530885	2	1-1	OM-RT-Standard-Item6		18	M1	7/17/23	7/31/23		Awaiting Shipping	No

Note:

- All lines ship from the M1 warehouse.
- Each line is from a different sales order.
- M1's Requested Ship Date is 7/17/23 on all lines, but the Scheduled Ship Date is 7/31/23, which is after the requested date.

All lines are behind schedule. M1 only has a quantity of 9 for the OM-RT-Standard-Item3, while M2 has enough inventory to fulfill the remaining quantity. To keep your customers happy, you want to:

- Split the line that has the OM-RT-Standard-Item3 into two lines. You want to ship one line with a quantity of 9 from M1 and another line with the remaining quantity of 6 from M2.
- Split the line that has the OM-RT-Standard-Item6 into three lines so you can fulfillment them on different dates.

Here's an example payload that you could use to split the line.

```
{
 "processRequestOfflineAfter": 240,
 "splitFulfillmentLineRequest": [
 {
 "FulfillLineId": 300100599645013,
 "ProductNumber": "OM-RT-Standard-Item3",
 "OrderedQty": "9",
 "FulfillmentOrganizationCode": "M1",
 "ScheduleArrivalDateTime": "2023-07-19T23:59:59+00:00",
 "ScheduleShipDateTime": "2023-07-17T23:59:59+00:00"
 },
 {
 "SplitFromFlineId": 300100599645013,
 "FulfillInstanceId": 1,
 "ProductNumber": "OM-RT-Standard-Item3",
 "OrderedQty": "6",
 "FulfillmentOrganizationCode": "M2",
 "ScheduleArrivalDateTime": "2023-07-19T23:59:59+00:00",
 "ScheduleShipDateTime": "2023-07-17T23:59:59+00:00"
 },
 {
 "FulfillLineId": 300100599645173,
 "ProductNumber": "OM-RT-Standard-Item6",
 "OrderedQty": "3",
 "FulfillmentOrganizationCode": "M1",
 "ScheduleArrivalDateTime": "2023-07-19T23:59:59+00:00",
 "ScheduleShipDateTime": "2023-07-17T23:59:59+00:00"
 },
 {
 "SplitFromFlineId": 300100599645173,
 "FulfillInstanceId": 1,
 "ProductNumber": "OM-RT-Standard-Item6",
 "OrderedQty": "7",
 "FulfillmentOrganizationCode": "M1",
 "ScheduleArrivalDateTime": "2023-07-26T23:59:59+00:00",
 "ScheduleShipDateTime": "2023-07-24T23:59:59+00:00"
 },
 {
 "SplitFromFlineId": 300100599645173,
 "FulfillInstanceId": 2,
 "ProductNumber": "OM-RT-Standard-Item6",
 "OrderedQty": "8",
 "FulfillmentOrganizationCode": "M1",
 "ScheduleArrivalDateTime": "2023-08-02T23:59:59+00:00",
 "ScheduleShipDateTime": "2023-07-31T23:59:59+00:00"
 }
 ]
}
```

For details about this example, including the payload and results in the Order Management work area, click *here*, then go to 3:52 in the demonstration.

Guidelines for Splitting

- You must include the reduced quantity on the original fulfillment line and the remaining quantity on the split lines. The sum of these quantities must equal the original quantity. Assume line *x* has a quantity of 10 and you split *x* into line *x* and line *y*. The sum of the quantities on *x* and *y* must equal 10.
- You can split a fulfillment line into more than two lines. For example, you can split a fulfillment line into four lines, then assign a different warehouse on each of those four lines.

- The values in the scheduling attributes on the split lines don't have to match the values on the original line.
- You can update the scheduling attributes on the original fulfillment line when you split it. For example, you can include updated values for the scheduled date, warehouse, supplier, and so on.
- If you use the splitFulfillmentLineRequest operation in REST API, then you can split the line even if it isn't on a manual step. For example, you can split a line that's in the Awaiting Shipping status. However, if you use the Split action on a fulfillment view in the Order Management work area, then you must make sure the fulfillment line is on a manual step. If you use the Check Availability action on a fulfillment view, then you must continue to unschedule the fulfillment line from the fulfillment view before you split it in the Check Availability dialog.
- You can't use a single request to split a fulfillment line and substitute an item on the original line at the same time. Instead, send one request that substitutes the item on the original line, then send another request that splits the line.
- If you create your own task and use a hold service on that task, and if your fulfillment line is on your task's wait step, then you can't split the fulfillment line.
- You can't use splitFulfillmentLineRequest to split a fulfillment line that includes an inventory transaction.
- If you split a fulfillment line, and if the fulfillment tolerance on the line isn't 0, and you ship the split line, then Oracle Shipping won't consider the cumulative quantity that you already shipped across all of the order line's fulfillment lines. Instead, shipping will consider only the quantity on the split line that Shipping is currently shipping.

Configured Items, Kits, and Shipment Sets

- You must include all of the items that are part of a configured item or kit, and you must use the same values in the scheduling attributes on each line that's part of the configured item or kit.
- You can't split a fulfillment line that's in a shipment set. Instead, you can remove the fulfillment line from the shipment set and then split it.
- If you split a configured item or a kit, then the split must be in proportion. For details, see *How Configure-to-Order Works*.

Other Considerations

You can't use REST API's splitFulfillmentLineRequest or updateSchedulingAttributeRequest on:

- A return line, coverage, or subscription.
- An item that you don't ship.
- A fulfillment line that's already undergoing change or that isn't on an active wait step.
- A fulfillment line that Order Management created while partially fulfilling a drop shipment.
- A remnant line of a configured item. For more about remnants, see *Fulfill Configured Items*.
- A sales order that you submitted before you opted into the Update Attributes on Split Order Lines for Partial Shipments feature.

Notes

- If the update or split fails for a fulfillment line, then Order Management won't update or split the line and instead return an error status. If your request includes more than one fulfillment line, then the update or split might be successful for some lines but fail for the other lines. Order Management returns a success status for the successful lines and an error status for the failed lines. If the failed line is part of a configured item, kit, or shipment set, then Order Management won't update or split any of the items that are part of the configured item, kit, or shipment set, and instead return an error on all the lines.

- updateSchedulingAttributeRequest sets the Override Schedule attribute to Yes on the fulfillment line that you update, and splitFulfillmentLineRequest sets it to Yes on the original fulfillment line and on the split lines. Order Management won't create a new revision for the sales order when you use updateSchedulingAttributeRequest or splitFulfillmentLineRequest.
- You can use the processRequestOfflineAfter operation to manage your REST request to split or update a fulfillment line. It works the same when you use REST API to split or update a fulfillment line as it does with holds. For details about using it, see *Use REST API to Apply and Release Holds*.
- If you use updateSchedulingAttributeRequest on a fulfillment line that's in the Manual Scheduling Required status because you set up scheduling as a manual task in the orchestration process or you unscheduled the line, and if you include a scheduled date and a warehouse attribute, or a scheduled date and a supplier and supplier site attribute in your REST payload, then Order Management will automatically schedule the fulfillment line. Order Management applies this same behavior when you use splitFulfillmentLineRequest. For example, if you include these scheduling attributes in your payload, then it will automatically schedule the original line and the split lines.

For more about import, see *Overview of Importing Orders Into Order Management*.

Other Features

You can also update or split a fulfillment line with other Oracle applications:

- Oracle Global Order Promising, see *Split Order Lines When Supply Meets a Threshold* and *Split or Substitute Fulfillment Lines*.
- Oracle Backlog Management, see the Split Order Lines to Reduce Delay During Rescheduling feature. Click *here* for more.

Privileges

You need these privileges to use updateSchedulingAttributeRequest in the Sales Orders for Order Hub REST API:

- Create Sales Orders Using REST Services (FOM_SALES_ORDER_REST_POST_PRIV)
- Modify Orchestration Order Fulfillment Line Attributes (DOO_MODIFY_ORCHESTRATION_ORDER_FULFILLMENT_LINE_ATTRIBUTES_PRIV)

You need these privileges to use splitFulfillmentLineRequest in the Sales Orders for Order Hub REST API:

- Create Sales Orders Using REST Services (FOM_SALES_ORDER_REST_POST_PRIV)
- Split Orchestration Order Fulfillment Line (DOO_SPLIT_ORCHESTRATION_ORDER_FULFILLMENT_LINE_PRIV)

Import Shipping Method

The Shipping Method attribute is a combination of three attributes. You use different attributes depending on the technology you use to import.

Sales Orders for Order Hub Resource in REST API	SourceSalesOrderImportTemplate Through File-Based Data Import
ShippingCarrierId	Shipping Carrier Code
ShippingMode	Shipping Mode

Sales Orders for Order Hub Resource in REST API	SourceSalesOrderImportTemplate Through File-Based Data Import
ShippingServiceLevel	Shipping Service Level

If you import a shipping method through REST API or FBDI, you must make sure the method is valid.

- If you include a value for one of the three attributes, then you must include a value for all of them. For example, if you include a value for ShippingCarrierId, then you must include a value for ShippingMode and ShippingServiceLevel.
- The WSH_MODE_OF_TRANSPORT lookup type must contain the value you import for ShippingMode.
- The WSH_SERVICE_LEVELS lookup type must contain the value you import for ShippingServiceLevel.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

The combination of ShippingCarrierId, ShippingMode, and ShippingServiceLevel must be valid with each other. Here's an easy way to identify valid combinations that you can import.

1. Go to the Order Management work area, then create a sales order.
2. In the Order Lines area, click **Shipment Details**.
3. On the Shipment Details tab, in the Shipping Method attribute, click **down arrow > Search**.
4. In the Select Shipping Method dialog, leave the attributes in the Search area empty and click **Search**.
5. Select a set of values from the search results and use them in your REST API payload or FBDI template.

   Here are some example results.

Shipping Method	Carrier	Mode of Transport	Service Level
Airborne Parcel Express	Airborne	Parcel	Express
Airborne Parcel 2nd day air	Airborne	Parcel	2nd day air
Airborne Parcel Standard	Airborne	Parcel	Standard
Big Rigs Truckload Ground	Big Rigs	Truckload	Ground
Nautilus Rail Standard	Nautilus	Rail	Standard
Nautilus Ocean Standard	Nautilus	Ocean	Standard

Shipping Method	Carrier	Mode of Transport	Service Level

Here are some examples of combinations that aren't valid.

Shipping Method	Description
Nautilus Standard	Standard isn't a mode of transport.
Airborne Express Parcel	Express isn't a mode of transport. Its a service. Parcel isn't a service. Its a mode of transport.

Identify Lookup Values You Can Use

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Manufacturing and Supply Chain Materials Management
    - Functional Area: Carriers and Transit Times
    - Task: Manage Carriers Lookups
2. Identify the modes of transport you can use.
    - On the Manage Carriers Lookups page, search Lookup Type for WSH_MODE_OF_TRANSPORT.
    - In the search results, examine the values in the Lookup Meaning column.

      Example results include Air, Less than Truckload, Ocean, Parcel, Rail, Truckload, and so on. Set the ShippingMode attribute in your import payload to one of these values.

      If you don't see the lookup code you need, then click **Actions > New** in the Lookup Codes area, and add it.
3. Identify the service levels you can use.
    - On the Manage Carriers Lookups page, search Lookup Type for WSH_SERVICE_LEVELS.
    - In the search results, examine the values in the Lookup Meaning column.

      Example results include Door-to-door, Express, FedEx 2day, and so on. Set the ShippingServiceLevel attribute in your import payload to one of these values.

      If you don't see the lookup code you need, then click **Actions > New** in the Lookup Codes area, and add it.

Troubleshoot

If you import a shipping method that isn't valid, then the communication with Global Order Promising might fail. You will notice when the Order Entry Specialist attempts to submit a sales order but receives an error, similar to:

```
You can't submit the sales order because the combination of the mode of transport, service level, and
 carrier that determines the shipping method on the order header isn't valid.
```

Here are some solutions you can try.

Trouble	Shoot
The DOO_VALIDATE_ORDER_SHIP_METHOD processing constraint or DOO_VALIDATE_FULFILL_LINE_SHIP_METHOD constraint examined the shipping method and found that it isn't valid.	Reimport the shipping method with the correct values.
The mapping in MSC_XREF_MAPPING isn't valid. The msc_xref_mapping entity in the Order Orchestration and Planning Data Repository maps values for the shipping method. For example, assume you see Airborne Parcel 2nd day air in the Order Management work area, but msc_xref_mapping maps the alphanumeric value to a numeric value, so the repository stores only the numeric value. • Map Airborne to 32510 and store 32510 in the carrier_id column of the repository. • Map Parcel to 39 and store 39 in the ship_mode_of_transport column of the repository. • Map 2nd day air to 821 and store 821 in the ship_class_of_service column of the repository. Assume you create a pretransformation rule or an order management extension that transforms the carrier ID to -999 before or during order submit. msc_xref_mapping doesn't map -999, so the carrier isn't valid and Order Management will display an error.	Make sure msc_xref_mapping correctly maps your values. Make sure your set ups don't modify the mappings that msc_xref_mapping references.

Related Topics

- Import Orders Into Order Management

Import Returns When You Split the Original Order Line

Create a return for a source order line that you split in Order Management.

If Order Management doesn't split the fulfillment line, then one order line in the source order in the source system maps to one fulfillment line in the sales order in Order Management. But if Order Management splits the fulfillment line into two or more fulfillment lines, then the lines reference the same source order, source order line, and schedule details. To

create a return for a source order line that you split in Order Management, you must provide details that identify which fulfillment line to return.

Use the DocumentReferenceType attribute to identify the original order line you're returning.

DocumentReferenceType	Description
ORIGINAL_SALES_ORDER	Identify the original order line in the source system from a channel, such as a legacy order capture system.
ORIGINAL_ORCHESTRATION_ORDER	Identify the original sales order line in Order Management. You must use DocumentSubLineIdentifier.

If the original order line.

- Isn't split, you can use ORIGINAL_SALES_ORDER or ORIGINAL_ORCHESTRATION_ORDER.
- Is split, you must use ORIGINAL_ORCHESTRATION_ORDER, or ORIGINAL_SALES_ORDER and ORIGINAL_ORCHESTRATION_ORDER.

Use ORIGINAL_SALES_ORDER

Use ORIGINAL_SALES_ORDER in the DocumentReference section to specify details about the source order you originally created in your source system.

Attribute	Description	Example Value
DocumentReferenceType	Not applicable.	ORIGINAL_SALES_ORDER
DocumentIdentifier	Value that uniquely identifies the original source order. This is the source order in your source system that you originally used to order the item.	CS_SO_1234445555565656
DocumentAdditionalIdentifier	Name of the source system. You can use the Code attribute on the Manage Planning Source Systems page to set this value.	LEG
DocumentNumber	Number of the original sales order.	CS_SO_122_0034
DocumentLineIdentifier	Number that uniquely identifies the order line on the original sales order that contains the item you're returning.	101
DocumentLineNumber	Identify the line number.	Not applicable

Note

- Provide values that uniquely identify the source order line. For instance, you can provide identifiers for the source order and source order line to uniquely identify the source order line. You can also provide other values, such as source order number and source order line number, to uniquely identify the source order line.
- Include values for DocumentNumber or DocumentIdentifier, or DocumentNumber and DocumentIdentifier.
- Include values for DocumentLineNumber or DocumentLineIdentifier, or DocumentLineNumber and DocumentLineIdentifier.

Use ORIGINAL_ORCHESTRATION_ORDER

Add another DocumentReference section, and use ORIGINAL_ORCHESTRATION_ORDER to identify details about the order you originally created in Oracle Order Management.

Attribute	Description	Example Value
DocumentReferenceType	Not applicable.	ORIGINAL_ORCHESTRATION_ORDER
DocumentIdentifier	Value that uniquely identifies the order header of the original sales order in Order Management. This is the sales order you originally used to order the item.	300100177488708
DocumentAdditionalIdentifier	Name of the source system.	GPR
DocumentNumber	Number of the original sales order.	45098
DocumentSubLineIdentifier	Value that uniquely identifies the fulfillment line that fulfilled the item you're returning.	300100177495497
DocumentLineIdentifier LineNumber DocumentLineNumber DocumentSubLineNumber	Leave these values empty.	Not applicable

Identify the Source System

You use different attributes to identify the name of the source system depending on whether you use the import template or web service.

How You Import	Attribute You Use
Template	Source Transaction System
Web Service	DocumentAdditionalIdentifier

Example Payload

Here's an example payload. It uses ORIGINAL_SALES_ORDER and ORIGINAL_ORCHESTRATION_ORDER. You use the DocumentReference section to specify details about the original sales order. Use the Order Information Service to get the values you need for your payload. For details, see *Web Services That You Can Use to Integrate Order Management*.

```
<ns2:DocumentReference>
 <ns2:DocumentReferenceType>ORIGINAL_SALES_ORDER</ns2:DocumentReferenceType>
 <ns2:DocumentIdentifier>CS_SO_1234445555565656</ns2:DocumentIdentifier>
 <ns2:DocumentAdditionalIdentifier>LEG</ns2:DocumentAdditionalIdentifier>
 <ns2:DocumentNumber>CS_SO_122_0034</ns2:DocumentNumber>
 <ns2:DocumentAdditionalNumber/>
 <ns2:DocumentLineIdentifier>101</ns2:DocumentLineIdentifier>
 <ns2:DocumentAdditionalLineIdentifier/>
 <ns2:DocumentLineNumber/>
 <ns2:DocumentAdditionalLineNumber/>
 <ns2:DocumentAdditionalSubLineIdentifier/>
 <ns2:DocumentSubLineNumber/>
 <ns2:DocumentAdditionalSubLineNumber/>
</ns2:DocumentReference>

<ns2:DocumentReference>
 <ns2:DocumentReferenceType>ORIGINAL_ORCHESTRATION_ORDER</ns2:DocumentReferenceType>
 <ns2:DocumentIdentifier>300100177488708</ns2:DocumentIdentifier>
 <ns2:DocumentAdditionalIdentifier>GPR</ns2:DocumentAdditionalIdentifier>
 <ns2:DocumentNumber>45098</ns2:DocumentNumber>
 <ns2:DocumentAdditionalNumber/>
 <ns2:DocumentLineIdentifier>300100177488711</ns2:DocumentLineIdentifier>
 <ns2:DocumentSubLineIdentifier>300100177495497</ns2:DocumentSubLineIdentifier>
 <ns2:DocumentAdditionalLineIdentifier/>
 <ns2:DocumentLineNumber>1</ns2:DocumentLineNumber>
 <ns2:DocumentAdditionalLineNumber/>
 <ns2:DocumentSubLineNumber>2</ns2:DocumentSubLineNumber>
 <ns2:DocumentAdditionalSubLineNumber/>
</ns2:DocumentReference>
```

Here's the entire payload. It returns a quantity of 1 Each for item AS92888.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:BatchName>Test001</ns2:BatchName>
 <ns2:Order>
 <ns2:SourceTransactionIdentifier>ppar_mar22_12</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionNumber>ppar_mar22_12</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:BuyingPartyContactId>1560</ns2:BuyingPartyContactId>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2018-01-01T06:08:52.0340</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:RequestingLegalUnitIdentifier>204</ns2:RequestingLegalUnitIdentifier>
 <ns2:FreezePriceFlag>true</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>true</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>true</ns2:FreezeTaxFlag>
 <ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
 <ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
 <ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
 <ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
 <ns2:BillToCustomerName>1006</ns2:BillToCustomerName>
 <ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
 <ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
 <ns2:Line>
 <ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
```

```
<ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
<ns2:ProductNumber>AS92888</ns2:ProductNumber>
<ns2:OrderedQuantity>1</ns2:OrderedQuantity>
<!--ns2:OrderedUOMCode>BX1</ns2:OrderedUOMCode-->
<ns2:OrderedUOM>Each</ns2:OrderedUOM>
<ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:RequestedShipDate>2018-01-01T06:08:52.0340</ns2:RequestedShipDate>
<ns2:PaymentTermsCode>4</ns2:PaymentTermsCode>
<ns2:TransactionCategoryCode>RETURN</ns2:TransactionCategoryCode>
<ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
<ns2:ScheduleShipDate>2018-11-20T06:08:52.0340</ns2:ScheduleShipDate>
<ns2:ScheduleArrivalDate>2018-11-20T06:08:52.0340</ns2:ScheduleArrivalDate>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
<ns2:BillToCustomerName>1006</ns2:BillToCustomerName>
<ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
<ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:ShippingInstructions>BM Ship Instructions- Ship it in a day</ns2:ShippingInstructions>
<ns2:UnitListPrice currencyCode="USD">105</ns2:UnitListPrice>
<ns2:PartialShipAllowedFlag>FALSE</ns2:PartialShipAllowedFlag>
<ns2:FOBPoint>DEST</ns2:FOBPoint>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>1</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>-100</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>-10</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>-10</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>-100</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>-90</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>-90</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>-9</ns2:ChargeCurrencyUnitPrice>
```

```
<ns2:HeaderCurrencyUnitPrice>-9</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>

<ns2:DocumentReference>
<ns2:DocumentReferenceType>ORIGINAL_SALES_ORDER</ns2:DocumentReferenceType>
<ns2:DocumentIdentifier>CS_SO_1234445555565656</ns2:DocumentIdentifier>
<ns2:DocumentAdditionalIdentifier>LEG</ns2:DocumentAdditionalIdentifier>
<ns2:DocumentNumber>CS_SO_122_0034</ns2:DocumentNumber>
<ns2:DocumentAdditionalNumber/>
<ns2:DocumentLineIdentifier>101</ns2:DocumentLineIdentifier>
<ns2:DocumentAdditionalLineIdentifier/>
<ns2:DocumentLineNumber/>
<ns2:DocumentAdditionalLineNumber/>
<ns2:DocumentAdditionalSubLineIdentifier/>
<ns2:DocumentSubLineNumber/>
<ns2:DocumentAdditionalSubLineNumber/>
</ns2:DocumentReference>

<ns2:DocumentReference>
<ns2:DocumentReferenceType>ORIGINAL_ORCHESTRATION_ORDER</ns2:DocumentReferenceType>
<ns2:DocumentIdentifier>300100177488708</ns2:DocumentIdentifier>
<ns2:DocumentAdditionalIdentifier>GPR</ns2:DocumentAdditionalIdentifier>
<ns2:DocumentNumber>45098</ns2:DocumentNumber>
<ns2:DocumentAdditionalNumber/>
<ns2:DocumentLineIdentifier>300100177488711</ns2:DocumentLineIdentifier>
<ns2:DocumentSubLineIdentifier>300100177495497</ns2:DocumentSubLineIdentifier>
<ns2:DocumentAdditionalLineIdentifier/>
<ns2:DocumentLineNumber>1</ns2:DocumentLineNumber>
<ns2:DocumentAdditionalLineNumber/>
<ns2:DocumentSubLineNumber>2</ns2:DocumentSubLineNumber>
<ns2:DocumentAdditionalSubLineNumber/>
</ns2:DocumentReference>

</ns2:Line>
<ns2:OrderPreferences>
<!--Optional:-->
<ns2:CreateCustomerInformationFlag>false</ns2:CreateCustomerInformationFlag>
<!--Optional:-->
<ns2:SubmitFlag>true</ns2:SubmitFlag>
</ns2:OrderPreferences>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Related Topics

- Web Services That You Can Use to Integrate Order Management

Cancel Backorder During Import

You can use the Order Import web service to cancel a backordered line.

- Assume you import a quantity of 10 on order line 1001.
- Some time later, Shipping splits the line into two lines, line 1002 with a quantity of 6 and line 1003 with a quantity of 4.

- Shipping ships line 1002 and sets the status for line 1002 to Awaiting Billing, but sets line 1003 to Backordered because no more inventory is available to ship.
- You receive a request from your customer to cancel the sales order.
- The order import service uses SourceTransactionLineIdentifier to identify the entire quantity for line 1001. It doesn't distinguish between lines 1002 and 1003 because it isn't aware that these lines exist.
- Line 1003 hasn't shipped so you don't need to cancel the quantity for it. You only need to cancel the quantity of the line that has shipped. So, set the OrderedQuantity attribute to 6 on order line 1001 in your web service payload.

Here's an example.

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">>
<ns1:createOrders
xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
<ns1:request
xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
<ns2:BatchName ></ns2:BatchName>
<ns2:Order>
<ns2:SourceTransactionNumber>ORDER_CANCEL_12</ns2:SourceTransactionNumber>
<ns2:SourceTransactionSystem>LEG</ns2:SourceTransactionSystem>
<ns2:SourceTransactionIdentifier>ORDER_CANCEL_12</ns2:SourceTransactionIdentifier>
<ns2:CustomerPONumber></ns2:CustomerPONumber>
<ns2:BuyingPartyName>Computer Service and Rentals</ns2:BuyingPartyName>
<ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
<ns2:TransactionOn>2017-12-06T19:27:11</ns2:TransactionOn>
<ns2:TransactionTypeCode></ns2:TransactionTypeCode>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:RequestingLegalUnit>204</ns2:RequestingLegalUnit>
<ns2:BillToCustomerName>Computer Service and Rentals</ns2:BillToCustomerName>
<ns2:ShipToPartyName>Computer Service and Rentals</ns2:ShipToPartyName>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:FreezeShippingChargeFlag>true</ns2:FreezeShippingChargeFlag>
<ns2:FreezeTaxFlag>true</ns2:FreezeTaxFlag>


<ns2:Line>
<ns2:SourceTransactionLineIdentifier>1001</ns2:SourceTransactionLineIdentifier>

<ns2:SourceTransactionScheduleIdentifier>1001</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
<ns2:ProductNumber>AS54888</ns2:ProductNumber>
<ns2:OrderedQuantity>6</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
<ns2:RequestedFulfillmentOrganizationIdentifier>204</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:ParentLineReference></ns2:ParentLineReference>
<ns2:RootParentLineReference></ns2:RootParentLineReference>
<ns2:ShippingInstructions></ns2:ShippingInstructions>
<ns2:PackingInstructions></ns2:PackingInstructions>
<ns2:RequestedShipDate>2017-12-06T00:00:00</ns2:RequestedShipDate>
<ns2:PaymentTerms>30 Net</ns2:PaymentTerms>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:ShipToPartyName>Computer Associates International</ns2:ShipToPartyName>
<ns2:ShipToPartyContactName>Jimmy Anderson</ns2:ShipToPartyContactName>
<ns2:ShipToAddress1>102, CityView</ns2:ShipToAddress1>
<ns2:ShipToCity>CHATTANOOGA</ns2:ShipToCity>
```

```
<ns2:ShipToPostalCode>37401</ns2:ShipToPostalCode>
<ns2:ShipToState>TN</ns2:ShipToState>
<ns2:ShipToCountry>US</ns2:ShipToCountry>
<ns2:ShipToPartyType>ORGANIZATION</ns2:ShipToPartyType>
<ns2:ShipToPartyType>ORGANIZATION</ns2:ShipToPartyType>
<ns2:BillToCustomerName>Computer Associates International</ns2:BillToCustomerName>
<ns2:BillToAccountContactName>Jimmy Anderson</ns2:BillToAccountContactName>
<ns2:BillToAddress1>102, CityView</ns2:BillToAddress1>
<ns2:BillToCity>CHATTANOOGA</ns2:BillToCity>
<ns2:BillToPostalCode>37401</ns2:BillToPostalCode>
<ns2:BillToState>TN</ns2:BillToState>
<ns2:BillToCountry>US</ns2:BillToCountry>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:BillToPartyType>ORGANIZATION</ns2:BillToPartyType>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:UnitListPrice>150.0</ns2:UnitListPrice>
<ns2:UnitSellingPrice>150.0</ns2:UnitSellingPrice>
<ns2:ContractStartDate>2016-11-22</ns2:ContractStartDate>
<ns2:ExtendedAmount>150.0</ns2:ExtendedAmount>
<ns2:TaxExempt>S</ns2:TaxExempt>
<ns2:OrigSysDocumentReference>ORIGSYS</ns2:OrigSysDocumentReference>
<ns2:OrigSysDocumentLineReference>ORIGSYSLINE</ns2:OrigSysDocumentLineReference>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>10</ns2:PricedQuantity>
<ns2:PricedQuantityUOMCode>Ea</ns2:PricedQuantityUOMCode>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PricePeriodicityCode></ns2:PricePeriodicityCode>
<ns2:GsaUnitPrice></ns2:GsaUnitPrice>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>150.0</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>4</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>150.0</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>150.0</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC3</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>150.0</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>150.0</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>3</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>150.0</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>150.0</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
```

```
<ns2:ChargeCurrencyExtendedAmount>150.0</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
</ns2:Line>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Related Topics

- Import Orders Into Order Management
- Export and Import Setup Data with Inventory Organizations
- How Order Management Transforms Source Orders Into Sales Orders

Import Pricing

Freeze Price on Sales Orders

If your source order contains price details and you don't want to recalculate or reevaluate them, then you can set the freeze attributes in the source order to Y.

- Freeze Price
- Freeze Shipping Charge
- Freeze Tax

The term *freeze* means Oracle Pricing won't recalculate charges. For example, if you set Freeze Price to:

- **Y**. Don't recalculate charges.
- **N**. Recalculate charges.

Here's how Pricing maps these freeze attributes to other attributes on the header entity of the service data object (SDO) that you use for pricing.

Attribute in Source Order	Attribute on Header of Service Data Object
FreezePricing	CalculatePricingChargesFlag
FreezeShippingCharge	CalculateShippingChargesFlag
FreezeTax	CalculateTaxFlag

Pricing will use the values in your import payload to determine price, shipping charges, and tax. The attributes that you use depend on how you import price data. For details, see *Import Price Lists* and *Attributes That You Can Use with Web Services*.

Attributes That You Must Include When You Freeze Price

If you freeze pricing, then you must include values for these attributes in your import payload:

- Header Currency Unit Price
- Header Currency Extended Amount

If you freeze pricing and your sales order has a recurring charge, then you must include values for these attributes in your import payload:

- Charge Currency Duration Extended Amount. If you're using file-based data import, then use the Duration Charge Total attribute.
- Header Currency Duration Extended Amount. If you're using file-based data import, then use the Duration Header Total attribute.

Freeze Price for Coverage Items

If you copy a sales order in the Order Management work area that includes a configured item, and if a coverage covers the configured item, a configure option, or an option class, then Order Management copies each of these coverages to the new sales order.

Order Management copies a coverage that covers an included item differently depending on whether you freeze price, shipping charges, and tax.

Frozen	Not Frozen
Order Management doesn't copy a coverage that covers an included item to the new order.	Order Management copies each coverage that covers an included item to the new sales order.

For details, see *Set Up Coverages for Sales Orders*.

Freeze Price on Return Orders

If you set the FreezePriceFlag attribute to Y in your import payload for a return order, then the import doesn't calculate price but instead automatically reverses the value that your import payload contains for the ChargeCurrencyUnitPrice attribute. For example, if ChargeCurrencyUnitPrice equals 2960 in your import payload, then the import sets ChargeCurrencyUnitPrice to -2960 (negative 2960) on the return order.

So you must make sure your payload contains accurate data.

Assume you have this import payload.

```
<ns2:OrderChargeComponent>
 <ns2:HeaderCurrencyCode>SAR</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>2960</ns2:HeaderCurrencyExtendedAmount>
 <ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
 <ns2:SequenceNumber>2</ns2:SequenceNumber>
 <ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
 <ns2:HeaderCurrencyUnitPrice>1480</ns2:HeaderCurrencyUnitPrice>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
 <ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
 <ns2:ChargeCurrencyCode>SAR</ns2:ChargeCurrencyCode>
 <ns2:ChargeCurrencyExtendedAmount>2960</ns2:ChargeCurrencyExtendedAmount>
 <ns2:ChargeCurrencyUnitPrice>2960</ns2:ChargeCurrencyUnitPrice>
</ns2:OrderChargeComponent>
```

This payload will result in an incorrect value for ChargeCurrencyUnitPrice.

ChargeCurrencyUnitPrice should equal ChargeCurrencyExtendedAmount divided by Quantity. In this example, assume the quantity on the return line is 2. You should set ChargeCurrencyUnitPrice to 1480 instead of 2960 because ChargeCurrencyExtendedAmount divided by Quantity is 2960 divided 2, which equals 1480. The import will then automatically set ChargeCurrencyUnitPrice to -1480 to reverse the charge.

Don't Modify Freeze Attributes After You Import and Submit

If you import a sales order and submit it, then import a revision of the order, then you must make sure the values of the freeze attributes in your import payload match the values that you used the first time you imported. For example, if you set the FreezePricing attribute to `yes` the first time you import, then FreezePricing must contain `yes` in every subsequent import.

Related Topics

- Import Orders Into Order Management
- Attributes That You Can Use with Web Services
- Set Up Coverages for Sales Orders
- How Order Management Transforms Source Orders Into Sales Orders
- Import Price Lists

Import Rollup Charges for Configured Items

You can specify whether you want Oracle Pricing to calculate the rolled up price when you import an assemble-to-order or pick-to-order item.

I Want Pricing to Calculate the Price	What You Need to Do
Yes	Don't include the rolled up charge on the configured item's root line.
No	Include the rolled up charge on the root line.

For details, see *Set Up Pricing for Your Configuration Model*.

Freeze Pricing

You set these attributes to freeze pricing:

```
FreezePriceFlag is true
FreezeShippingChargeFlag is true
FreezeTaxFlag is true
```

If you don't set all three freeze attributes to true, then you must include the rolled up charge in your import payload.

If you freeze pricing, then you must use these values in your import payload:

Type of Item	PrimaryFlag	RollupFlag
Standard	`true`	`false`
Kit	`true`	`false`
You import the **root charge** for:	`true`	`false`

Type of Item	PrimaryFlag	RollupFlag
• Assemble-to-order model • Pick-to-order model		Include the model's base charge on the root line in your import.
You import the **rolled up** charge for: • Assemble-to-order model • Pick-to-order model	`true`	`true` Include the rolled up charge on the model's root line in your import. The rolled up charge equals the root charge plus the sum of all the configure options' charges.

For details, see *Freeze Price on Sales Orders*.

Example

The root order line for a configured item includes two charges:

- The first charge contains the base charge.
- The second charge contains the rolled up charge.

The rolled up charge equals the base charge plus the total charge for all the configure options that your customer selects.

Assume you need to import a quantity of 10 selling at $215 each for the STOVE_ATO_MODEL assemble-to-order configuration model, and you don't want Oracle Pricing to calculate the rolled up charge. Instead, you want to import the rolled up charge. Here's your set up:

Entity	Pseudocode

Order header.	`Order`
	`FreezePriceFlag is true`
	`FreezeShippingChargeFlag is true`
	`FreezeTaxFlag is false`

Configured item's root order line.	`Order`
	`Line`
	`ProductNumber is STOVE_ATO_MODEL`
	`OrderedQuantity is 10`

First OrderCharge entity on the root line. This entity contains the configured item's base charge.	`Order`
	`Line`
	`OrderCharge`
	`PrimaryFlag is true`
	`RollupFlag is false`
	`OrderChargeComponent`
	`HeaderCurrencyExtendedAmount is 100`
	`ChargeCurrencyUnitPrice is 10`
	`HeaderCurrencyUnitPrice is 10`
	`RollupFlag is false`
	`ChargeCurrencyExtendedAmount is 100`

Entity	Pseudocode
Second OrderCharge entity on the root line. This entity contains the configured item's rolled up charge.	Order Line OrderCharge. . . OrderCharge PrimaryFlag is true RollupFlag is true OrderChargeComponent HeaderCurrencyExtendedAmount is 2150 ChargeCurrencyUnitPrice is 215 HeaderCurrencyUnitPrice is 215 RollupFlag is true ChargeCurrencyExtendedAmount is 2150

Here's the entire payload for this example.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
 <ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
 <ns2:BatchName/>
 <ns2:Order>
 <ns2:SourceTransactionNumber>ATO_MODEL_PP_006</ns2:SourceTransactionNumber>
 <ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
 <ns2:SourceTransactionIdentifier>ATO_MODEL_PP_006</ns2:SourceTransactionIdentifier>
 <ns2:BuyingPartyId>1006</ns2:BuyingPartyId>
 <ns2:BuyingPartyContactId>1560</ns2:BuyingPartyContactId>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2022-10-08T23:58:52Z</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:FreezePriceFlag>true</ns2:FreezePriceFlag>
 <ns2:FreezeShippingChargeFlag>true</ns2:FreezeShippingChargeFlag>
 <ns2:FreezeTaxFlag>false</ns2:FreezeTaxFlag>
 <ns2:RequestingLegalUnitIdentifier>204</ns2:RequestingLegalUnitIdentifier>
 <ns2:OrigSystemDocumentReference/>
 <ns2:PartialShipAllowedFlag>true</ns2:PartialShipAllowedFlag>
 <ns2:Line>
 <ns2:SourceTransactionLineIdentifier>105</ns2:SourceTransactionLineIdentifier>
 <ns2:SourceTransactionScheduleIdentifier>105</ns2:SourceTransactionScheduleIdentifier>
 <ns2:SourceTransactionLineNumber>5</ns2:SourceTransactionLineNumber>
 <ns2:SourceTransactionScheduleNumber>5</ns2:SourceTransactionScheduleNumber>
 <ns2:ProductNumber>STOVE_ATO_MODEL</ns2:ProductNumber>
 <ns2:OrderedQuantity>10</ns2:OrderedQuantity>
 <ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
 <ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
 <ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
 <ns2:ParentLineReference/>
 <ns2:RootParentLineReference/>
 <ns2:RequestedShipDate>2022-10-08T23:08:52Z</ns2:RequestedShipDate>
 <ns2:PaymentTerms>30 Net</ns2:PaymentTerms>
 <ns2:PaymentTermsCode>4</ns2:PaymentTermsCode>
 <ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
 <ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
 <ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
 <ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
 <ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
 <ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
 <ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
 <ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
 <ns2:PartialShipAllowedFlag>true</ns2:PartialShipAllowedFlag>
 <ns2:UnitListPrice>5</ns2:UnitListPrice>
 <ns2:UnitSellingPrice>5</ns2:UnitSellingPrice>
```

```
<ns2:ExtendedAmount>100</ns2:ExtendedAmount>
<ns2:TaxExemptFlag>S</ns2:TaxExemptFlag>
<ns2:TaxClassificationCode>FUS_US_ET_STND_RT1</ns2:TaxClassificationCode>
<ns2:OrigSysDocumentReference>ORIGSYS</ns2:OrigSysDocumentReference>
<ns2:OrigSysDocumentLineReference>ORIGSYSLINE</ns2:OrigSysDocumentLineReference>
<ns2:UnitQuantity>1</ns2:UnitQuantity>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>10</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>100</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>10</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>10</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>100</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>100</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>100</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>10</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>10</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC1</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>10</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>true</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC11</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
```

```
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>2150</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>215</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>215</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>true</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC11</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC21</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>2150</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>2150</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>2150</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>215</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>215</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>true</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC11</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC11</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
</ns2:Line>
<ns2:Line>
<ns2:SourceTransactionLineIdentifier>106</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>106</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>6</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>6</ns2:SourceTransactionScheduleNumber>
<ns2:ProductNumber>FUEL_OC</ns2:ProductNumber>
<ns2:OrderedQuantity>10</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
<ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:ParentLineReference>105</ns2:ParentLineReference>
<ns2:RootParentLineReference>105</ns2:RootParentLineReference>
<ns2:RequestedShipDate>2022-10-08T23:08:52Z</ns2:RequestedShipDate>
<!--<ns2:PaymentTerms>30 Net</ns2:PaymentTerms>-->
<ns2:PaymentTermsCode>4</ns2:PaymentTermsCode>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
<ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
<ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
<ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
<ns2:PartialShipAllowedFlag>true</ns2:PartialShipAllowedFlag>
<ns2:UnitListPrice>5</ns2:UnitListPrice>
<ns2:UnitSellingPrice>5</ns2:UnitSellingPrice>
<ns2:ExtendedAmount>100</ns2:ExtendedAmount>
<ns2:TaxExemptFlag>S</ns2:TaxExemptFlag>
<ns2:TaxClassificationCode>FUS_US_ET_STND_RT1</ns2:TaxClassificationCode>
```

```
<ns2:OrigSysDocumentReference>ORIGSYS</ns2:OrigSysDocumentReference>
<ns2:OrigSysDocumentLineReference>ORIGSYSLINE</ns2:OrigSysDocumentLineReference>
<ns2:UnitQuantity>1</ns2:UnitQuantity>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>10</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>50</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>5</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>5</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC23</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>50</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>50</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>50</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>5</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>5</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC2</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC24</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
</ns2:Line>
<ns2:Line>
<ns2:SourceTransactionLineIdentifier>107</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>107</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>7</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>7</ns2:SourceTransactionScheduleNumber>
<ns2:ProductNumber>GAS_FUEL</ns2:ProductNumber>
<ns2:OrderedQuantity>10</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode>
<ns2:RequestedFulfillmentOrganizationIdentifier>207</ns2:RequestedFulfillmentOrganizationIdentifier>
<ns2:RequestingBusinessUnitIdentifier>204</ns2:RequestingBusinessUnitIdentifier>
<ns2:ParentLineReference>105</ns2:ParentLineReference>
<ns2:RootParentLineReference>105</ns2:RootParentLineReference>
<ns2:RequestedShipDate>2022-10-08T23:08:52Z</ns2:RequestedShipDate>
<!--<ns2:PaymentTerms>30 Net</ns2:PaymentTerms>-->
```

```
<ns2:PaymentTermsCode>4</ns2:PaymentTermsCode>
<ns2:TransactionCategoryCode>ORDER</ns2:TransactionCategoryCode>
<ns2:InventoryOrganizationIdentifier>204</ns2:InventoryOrganizationIdentifier>
<ns2:ShipToPartyIdentifier>1006</ns2:ShipToPartyIdentifier>
<ns2:ShipToPartySiteIdentifier>1036</ns2:ShipToPartySiteIdentifier>
<ns2:ShipToPartyContactIdentifier>1560</ns2:ShipToPartyContactIdentifier>
<ns2:BillToCustomerIdentifier>1006</ns2:BillToCustomerIdentifier>
<ns2:BillToAccountSiteUseIdentifier>1025</ns2:BillToAccountSiteUseIdentifier>
<ns2:BillToAccountContactIdentifier>4820</ns2:BillToAccountContactIdentifier>
<ns2:PartialShipAllowedFlag>true</ns2:PartialShipAllowedFlag>
<ns2:UnitListPrice>5</ns2:UnitListPrice>
<ns2:UnitSellingPrice>5</ns2:UnitSellingPrice>
<ns2:ExtendedAmount>100</ns2:ExtendedAmount>
<ns2:TaxExemptFlag>S</ns2:TaxExemptFlag>
<ns2:TaxClassificationCode>FUS_US_ET_STND_RT1</ns2:TaxClassificationCode>
<ns2:OrigSysDocumentReference>ORIGSYS</ns2:OrigSysDocumentReference>
<ns2:OrigSysDocumentLineReference>ORIGSYSLINE</ns2:OrigSysDocumentLineReference>
<ns2:UnitQuantity>1</ns2:UnitQuantity>
<ns2:OrderCharge>
<ns2:ChargeDefinitionCode>QP_SALE_PRICE</ns2:ChargeDefinitionCode>
<ns2:ChargeSubtypeCode>ORA_PRICE</ns2:ChargeSubtypeCode>
<ns2:PriceTypeCode>ONE_TIME</ns2:PriceTypeCode>
<ns2:PricedQuantity>10</ns2:PricedQuantity>
<ns2:PricedQuantityUOM>Each</ns2:PricedQuantityUOM>
<ns2:PricedQuantityUOMCode/>
<ns2:PrimaryFlag>true</ns2:PrimaryFlag>
<ns2:ApplyTo>PRICE</ns2:ApplyTo>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceChargeIdentifier>SC3</ns2:SourceChargeIdentifier>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:ChargeTypeCode>ORA_SALE</ns2:ChargeTypeCode>
<ns2:ChargeCurrencyCode/>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PricePeriodicityCode/>
<ns2:GsaUnitPrice/>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>2000</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_LIST_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>1</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>LIST_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>200</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>200</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC3</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC31</ns2:SourceChargeComponentIdentifier>
<ns2:ChargeCurrencyExtendedAmount>2000</ns2:ChargeCurrencyExtendedAmount>
</ns2:OrderChargeComponent>
<ns2:OrderChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>2000</ns2:HeaderCurrencyExtendedAmount>
<ns2:ChargeCurrencyExtendedAmount>2000</ns2:ChargeCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_NET_PRICE</ns2:PriceElementCode>
<ns2:SequenceNumber>2</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>NET_PRICE</ns2:PriceElementUsageCode>
<ns2:ChargeCurrencyUnitPrice>200</ns2:ChargeCurrencyUnitPrice>
<ns2:HeaderCurrencyUnitPrice>200</ns2:HeaderCurrencyUnitPrice>
<ns2:RollupFlag>false</ns2:RollupFlag>
<ns2:SourceParentChargeComponentId/>
<ns2:SourceChargeIdentifier>SC3</ns2:SourceChargeIdentifier>
<ns2:SourceChargeComponentIdentifier>SCC32</ns2:SourceChargeComponentIdentifier>
</ns2:OrderChargeComponent>
</ns2:OrderCharge>
```

```
</ns2:Line>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Import Tax on Sales Orders

Specify how to import tax on sales orders.

Specify Whether to Include Tax in Net Price

You can include or not include Tax in Net Price. In either case:

- Set the FreezeTaxFlag attribute to False.
- Add your ChargeComponent elements in this hierarchy:

```
<soap:Envelope
 <soap:Body>
 <ns1:process
 <ns1:OrchestrationOrderRequest
 <ns2:OrchestrationOrderRequestLine>
 <ns2:LineCharge>
 <ns2:ChargeComponent
```

- Get some example payloads. Go to *Technical Reference for Order Management (Doc ID 2051639.1)*, download the Payloads and Files attachment, then examine the examples that include or don't include tax in net price.

Include Tax in Net Price

1. Add details to the price adjustment. For example:

```
<ns2:ChargeComponent>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>-10</ns2:HeaderCurrencyExtendedAmount>
 <ns2:PriceElementCode>QP_DISCOUNT_ADJ</ns2:PriceElementCode>
 <ns2:SequenceNumber>2</ns2:SequenceNumber>
 <ns2:PriceElementUsageCode>PRICE_ADJUSTMENT</ns2:PriceElementUsageCode>
 <ns2:ChargeCurrencyUnitPrice>-1</ns2:ChargeCurrencyUnitPrice>
 <ns2:HeaderCurrencyUnitPrice>-1</ns2:HeaderCurrencyUnitPrice>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
 <ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
 <ns2:SourceChargeComponentIdentifier>SCC2</ns2:SourceChargeComponentIdentifier>
 <ns2:ChargeCurrencyExtendedAmount>-10</ns2:ChargeCurrencyExtendedAmount>
</ns2:ChargeComponent>
<ns2:ChargeComponent>
<ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
<ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
<ns2:HeaderCurrencyExtendedAmount>10</ns2:HeaderCurrencyExtendedAmount>
<ns2:PriceElementCode>QP_INCLUSIVE_TAX</ns2:PriceElementCode>
<ns2:SequenceNumber>3</ns2:SequenceNumber>
<ns2:PriceElementUsageCode>INCLUSIVE_TAX</ns2:PriceElementUsageCode>
```

   Note

   - Add the code starting right after the line that contains PriceElementCode up to and including `PriceElementUsageCode>INCLUSIVE_TAX`.

- Set ChargeCurrencyUnitPrice to -1 (negative 1).
- Set HeaderCurrencyUnitPrice to -1 (negative 1).

2. Add the charge component where you specify the tax to include in net price. For example:

```
<ns2:ChargeComponent>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>80</ns2:HeaderCurrencyExtendedAmount>
 <ns2:PriceElementCode>QP_NET_PRICE_EXC_INCLUSIVE_TAX</ns2:PriceElementCode>
 <ns2:SequenceNumber>5</ns2:SequenceNumber>
 <ns2:PriceElementUsageCode>NET_PRICE_TAX_EXCLUDED</ns2:PriceElementUsageCode>
 <ns2:ChargeCurrencyUnitPrice>8</ns2:ChargeCurrencyUnitPrice>
 <ns2:HeaderCurrencyUnitPrice>8</ns2:HeaderCurrencyUnitPrice>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
 <ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
 <ns2:SourceChargeComponentIdentifier>SCC5</ns2:SourceChargeComponentIdentifier>
 <ns2:ChargeCurrencyExtendedAmount>80</ns2:ChargeCurrencyExtendedAmount>
</ns2:ChargeComponent>
```

Note

- Set PriceElementCode to QP_NET_PRICE_EXC_INCLUSIVE_TAX.
- Set PriceElementUsageCode to NET_PRICE_TAX_EXCLUDED.

Don't Include Tax in Net Price

1. Add a charge component that specifies the tax to include. For example:

```
<ns2:ChargeComponent>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>10</ns2:HeaderCurrencyExtendedAmount>
 <ns2:PriceElementCode>QP_EXCLUSIVE_TAX</ns2:PriceElementCode>
 <ns2:SequenceNumber>4</ns2:SequenceNumber>
 <ns2:PriceElementUsageCode>EXCLUSIVE_TAX</ns2:PriceElementUsageCode>
 <ns2:ChargeCurrencyUnitPrice>1</ns2:ChargeCurrencyUnitPrice>
 <ns2:HeaderCurrencyUnitPrice>1</ns2:HeaderCurrencyUnitPrice>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
 <ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
 <ns2:SourceChargeComponentIdentifier>SCC4</ns2:SourceChargeComponentIdentifier>
 <ns2:ChargeCurrencyExtendedAmount>10</ns2:ChargeCurrencyExtendedAmount>
</ns2:ChargeComponent>
```

Note

- Set PriceElementCode to QP_EXCLUSIVE_TAX.
- Set PriceElementUsageCode to EXCLUSIVE_TAX.

2. Add the charge component where you specify the tax to include in net price. For example:

```
<ns2:ChargeComponent>
 <ns2:ChargeCurrencyCode>USD</ns2:ChargeCurrencyCode>
 <ns2:HeaderCurrencyCode>USD</ns2:HeaderCurrencyCode>
 <ns2:HeaderCurrencyExtendedAmount>100</ns2:HeaderCurrencyExtendedAmount>
 <ns2:PriceElementCode>QP_NET_PRICE_PLUS_TAX</ns2:PriceElementCode>
 <ns2:SequenceNumber>5</ns2:SequenceNumber>
 <ns2:PriceElementUsageCode>NET_PRICE_PLUS_TAX</ns2:PriceElementUsageCode>
```

```
 <ns2:ChargeCurrencyUnitPrice>10</ns2:ChargeCurrencyUnitPrice>
 <ns2:HeaderCurrencyUnitPrice>10</ns2:HeaderCurrencyUnitPrice>
 <ns2:RollupFlag>false</ns2:RollupFlag>
 <ns2:SourceParentChargeComponentId></ns2:SourceParentChargeComponentId>
 <ns2:SourceChargeIdentifier>SC1</ns2:SourceChargeIdentifier>
 <ns2:SourceChargeComponentIdentifier>SCC5</ns2:SourceChargeComponentIdentifier>
 <ns2:ChargeCurrencyExtendedAmount>100</ns2:ChargeCurrencyExtendedAmount>
<ns2:ChargeComponent>
```

Note

- Set PriceElementCode to QP_NET_PRICE_PLUS_TAX.
- Set PriceElementUsageCode to NET_PRICE_PLUS_TAX.

How Order Management Displays Your Tax

How You Include Tax	Values in Price Breakdown
Include Tax in Net Price	Pay Now is equal to Net Price. Total Tax includes only the inclusive tax.
Don't Include Tax in Net Price	Pay Now equals tax plus Net Price. Total Tax includes only the exclusive tax.

How Order Management Integrates Tax with Accounts Receivable

- If you freeze price or tax, then Pricing Administration doesn't calculate tax, and Order Management uses the price that you import to display tax on the sales order.
- Order Management creates a tax estimate. It doesn't create the actual tax that's charged in Accounts Receivable.
- Order Management doesn't send tax values to Accounts Receivable because conditions that affect tax during fulfillment might change. Assume you set the Warehouse attribute to Los Angeles on a sales order, then click Submit. Assume fulfillment depletes the inventory in Los Angeles before it fulfills the sales order, so fulfillment decides to ship from Denver instead of Los Angeles so it can meet the requested ship date. The tax rate in Los Angeles might be different than the tax rate in Denver. So, Accounts Receivable uses the actual fulfillment conditions to calculate its own tax regardless of whether you freeze price or tax. For details, see *Change Your Tax Rate*.

Related Topics

- Import Orders Into Order Management
- Attributes That You Can Use with Web Services
- Set Up Coverages for Sales Orders
- How Order Management Transforms Source Orders Into Sales Orders
- Import Price Lists

Example of Importing Price Details

Specify how to use price details when you import a sales order.

Example Values for This Topic

Assume there are two charges for the order line.

Original Order	Import Payload
UnitListPrice equals 100. Shipping charges equal 20. Tax equals 10.	Price equals 120. Shipping charges equal 24. Tax equals 12.

Assume.

- Current price for the item is 130.
- Shipping charges at runtime are price multiplied by 20%.
- Tax calculated at runtime is total net price plus shipping multiplied by 10%.

Values That You Can Use for Freeze Attributes with Unreferenced Returns

Here are the values you can use when your import includes a sales order line or a return line that doesn't reference the original sales order.

FreezePriceFlag	FreezeShippingChargeFlag	FreezeTaxFlag	Description
Y	Y	N	Pricing uses the price and shipping charges from the import payload, but calculates the tax. Price equals 120. Shipping charges equal 25. Tax equals 14.50.
Y	N	N	Pricing uses the price from the payload, but calculates shipping and tax. Price equals 120. Shipping charges equal 24. Tax equals 14.40.
N	Y	N	Pricing uses the shipping charges from the payload, but gets the price from your set up in Pricing Administration and calculates tax.

FreezePriceFlag	FreezeShippingChargeFlag	FreezeTaxFlag	Description
			Price equals 130. Shipping charges equal 24. Tax equals 15.40.

Values That You Can't Use for Freeze Attributes with Unreferenced Returns

Here are the values that you can't use when your import includes a sales order line or a return line that doesn't reference the original sales order.

FreezePriceFlag	FreezeShippingChargeFlag	FreezeTaxFlag	Description
N	N	Y	Pricing can calculate the price and shipping, but can't validate that the imported tax is correct for the price and the shipping it calculates.
Y	N	Y	Pricing can calculate shipping, but can't validate that the imported price and tax is correct for the shipping it calculates.
N	Y	Y	Pricing can calculate shipping and tax, but can't validate that the imported price is correct for the shipping and tax that it calculates.

Referenced Return Line

Assume your import includes a return line that does reference the original sales order.

FreezePriceFlag	FreezeShippingChargeFlag	FreezeTaxFlag	Description
Y	Y	Y	Pricing ignores the original sales order. Instead, it uses values from the import payload. Price equals 120. Shipping charges equal 24. Tax equals 12.
Y	N	Y	If at least one of these attributes equals N, then Pricing prices everything from the original sales order, and ignores price, shipping charges, and tax that the import payload contains. Price equals 100.

FreezePriceFlag	FreezeShippingChargeFlag	FreezeTaxFlag	Description
			Shipping charges equal 20. Tax equals 10.

Required Attributes

If you freeze pricing, then you must include values for required attributes in your import payload. For details, see *Freeze Price on Sales Orders*.

Related Topics

- Import Orders Into Order Management
- Attributes That You Can Use with Web Services
- Set Up Coverages for Sales Orders
- How Order Management Transforms Source Orders Into Sales Orders
- Import Price Lists

Troubleshoot

Troubleshoot Problems With Order Import

Fix problems that happen in your Order Management implementation when you import a source order.

Error Messages

Problem	Solution
I encounter an error message. `Order management did not import source order TEST_NOV21_02 because of the following error: An order was not created due to an unknown error. Check the run time user interface for processing status. Number of orders that did not pass validation: 1. Some of these orders may be available in the order entry user interface.` This problem might happen when the extended amount for the charge components on the coverage line doesn't contain a value.	If your order import includes a coverage item, and if your import data already defines pricing for the item before you do the import, then make sure the extended amount for the charge components on the coverage line contains a value. Make sure the Duration Header Total attribute on the DOO_ORDER_CHARGE_COMPS_INT tab of the order import template contains a value for each coverage line.
I encounter an error message.	Set the RollupFlag attribute to N on the DOO_ORDER_CHARGES_INT tab of the order import template.

Problem	Solution
Order xxx will not be imported due to the following error: An order was not created due to an unknown error. null. Request Id: 1018270 Batch Name: SOF1 Source System: OPS Order Number: null Allow Auto Purge: Y Number of orders that were imported: 0. Check the run time user interface for processing status. **Number of orders that did not pass validation: 1.** **Some of these orders may be available in the order entry user interface. Overall Result : ERROR**	
I import a sales order, but then encounter an error when I use the Order Management work area to change an attribute on the order, such as price, date, quantity, and so on. **You cannot save the draft Order because of following error: You cannot edit this Sales Order because you can not modify the Price, Tax Or the Shipping Charge. You cannot edit this Sale Order because it was created in a different Source System.**	Disable the DOO_IMPORTED_DRAFT_HEADER_UPDATE processing constraint. Note that, if you disable the constraint, and if you import a sales order that you already priced, and if you then use the Order Management work area to change the quantity or any other attribute that affects price, then Order Management doesn't reprice the order.
I encounter an error message. **A sales order was not created because source order does not include a value for required attribute ShipToPartyName. Provide a value for this attribute, and then submit the source order again.** This error might happen when you send contact details for the party or customer but don't include party or customer details.	DOO_ORDER_ADDRESSES_INT Party Identifier Party Number Party Name Customer Identifier Customer Number Customer Name Party Contact Identifier Party Contact Number Party Contact Name

Problems With Different Kinds of Data

I import a sales order and submit it. Order Management ships the order, but it gets stuck in Shipped status and I encounter an error that indicates the value of an attribute can't contain more than 30 characters. I attempt to use a recover action, but recover doesn't fix the problem.

This problem typically happens because Accounts Receivable in Oracle Applications requires that the value for some attributes not exceed 30 characters. Its possible that the value for an attribute in your import payload exceeds 30 characters. Have a look at them.

- SourceTransactionNumber
- SourceTransactionScheduleNumber
- SourceTransactionLineIdentifier

If you don't use Oracle Receivables to invoice your sales orders, and if your value must exceed 30 characters, then you can disable a constraint.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints.
2. On the Manage Processing Constraints page, disable the constraint.

Attribute	Value
Constraint Name	DOO_SOURCE_ATTRIBUTE_LENGTH_VALIDATION
Display Name	Source Attributes Length Validation

If you import a source order that has values that exceed 30 characters, and if the Source Attributes Length Validation constraint is enabled, then Order Management sets the order status to Draft and displays an error message in the Order Management work area. You must discard the draft, fix the character limit or disable the constraint, then import the source order again.

I Can't Import Binary Data

If you must import binary data, then you must convert your binary data to text before you do the import, and you must use the base64 encoding scheme when do the conversion.

Here's an example of payload data that failed import.

```
<ord:Attachment>
 <ord:FileContent>text/plain</
ord:FileContent><ord:Data>VGhlIEdlbmVyYWwgVGVybXMgYW5kIENvbmRpdGlvbiBpbiBDb2h1IHF1b3RhdGlvbiANCnNoYWxsIHByZXZhaWwgaW
ord:Data>
 <ord:FileName>text_document.txt</ord:FileName>
 <ord:url>text_document.txt</ord:url>
 <ord:DataTypeCode>FILE</ord:DataTypeCode>
 <ord:Category>Order Confirmation Notes</ord:Category>
</ord:Attachment>
```

This import failed because the ampersand (`&`) in the Data tag isn't in base64 encoding.

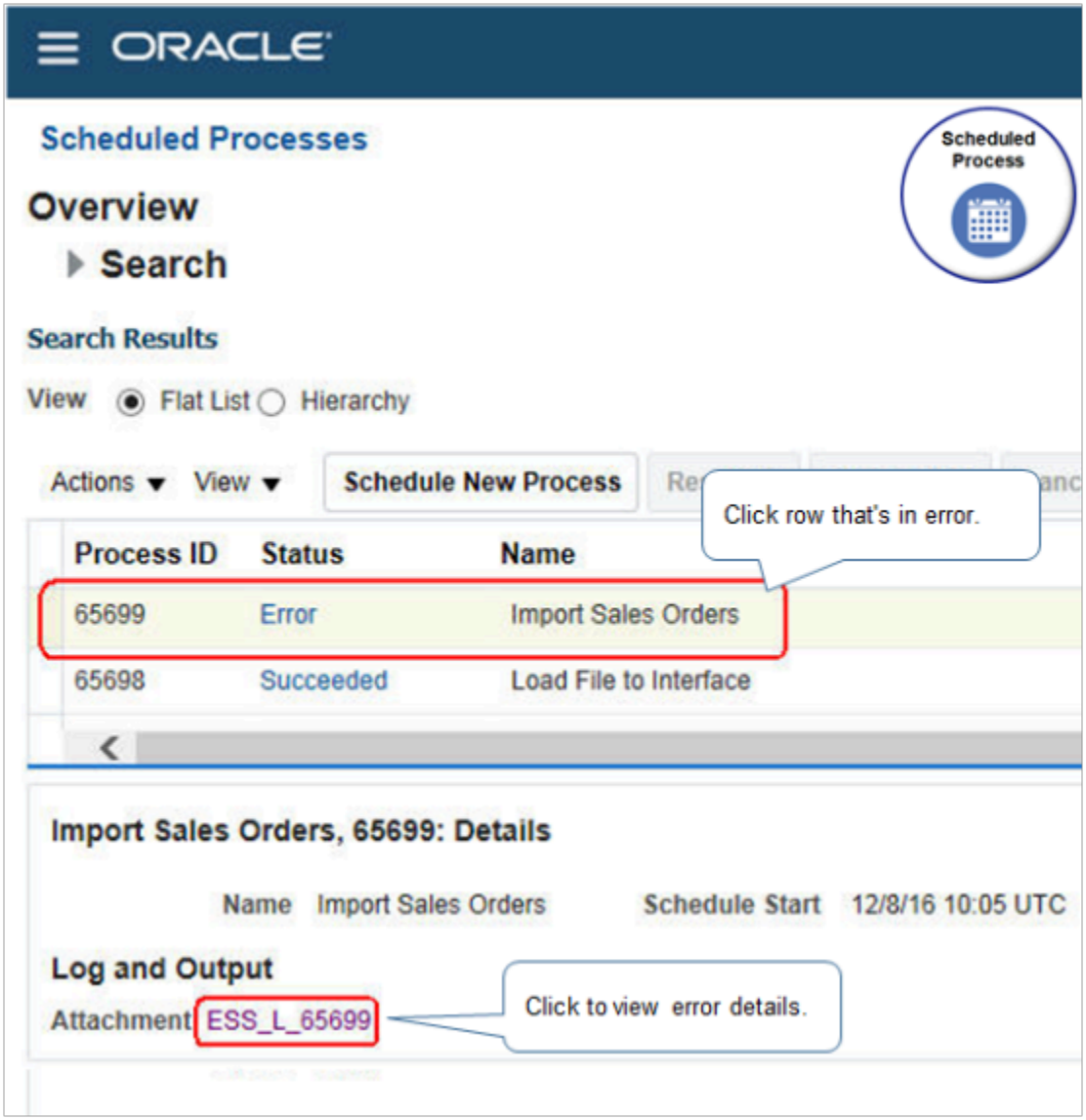
Problem	Solution
I encounter an error message.	Include details for one or more party attributes.

Problem	Solution
`A sales order was not created because source order does not include a value for required attribute ShipToPartyName. Provide a value for this attribute, and then submit the source order again.` This error might happen when you send contact details for the party or customer but don't include party or customer details.	• Party Identifier • Party Number • Party Name • Party Contact Identifier • Party Contact Number • Party Contact Name Include details for one or more customer attributes. • Customer Identifier • Customer Number • Customer Name

Scheduled Processes

Assume you run the *Import Sales Orders* scheduled process, it finishes running and reports status Error.

For example:



Note

1. On the Scheduled Processes page, in Search Results, click the **row** that's in error.
2. In the Log and Output area, next to Attachment, click the **link**.
3. Open the attachment and examine the error message.

For example:

```
Request Id: 65735
Batch Name: 10810
Source System: null
Order Number: null
Allow Auto Purge: Y
Order PMC_IMP_161208_002 will not be imported due to the following error: An order was not created because
 no matching row for attribute ProductNumber with the value AS54888777 was found for the source order with
 the following details: source order PMC_IMP_161208_002, source order line 1, source order schedule 11.
 Check the attribute value, and resubmit the order..
Number of orders that were imported: 0. Check the run time user interface for processing status.
Number of orders that did not pass validation: 1. Some of these orders may be available in the order entry
 user interface.
```

```
Overall Result : ERROR
```

This error happen because data in your order import template isn't correct. For example, ProductNumber contains AS54888777 but the Oracle database contains AS54888. Use SQL to query the Oracle database and identify the correct data to import.

For another example.

```
Request Id: 65699
Batch Name: 10810
Source System: null
Order Number: null
Allow Auto Purge: Y
Order PMC_IMP_161208_001 will not be imported due to the following error: An order was not created because
 the value 300,000,001,469,001,777 provided for the attribute SOLD_TO_PARTY_ID is invalid for the source
 order with the following details: source order PMC_IMP_161208_001. Check the attribute value, and resubmit
 the order..
Number of orders that were imported: 0. Check the run time user interface for processing status.
Number of orders that did not pass validation: 1. Some of these orders may be available in the order entry
 user interface.
Overall Result : ERROR
```

This error happen because the SOLD_TO_PARTY_ID attribute contains 300,000,001,469,001,777 but the Oracle database contains 300000001469001. Use SQL to query the Oracle database and identify the correct data to import.

Configured Items

See *Troubleshoot Problems with Configure-to-Order*.

Cross-References

I encounter an error on the Manage Order Orchestration Messages page.

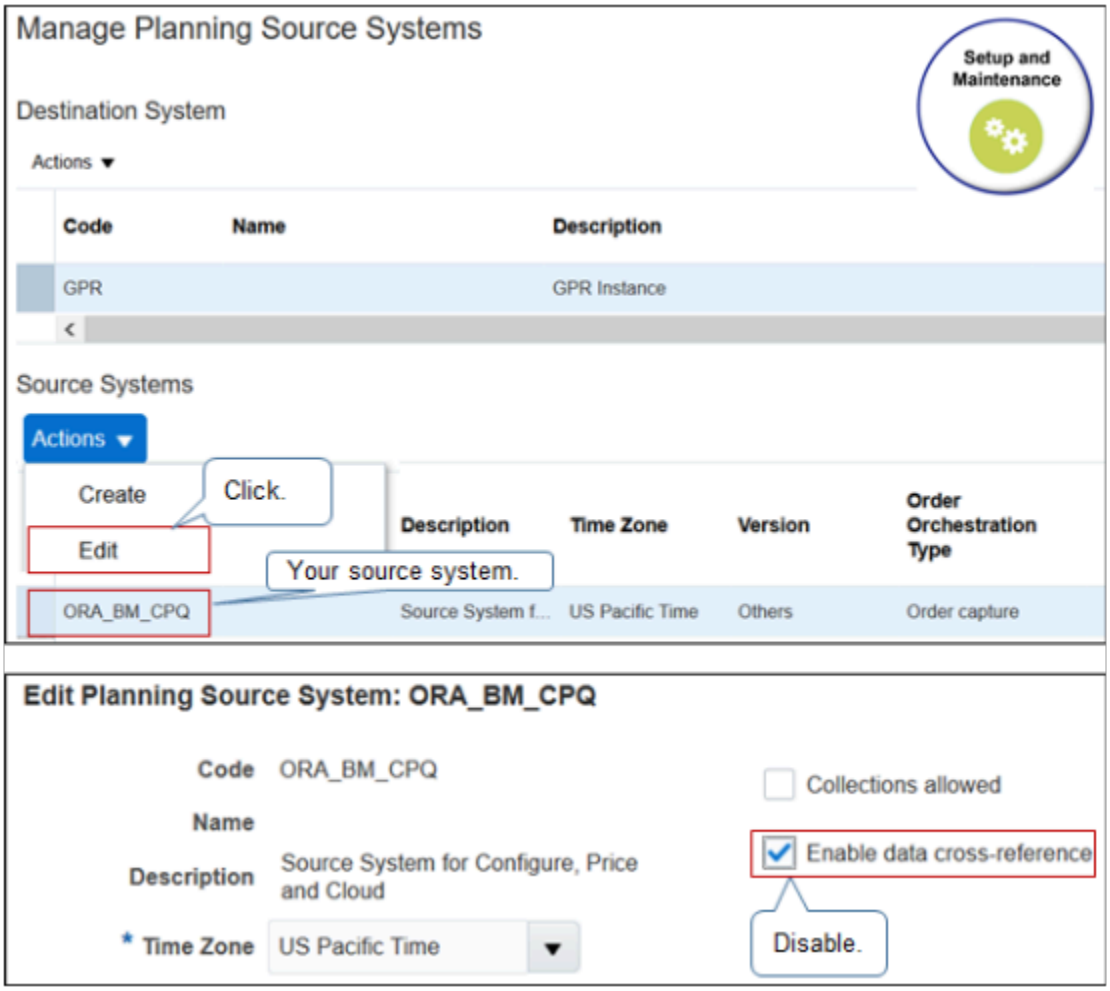
```
A cross-referenced value was not found for attribute CURRENCY_CODE in source system XXXXXXX
```

You might also encounter this error in the response to your import payload.

```
<bpelFault>
<faultType>1</faultType>
<processResponse xmlns="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/DooDecompReceiveOrderComposite">
<part name="payload">
<processResponse xmlns:client="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/DooDecompReceiveOrderComposite" xmlns="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/DooDecompReceiveOrderComposite">
<client:OrchestrationOrderResponse>
<ns4:SourceTransactionNumber xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/model/">HITECH0001</ns4:SourceTransactionNumber>
<ns4:SourceTransactionIdentifier xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/model/">36466323</ns4:SourceTransactionIdentifier>
<ns4:SourceTransactionSystem xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/model/">ORA_BM_CPQ</ns4:SourceTransactionSystem>
<ns4:ReturnStatus xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/model/">ERROR</ns4:ReturnStatus>
<ns4:MessageName xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/model/">env:Server</ns4:MessageName>
<ns4:MessageDescription xmlns:ns4="http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/
receiveSalesOrder/model/">A cross-referenced value was not found for attribute CURRENCY_CODE in source
 system ORA_BM_CPQ.</ns4:MessageDescription>
</client:OrchestrationOrderResponse>
</processResponse>
</part>
</processResponse>
</bpelFault>
```

This problem happens because the source system, such as CPQ, doesn't require cross references but the Enable Data Cross-Reference option is enabled on the Manage Planning Source Systems page in the Setup and Maintenance work area.

To fix the problem, disable the Enable Data Cross-Reference option.



I encounter an error on the Manage Order Orchestration Messages page.

```
Cross-Referenced Value Not Found for UOM_CODE
```

My import payload includes:

```
<ns2:OrderedQuantity>1</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>EA</ns2:OrderedUOMCode>
```

Two reasons might cause this problem.

- You didn't collect data from the source system for Global Order Promising.
- Your import payload doesn't include the correct UOM code.

First, in the Setup and Maintenance work area, open the Review Collected Order Reference Data page, click **Units of Measure**, then search for the unit of measure that your import payload references. If you can't find it, then you must collect data. For details, see *Quick Start for Setting Up Order-to-Cash*.

Next, run SQL against the Order Management database.

```
select
 mai.instance_id,
 mai.instance_code,
 muom.unit_of_measure,
 muom.uom_code
from
 msc_units_of_measure MUOM,
 msc_apps_instances MAI
where
MAI.instance_id = MUOM.SR_Instance_id;
```

This SQL determines whether the inventory and data collection tables contain the UOM you're importing.

Assume the query returns:

INSTANCE_ID	INSTANCE_CODE	UNIT_OF_MEASURE	UOM_CODE
300000000120910	OPS	EA	AbC

Next, run SQL to get the unit of measure that the UOM code references.

```
SELECT uom_code,
 uom_class,
 base_uom_flag,
 unit_of_Measure,
 description
FROM inv_units_of_measure_B iuomb,
 inv_units_of_measure_tl iuomtl
WHERE iuomb.unit_of_measure_id = iuomtl.unit_of_measure_id
and iuomtl.language = 'US'
ORDER BY UOM_code;
```

Assume the query returns:

UOM_CODE	UOM_CLASS	BASE_UOM_FLAG	UNIT_OF_MEASURE	DESCRIPTION
AbC	1	Y	EA	EACH

Modify your input payload so it uses the UOM_CODE that the query returns. For example, replace this:

```
<ns2:OrderedQuantity>1</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>EA</ns2:OrderedUOMCode>
```

With this:

```
<ns2:OrderedQuantity>1</ns2:OrderedQuantity>
<ns2:OrderedUOMCode>AbC</ns2:OrderedUOMCode>
```

As an alternative, modify your input payload or so it uses the OrderedUOM attribute instead of OrderedUOMCode, and references the value from UNIT_OF_MEASURE instead UOM_CODE.

```
<ns2:OrderedQuantity>1</ns2:OrderedQuantity>
<ns2:OrderedUOM>EA</ns2:OrderedUOM>
```

Revisions and Returns

Problem	Solution
I import a referenced return order, submit it, but Receivables doesn't create a credit memo for the return.	Make sure the business unit on the order line that you import is the same as the business unit on the original order line.
I encounter an error message while importing an order revision. `Order management didn't import source order 3502701192-10 because of the following error: A sales order was not created because source order 3502701192-10 doesn't include a value for required attribute BILL_TO_CUSTOMER_ID, on source order line 1, with source order schedule 1. Provide a value for this attribute, and then submit the source order again.`	If you import an order revision, then you must include the Source Transaction Line Identifier and Source Transaction Schedule Identifier on the DOO_ORDER_ADDRESSES_INT tab of your import worksheet.
I include a value in the OriginalSourceOrderNumber attribute but the import fails.	If you include a value for the OriginalSourceOrderNumber attribute, then you must also include a value in the OriginalSourceLineNumber attribute in the originalOrderReference entity. You must also include order line details. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

I Can't Get Specific Versions of Sales Orders

To get the latest version of a sales order through REST API, you can use `/salesOrdersForOrderHub/{OrderKey}`, where `{OrderKey}` is *sourceOrderSystem*:*sourceOrderId*.

However, if the order is in draft status, then the draft becomes the latest version of the order. If you have a draft revision, then you can't use the order's URL to get the sales order that you submitted. Instead, you must use the `q` parameter or one of the finders to return all versions of the order, and you can't use a GET request on the order header or order line to get extensible flexfields. Instead, you must do a GET on the additionalInformation child resource, and you must use that resource's URL with an `expand=all` parameter to expand the flexfield contexts.

As an alternative, you can use /salesOrdersForOrderHub/*headerId* to get to a specific version of the order. For example, use salesOrdersForOrderHub/57684 to get sales order with headerId 57684.

There Are Gaps in My Order Lines

There are gaps in my order lines when I import a revision or a return.

For example, I import a return order that has two referenced return lines, line 1 and line 2. Some time later, I import a revision to this return. The revision adds another line to the return, but the Manage Orders page displays this line as line number 4. I expected it to display line 3, not 4.

Assume you run an SQL to get details about your order.

```
SELECT dha.source_order_number ,
 dfla.source_line_number ,
 dfla.fulfill_line_number,
 dfla.status_code ,
 dla.line_number ,
```

```
 dla.display_line_number
FROM fusion.doo_headers_all dha ,
 fusion.doo_fulfill_lines_all dfla,
 fusion.doo_lines_all dla
WHERE dha.header_id = dfla.header_id
 AND dla.line_id = dfla.line_id
 AND dha.submitted_flag = 'Y'
 AND dfla.source_order_number = '&SOURCE_ORDER_NUMBER'
ORDER BY dla.line_number
```

Assume the SQL returns these values.

SOURCE_ORDER_NUMBER	HEADER_ID	SOURCE_LINE_NUMBER	FULFILL_LINE_ID	FULFILL_LINE_NUMBER	LINE_NUMBER	DISPLAY_LINE_NUMBER
12323	300000100663978	1	300000100663987	1	1	1
12323	300000100663978	2	300000100664018	1	2	2
12323	300000100663978	3	300000100664052	1	3	3
12323	300000100663978	4	300000100670796	1	6	4
12323	300000100663978	5	300000100670811	1	7	5
12323	300000100663978	8	300000100659060	1	9	6

Notice that SOURCE_LINE_NUMBER has a gap between 5 and 8, and LINE_NUMBER has a gap between 3 and 6, and 7 and 9. Its possible that you used SOURCE_LINE_NUMBER or LINE_NUMBER in your import payload. Some import technologies, such as OrderInformationService, include these attributes in their response payload but you must not use them in your import payload when you import a revision or return.

If you're importing a revision or return and you don't want a gap in the order line number or fulfillment line number, then include a value for FULFILL_LINE_ID but not SOURCE_LINE_NUMBER in your payload. The Order Management work area will use the DISPLAY_LINE_NUMBER as the value to display on the Manage Orders page and other fulfillment views.

Other Problems

Problem	Solution
I use OrderImportService to import customer data, including the Contact attribute and the ContactEmail attribute. I use the Customer Data Management page and verify that the import successfully imported contact and contact email. I prefer to manage the contact and contact email for each account on the Manage	Manage Customers doesn't support managing contact and contact email for each account.

Problem	Solution
Customers page, but I can't view these attributes on this page.	
It takes a long time to create the CSV file when I do file-based data import. For example, every day I import over 300 sales orders from my source system into Order Management. I can load the header details in the first worksheet with no problems, but after I after import about 1,900 lines from the second worksheet, DOO_ORDER_LINES_ALL_INT, the server performance declines to 15 to 20 seconds delay for each line. I wait for over an hour but its still processing the lines.	Try this on the computer you're using when you create the CSV file. • Install the latest version of **`Microsoft Visual C++ Redistributable Packages`**. • Disable hardware acceleration in Excel. • Disable Excel add-ins that you don't need.
I use the Source Transaction Line Identifier attribute on the DOO_ORDER_LINES_ALL_INT worksheet when I import through file-based data import, then notice that I have sales orders where source line numbers are 20, 21, and 22, but the line numbers are 1, 2, and 3. Is there way to make the line numbers use the same values that I set in Source Transaction Line Identifier?	Source Transaction Line Identifier and the line number don't store the same data. Source Transaction Line Identifier identifies a transaction line in your source system. LINE_NUMBER stores a number for the order line in Order Management. Order Management automatically creates the line number and stores it in the LINE_NUMBER column of the DOO_LINES_ALL table in the Oracle database. Order Management does this to make sure each line number is unique in the sales order. You can't modify this value.
The import reports only one error at a time. For example: • I import 100 order lines. • There's a problem on 10 different lines, starting with the fifth line that I import. • The import stops processing at line 5 and doesn't continue processing the other 95 lines. • I have to run the import successively and troubleshoot each of the other 9 lines that are in error separately. • I expected the import to process all lines and then provide 10 separate error messages, one for each line that's in error.	This is expected behavior. The import stops at the first error it encounters so you can fix that error. Sometimes an error on one line affects an error on another line. So its important to fix each error before continuing to the next error.
I want to use REST API to import a cancel on an order line. I want to view the import in the Order Management work area in draft status, and then manually submit it, but Order Management automatically submits it before I have a chance to do that.	Set the SubmittedFlag attribute to false on the header entity in your REST API payload. This will prevent Order Management from automatically submitting the order. For details, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Sales Orders for Order Hub**.

Related Topics

- Overview of Importing Orders Into Order Management
- Roadmap for Setting Up Order-to-Cash
- Set Up User Roles and Privileges in Order Management
- Use SQL to Query Order Management Data

Compare Import Data to Oracle Database Data

Query the Oracle database to determine whether your import data meets database requirements.

Assume you run the ProcessOrderRequest operation on the ReceiveOrderRequestService web service at https://server:port/soa-infra/services/default/DooDecompReceiveOrderExternalComposite/ReceiveOrderRequestService.

Assume the header section of the payload includes this code.

```
<ns1:OrchestrationOrderRequest>
<ns2:BuyingPartyName>XXXXXXXXXX</ns2:BuyingPartyName>
<ns2:BuyingPartyId>7777777777</ns2:BuyingPartyId>
```

And the order line section of the payload includes this code.

```
<ns2:OrchestrationOrderRequestLine>
<ns2:BillToCustomerName>ZZZZZZZZZZ</ns2:BillToCustomerName>
<ns2:BillToCustomerIdentifier>8888888888</ns2:BillToCustomerIdentifier>
<ns2:BillToAccountSiteUseIdentifier>9999999999</ns2:BillToAccountSiteUseIdentifier>
```

Run an SQL query to determine whether these values are correct. For details, see *Use SQL to Query Order Management Data*.

Here's how you can map.

Attribute in Payload	SQL
BuyingPartyName	PARTY_NAME
BuyingPartyId	PARTY_ID
BillToCustomerName	PARTY_NAME
BillToCustomerIdentifier	CUST_ACCOUNT_ID
BillToAccountSiteUseIdentifier	ORIG_SYSTEM_REFERENCE

Error with SOLD_TO_PARTY_ID

Assume you encounter an error.

```
An order was not created because a value was not provided for the required attribute SOLD_TO_PARTY_ID in
 the source order with the following details: source order 12345. Provide a value for SOLD_TO_PARTY_ID, and
 resubmit the order.
```

Run SQL to identify the correct data you should use.

```
SELECT HZ.PARTY_ID ,
 CUSTOMERACCOUNTSITEPEO.CUST_ACCOUNT_ID ,
 HZ.PARTY_NAME ,
 lOCATIONPEO.ADDRESS_STYLE ,
 lOCATIONPEO.COUNTRY ||','|| LOCATIONPEO.ADDRESS1 ||','|| LOCATIONPEO.CITY ||','|| LOCATIONPEO.POSTAL_CODE
 LOCATION ,
 CUSTOMERACCOUNTSITEPEO.STATUS "Account Site Status" ,
 CUSTOMERACCOUNTSITEPEO.BILL_TO_FLAG ,
 CUSTOMERACCOUNTSITEPEO.SHIP_TO_FLAG ,
 CUSTOMERACCOUNTSITEUSEPEO.SITE_USE_CODE ,
 CUSTOMERACCOUNTSITEUSEPEO.PRIMARY_FLAG ,
 CUSTOMERACCOUNTSITEUSEPEO.STATUS "Cust_Site_Use Status" ,
 CUSTOMERACCOUNTSITEUSEPEO.ORIG_SYSTEM_REFERENCE ,
 CUSTOMERACCOUNTSITEUSEPEO.CUST_ACCT_SITE_ID ,
 CUSTOMERACCOUNTSITEPEO.SET_ID
FROM fusion.HZ_PARTIES HZ ,
 fusion.HZ_PARTY_SITES PartySitePEO ,
 fusion.HZ_LOCATIONS LocationPEO ,
 fusion.HZ_CUST_ACCT_SITES_ALL CustomerAccountSitePEO ,
 fusion.HZ_CUST_SITE_USES_ALL CustomerAccountSiteUsePEO
WHERE HZ.PARTY_ID = PARTYSITEPEO.PARTY_ID
 AND
 (PARTYSITEPEO.LOCATION_ID = LOCATIONPEO.LOCATION_ID)
 AND
 (PARTYSITEPEO.PARTY_SITE_ID = CUSTOMERACCOUNTSITEPEO.PARTY_SITE_ID)
 AND
 (CUSTOMERACCOUNTSITEPEO.CUST_ACCT_SITE_ID = CUSTOMERACCOUNTSITEUSEPEO.CUST_ACCT_SITE_ID)
 and upper(HZ.PARTY_NAME) like upper('&CUSTOMER_NAME%')
```

Error with BILL_TO

Assume you encounter an error.

```
An order was not created because a value was not provided for the required attribute BILL_TO_CUSTOMER_ID
 in the source order with the following details: source order XXXXXXXXX, source order line 1, source order
 schedule 1. Provide a value for BILL_TO_CUSTOMER_ID, and resubmit the order.
```

Or this error.

```
An order was not created because a value was not provided for the required attribute BILL_TO_SITE_USE_ID
 in the source order with the following details: source order XXXXXXXXX, source order line 1, source order
 schedule 1. Provide a value for BILL_TO_SITE_USE_ID, and resubmit the order.
```

Then run SQL to identify the correct data you should use.

```
SELECT accounts.party_id ,
 accounts.account_name ,
 accounts.account_number ,
 party_sites.party_site_name ,
 sites.cust_acct_site_id ,
 sites.cust_account_id ,
 site_uses.site_use_code ,
 site_uses.primary_flag ,
 site_uses.site_use_id ,
 site_uses.location ,
 locations.ADDRESS1 || ' ' || locations.ADDRESS2 || ' ' || locations.ADDRESS3 || ' ' || locations.ADDRESS4
 || ' ' || locations.CITY || ' ' || locations.POSTAL_CODE || ' ' || locations.STATE ||
 ' ' || locations.country ,
 territory.territory_short_name
FROM fusion.hz_cust_accounts accounts ,
 fusion.hz_cust_acct_sites_all sites ,
 fusion.hz_party_sites party_sites ,
 fusion.hz_cust_site_uses_all site_uses ,
 fusion.hz_locations locations ,
 fusion.fnd_territories_vl territory
```

```
WHERE sites.cust_account_id = accounts.cust_account_id
 AND party_sites.party_site_id = sites.party_site_id
 AND site_uses.cust_acct_site_id = sites.cust_acct_site_id
 AND locations.location_id = party_sites.location_id
 -- AND site_uses.site_use_code = 'BILL_TO'
 -- AND site_uses.primary_flag = 'Y'
 AND site_uses.STATUS = 'A'
 AND accounts.STATUS = 'A'
 AND sites.STATUS = 'A'
 AND locations.COUNTRY = territory.territory_code
 AND EXISTS
 (
 SELECT 1
 FROM fusion.fnd_setid_assignments
 WHERE set_id = sites.set_id
 AND reference_group_name = 'HZ_CUSTOMER_ACCOUNT_SITE'
 AND determinant_type = 'BU'
 )
 and upper(accounts.account_name ) like upper('&CUSTOMER_NAME%')
```

Example SQL Output

Assume SQL returns results.

PARTY_ID	ACCOUNT_ID	PARTY_NAME	SITE_USE_CODE	ORIG_SYSTEM_REFERENCE
300000001469001	300000001469002	Computer Service and Rentals	SHIP_TO	300000001469006
300000001469001	300000001469002	Computer Service and Rentals	BILL_TO	300000001469016
300000001469001	300000001469002	Computer Service and Rentals	BILL_TO	300000002494053
300000001469001	300000001469002	Computer Service and Rentals	BILL_TO	300000002494059
300000001469001	300000001469002	Computer Service and Rentals	SHIP_TO	300000002494060

Correct the Problem

Here's the mapping you can use to correct the problem.

Attribute in Payload	Value from SQL
BuyingPartyName	Computer Service and Rentals
BuyingPartyId	300000001469001
BillToCustomerName	Computer Service and Rentals

Attribute in Payload	Value from SQL
BillToCustomerIdentifier	300000001469002
BillToAccountSiteUseIdentifier	300000001469016

Here's the corrected payload for the order header:

```
<ns1:OrchestrationOrderRequest>
<ns2:BuyingPartyName>Computer Service and Rentals</ns2:BuyingPartyName>
<ns2:BuyingPartyId>300000001469001</ns2:BuyingPartyId>
```

Here's the corrected payload for the order line:

```
<ns2:OrchestrationOrderRequestLine>
<ns2:BillToCustomerName>Computer Service and Rentals</ns2:BillToCustomerName>
<ns2:BillToCustomerIdentifier>300000001469002</ns2:BillToCustomerIdentifier>
<ns2:BillToAccountSiteUseIdentifier>300000001469016</ns2:BillToAccountSiteUseIdentifier>
```

Related Topics

- Roadmap for Setting Up Order-to-Cash
- Overview of Importing Orders Into Order Management
- How Order-to-Cash Works with Order Capture Systems

Transform

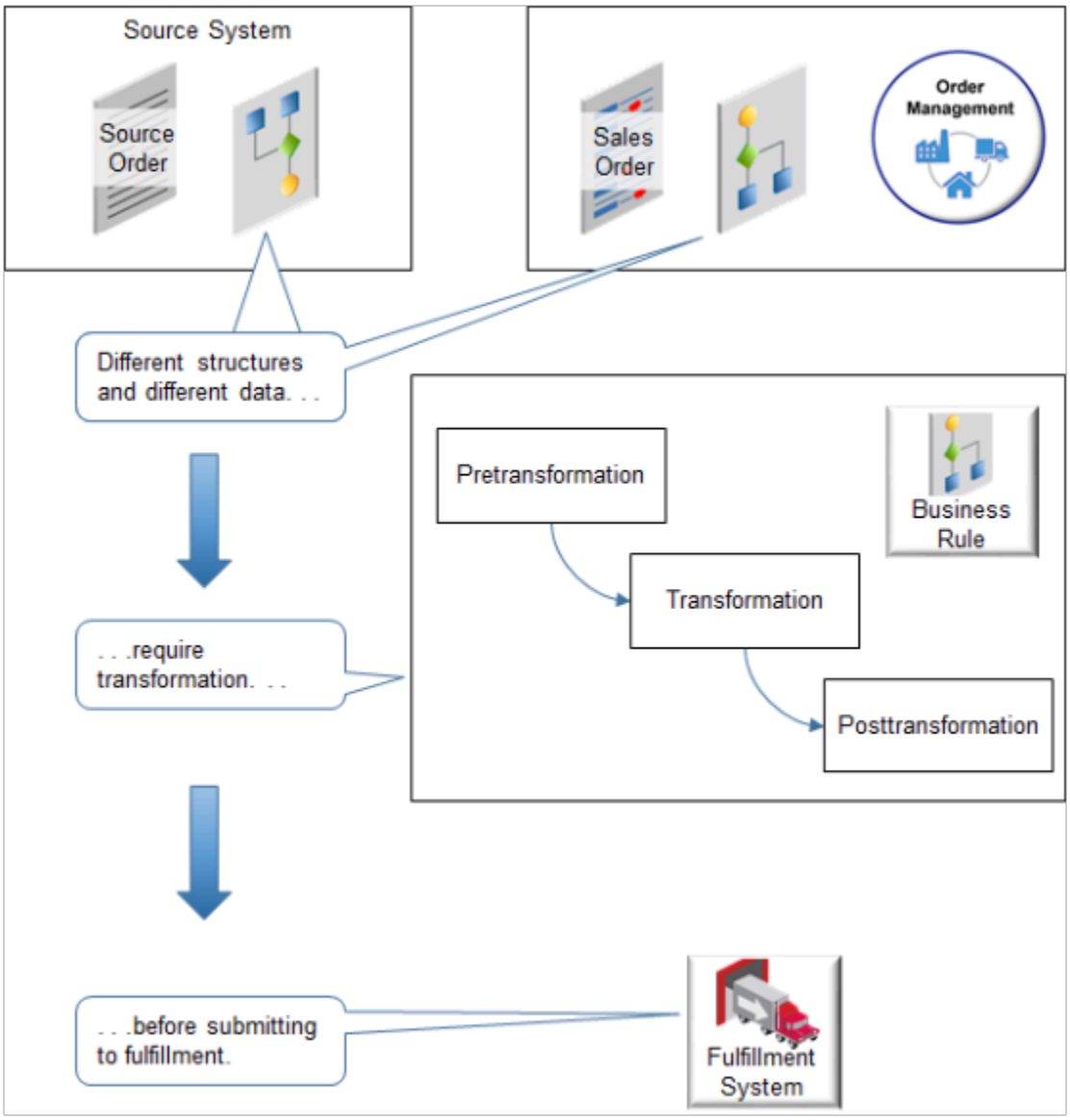
Overview

Overview of Transformation Rules

Write a business rule that populates order attributes before Order Management transforms a source order, while it transforms a source order, and after it transforms a source order.

A source system is a system in your implementation that captures the details of a sales order. The source orders that your source system captures probably use attributes, structures, and data that are different from what Order Management uses. If you use more than one source system, then the details across source systems might be different.

Order Management uses a specific structure in sales orders so it can fulfill each of them consistently and effectively. Transformation rules transform each source order into a structure that Order Management can use, such as the arrangement of attributes on the order header, order lines, and fulfillment lines, and the relationships between header, order lines, and fulfillment lines.



When Do Rules Run?

Order Management runs transformation rules differently depending on where you manage your sales orders.

Manage Sales Orders in the Order Management Work Area

What You Do	What Rules Run
Create a sales order or copy a sales order.	Pretransformation rules run each time that you: • Modify an attribute that contains customer details. • Modify the order type. • Add an order line.
Copy a sales order and don't:	Pretransformation rules don't run.

What You Do	What Rules Run
• Modify an attribute that contains customer details. • Modify the order type. • Add an order line.	
Create a sales order and click Submit.	Transformation rules and posttransformation rules run one time after you click Submit.
Copy a sales order or create a return, then click Submit.	Transformation rules and posttransformation rules don't run on return lines after you click Submit.
Revise a sales order.	Transformation rules don't run.
Create a return and: • Modify attributes that contain customer details. • Modify the order type. • Add an order line.	Pretransformation rules run.

Order Management doesn't run transformation rules or posttransformation rules with returns. If you need to do transformation or posttransformation on a return line, then create an order management extension instead. For details, see *Overview of Creating Order Management Extensions*.

Import a Source Order

If you import a source order, then Order Management:

- Runs transformation and posttransformation rules when it validates the draft sales order during import. It runs transformation rules and posttransformation rules only on order lines and not on return lines.
- Doesn't run transformation rules on orders that you import through the Sales Orders for Order Hub REST API.

  For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

If you import a source order through a SOAP web service, then Order Management runs rules in a sequence. You can't modify this sequence:

1. Pretransformation rules.
2. Transformation rules.
3. Posttransformation rules.

Related Topics

- Set Up Transformation
- Create Transformation Rules
- Use Groups to Manage and Control Sales Orders
- Manage Pretransformation Rules
- How Order Management Transforms Source Orders Into Sales Orders

Pretransformation Rules

Use a pretransformation rule to populate an order attribute or set its default value before Order Management transforms the source order.

Example Usage	Description
Populate an attribute on a fulfillment line.	Assume your sales order includes the Request Date attribute, and you must display it on fulfillment lines, but only for the AS54888 desktop computer. Here's your rule. • If the item is AS54888 desktop computer, then display the Request Date attribute on the fulfillment line.
Populate an attribute so you can use it in a transformation rule.	Assume your sales order includes the AS54888, you must convert its size from centimeters to inches, then display it on the fulfillment line. Here's your rule. • If the item is a AS54888, then convert the Size attribute and display it on the fulfillment line.

Order Management runs pretransformation rules each time the Order Entry Specialist modifies the sales order.

Learn how to use a pretransformation rule to set default values. For details, see *Set Default Values for Other Attributes on Sales Orders*.

Set Transaction Type Before You Transform

You can run a pretransformation rule only on some events, such as when you select the business unit, customer, or order type, or when you add an order line.

Also, pretransformation and posttransformation rules might not run when you create a sales order through a web service, REST API, or file-based data import.

If you must set the transaction type before you transform, and if you must set it when an event happens that doesn't start a pretransformation rule, or if you create a sales order through a web service, REST API, or file-based data import, then don't use a transformation rule to set the attribute values. Use an order management extension instead. Read the Set the Billing Transaction Type According to Order Type subtopic. For details, see *Extend Order Headers*.

If you're returning a sales order, then you must set the transaction type to Credit Memo.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Related Topics

- Set Up Transformation
- Create Transformation Rules
- Use Groups to Manage and Control Sales Orders
- Manage Pretransformation Rules
- How Order Management Transforms Source Orders Into Sales Orders

Transformation Rules

Use a transformation rule to transform an item, such as the AS54888 Desktop Computer.

Order Management uses item relationships, item structures, transactional attributes, and business rules to transform a sales representation of the item in the source order to a fulfillment representation of the item on fulfillment lines.

- You use the Manage Product Transformation Rules page in the Setup and Maintenance work area to write the transformation rule. You can also use the Product Information Management work area to set up item relationships, item structures, and transactional attributes.
- You can't use a transformation rule to add a product model to a sales order.
- An order line that a transformation rule creates gets most attributes from the line that the rule uses to create the new line. You can't edit, revise, or price an order line that a transformation rule creates.

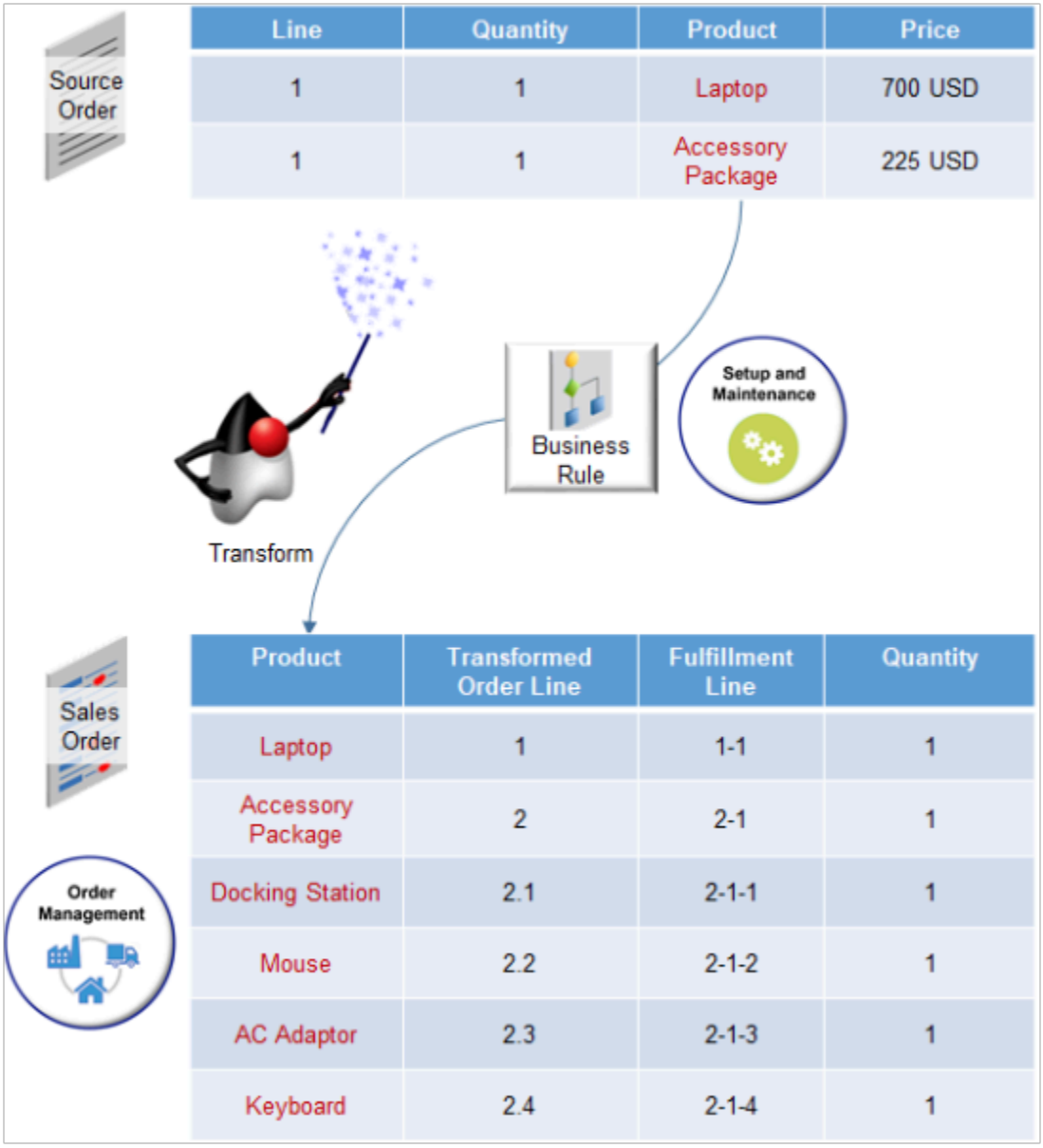
  For example, assume you sell a laptop computer that ships to different geographical regions. Each region requires a different electrical adapter, such as 110 volts or 240 volts. You create a rule that uses the geographical region where you ship the item to determine the adapter to include in the sales order. You create a transformation rule that transforms the source order to a sales order that includes order line a and order line b, then adds the adapter to order line b. Assume the Ship To address on order line a is 550 Vision Way. The rule sets Ship To address on order line b to 550 Vision Way. You can't change this value on order line b.

  The Create Order page and the Edit Order page in the Order Management work area don't display an order line that a transformation rule creates because transformation adds the line only after your user clicks Submit. You can view the new line in a fulfillment view, such as View Order. If you create an order revision, you can also view the new line on the Create Revision or Edit Revision page.

Product-to-Product Transformation

Transform a single item to one or more items according to item structure, item relationship, and the transformation rule. You can write a rule that transforms an item to another item, and that creates individual fulfillment lines.

Assume your source order includes a laptop that comes with an accessory package that includes more than one item, such as a docking station, mouse, and so on. You write a product-to-product transformation rule that transforms the source order into a sales order that includes individual fulfillment lines for the laptop and each item in the accessory package.



Here's how it works.

- Transform two source order lines into six fulfillment lines.
- Use the price from the source order to populate the price in the sales order.
- Transform the line for the laptop in the source order to fulfillment line 1 in the sales order.
- Transform the accessory package in the source order to more than one fulfillment line in the sales order. Each line represents part of the content of the accessory package, such as one line for the docking station, one line for the mouse, and so on.

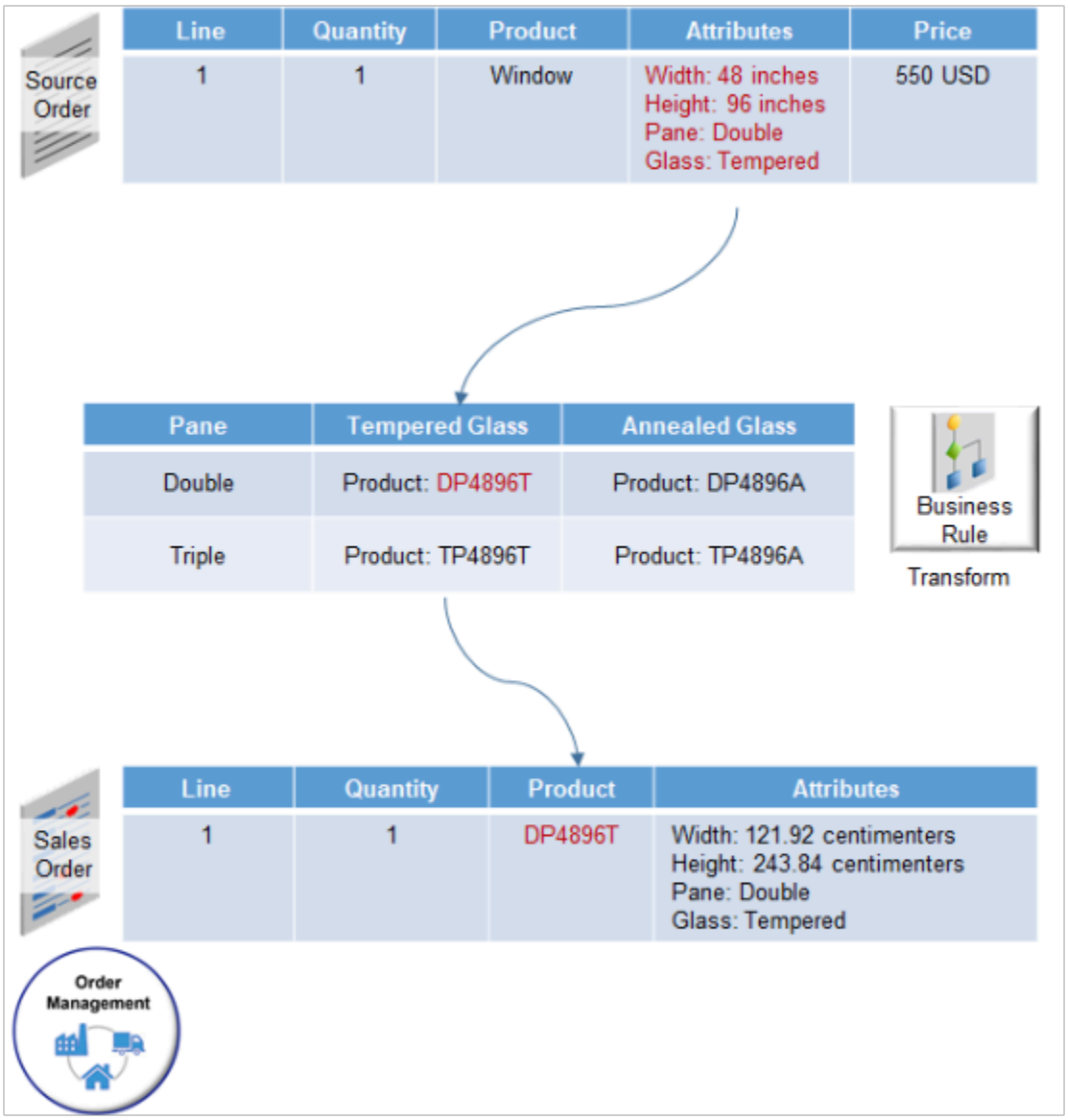
Product-to-Attribute Transformation

Transform the item in a source order according to the attributes of another item.

Assume you sell the Window item. You can write a rule that uses these attributes in the source order to get the number for an item that uses the same dimensions, but that uses double-pane, tempered glass:

- Width
- Height
- Pane
- Glass

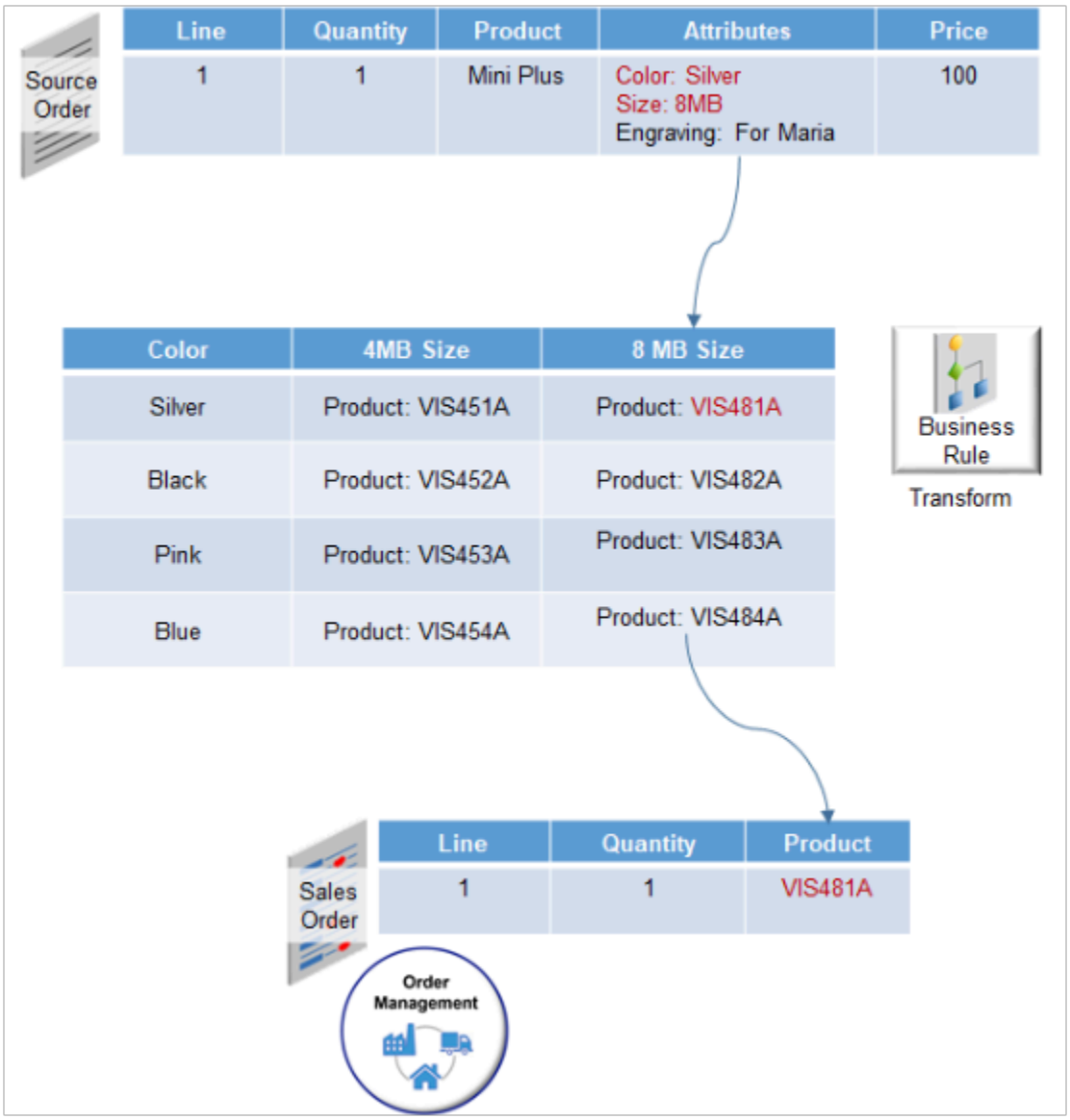
The transformed order will include the item number for the combination of attributes.



Attribute-to-Product Transformation

Use attributes to transform an attribute to an item number. Add the transformation to an item that already exists or replace the item in the source order.

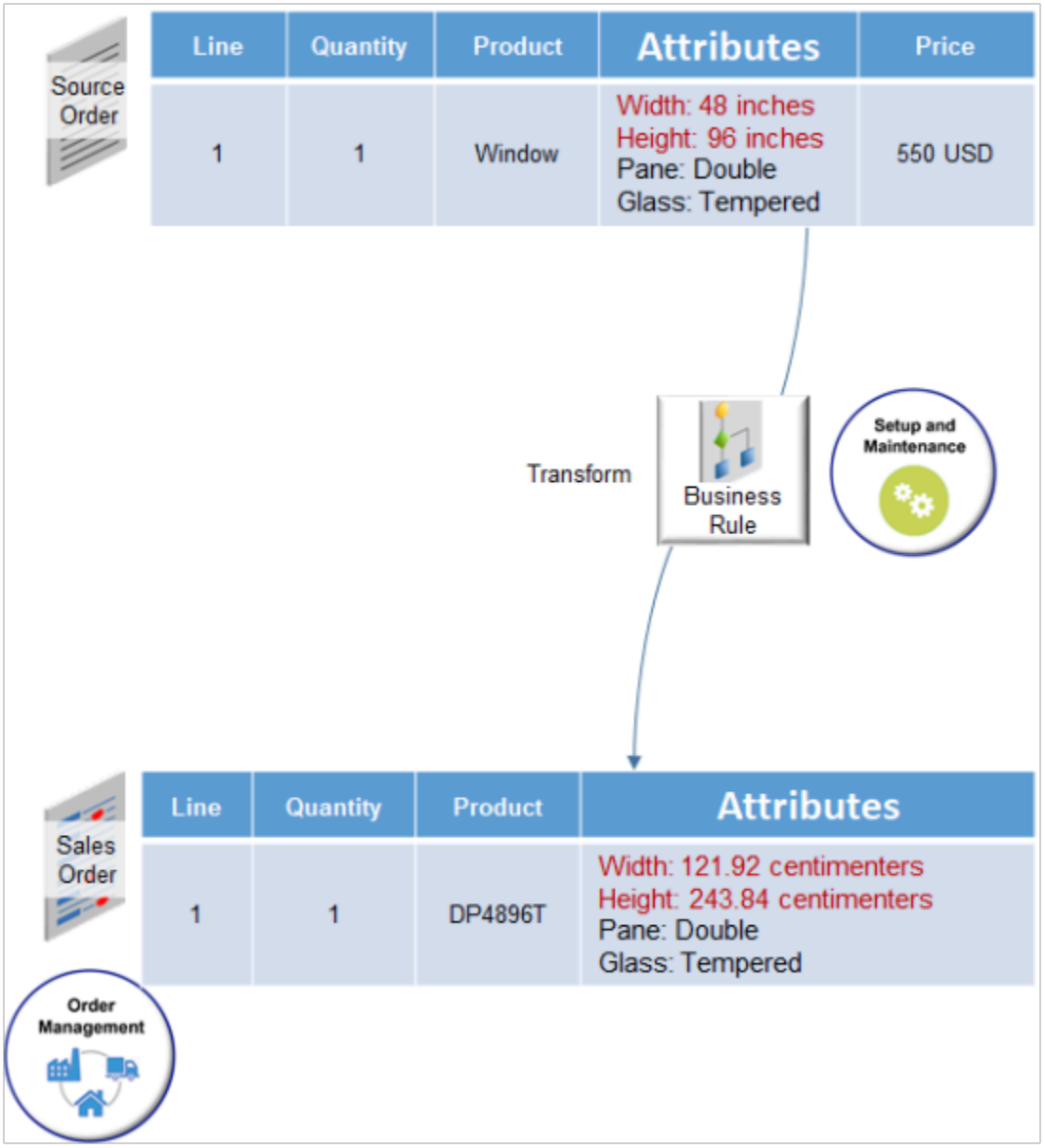
Assume you sell an MP3 player that includes the Color attribute and Size attribute, and that you must use a combination of them to reference an item number. You can create a rule that transforms Color and Size of item Mini Plus to item VIS481A.



Attribute-to-Attribute Transformation

Transform the value of an order line attribute in a source order to a different order line attribute in a sales order.

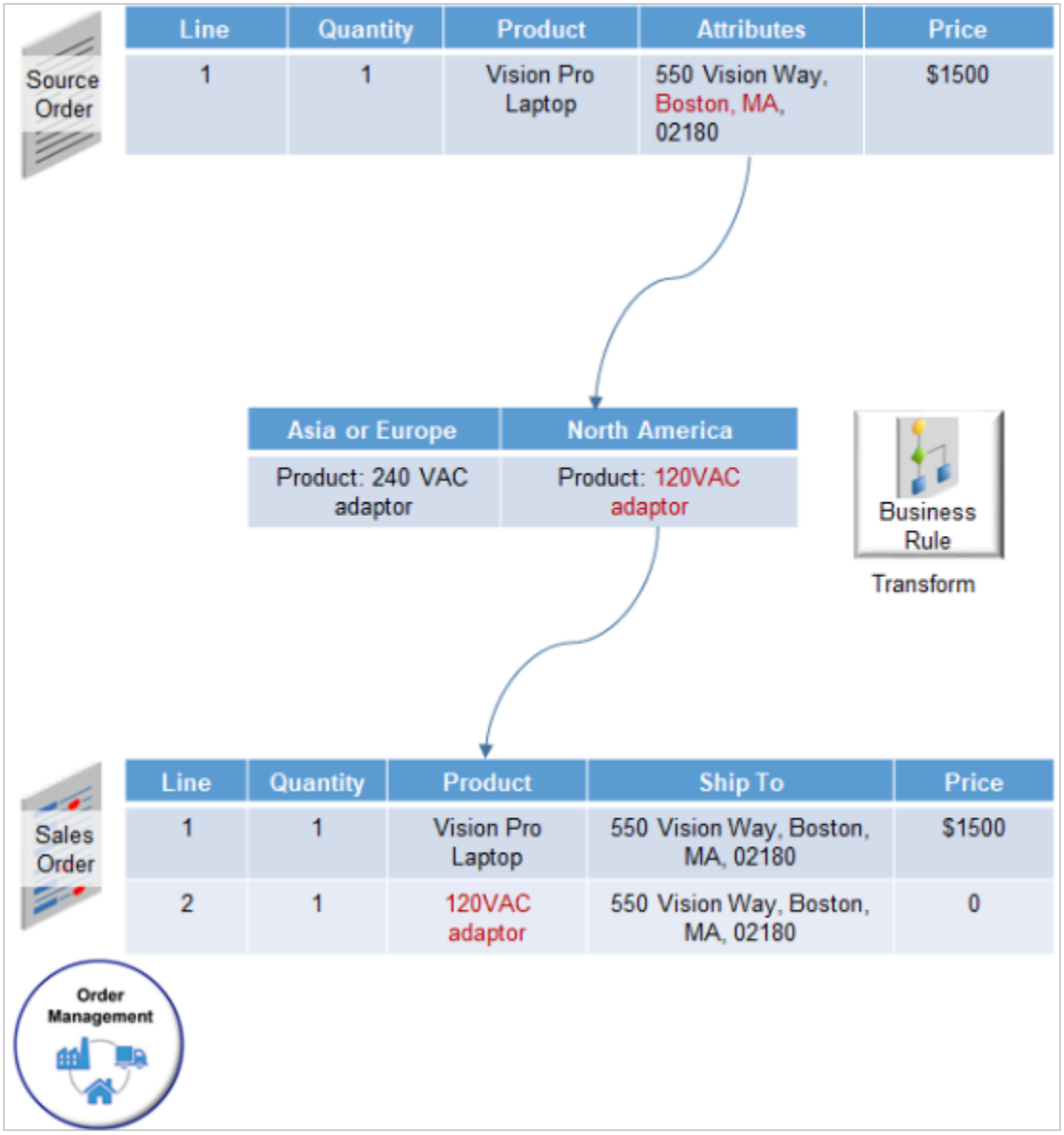
Assume your company resides in Europe, you receive orders from an office in the United States that measures the item size in inches, but you must display the size in centimeters. Create a rule that transforms the width and height from inches on the source order to centimeters on the sales order.



Context-to-Product Transformation

Use the context of the source order to determine the item in the sales order.

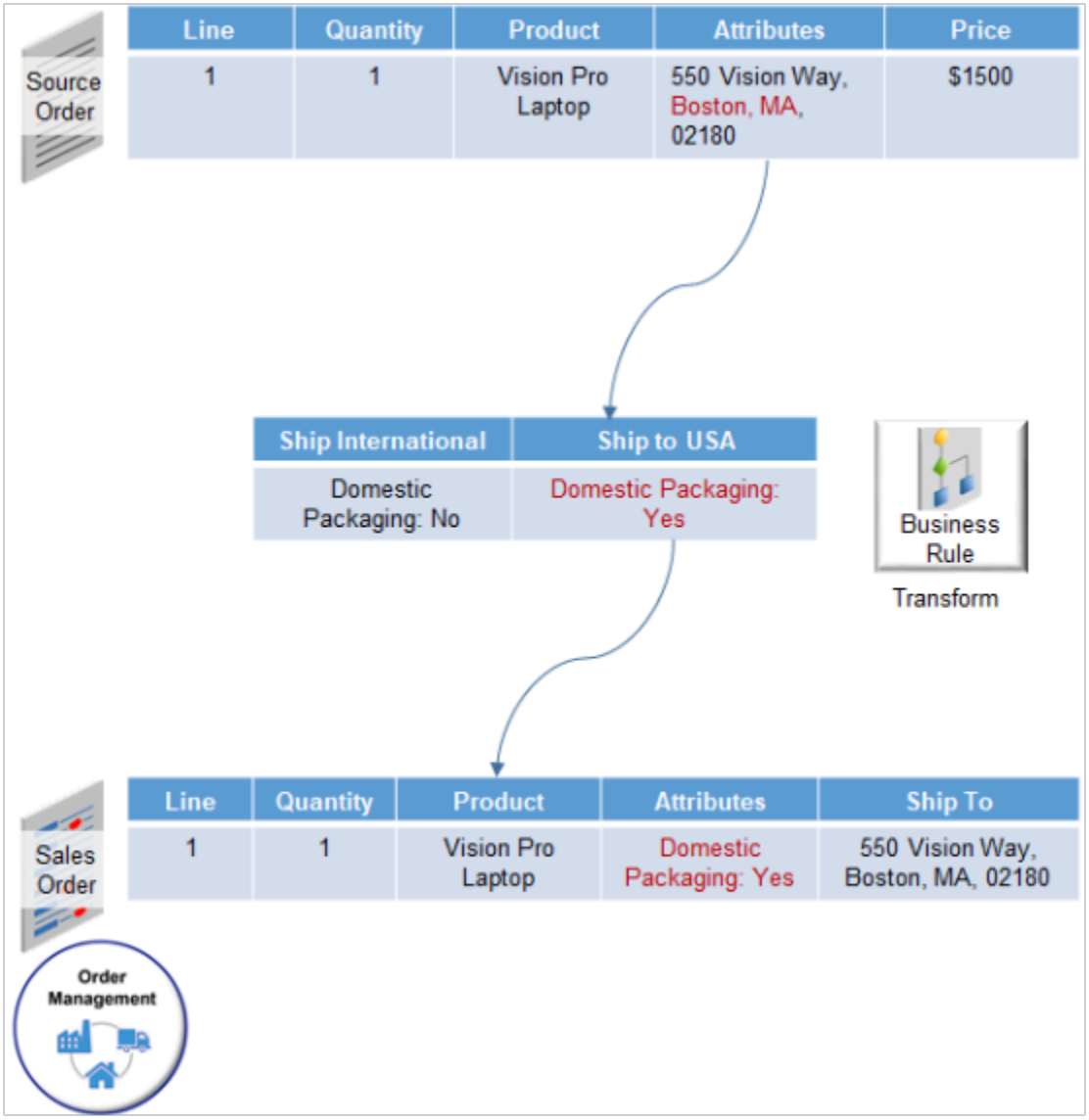
Assume you sell laptop computers that ship to different geographical regions. Each region requires a different electrical adapter, such as 110 volts or 240 volts. Create a rule that uses the geographical region where you ship the item to determine the adapter to include in the sales order, transforms the source order to a sales order that includes two order lines, and then adds the adapter to one of these lines.



Context-to-Attribute Transformation

Transform the source order context to an attribute.

Assume you ship some laptop computers to domestic locations in the USA and others to international locations in other countries. The destination requires different packaging. So you create a Domestic Packaging flexfield to store the details. Assume an attribute in the source order includes the domestic address, so the context is domestic, and the transformation rule sets the Domestic Packaging attribute to Yes.



Related Topics

- Set Up Transformation
- Create Transformation Rules
- Use Groups to Manage and Control Sales Orders
- Manage Pretransformation Rules
- How Order Management Transforms Source Orders Into Sales Orders

Posttransformation Rule

Use a posttransformation rule to populate an order attribute after Order Management transforms the item.

Example Usage	Description
Populate order lines so they reference different warehouses.	Create a rule that transforms the source order into a sales order that includes two lines. • Order line 1 for the laptop computer • Order line 2 for an AC adapter Write a rule that populates order line 2 so it sets the Warehouse attribute to a value that's different from the warehouse that supplies the laptop computer on line 1.
Populate an order with a new attribute.	Assume your source order uses the MM/DD/YYYY format for requested date. Your staff finds it useful to also know the day of the week because delivery costs more on Saturday or Sunday. You write a rule that populates the day of the week in the new sales order.

Related Topics

- Set Up Transformation
- Create Transformation Rules
- Use Groups to Manage and Control Sales Orders
- Manage Pretransformation Rules
- How Order Management Transforms Source Orders Into Sales Orders

Procedures

Set Up Transformation

Set up transformation so Oracle Order Management can correctly transform each source order.

1. Use the Product Information Management work area.
    - Set up the product that your transformation rule will reference.
    - If your transformation rule will reference.
        - The product structure, then set up the product structure.
        - Transactional item attributes, then set up transactional item attributes.
    - Create item substitution rules when you set up your item. Order Management doesn't validate item substitutions.

      For example, assume a fulfillment line includes a nonconfigured item, the Order Entry Specialist examines availability for the item, then Order Management displays more than one option. If you don't set up each item substitution rule correctly, then the Order Entry Specialist might select an option that uses an incorrect substitution, such as a kit.
2. Create your transformation rules.

Related Topics

- Overview of Transformation Rules
- Create Transformation Rules
- How Order Management Transforms Source Orders Into Sales Orders

Create Transformation Rules

Create a complex transformation rule that uses a bucket set in a decision table.

Learn how to use a simplified rule builder. For details, see *Use Visual Information Builder*.

Assume you use priority shipping for each sales order that includes a Green Server. You need a rule.

- If the item is a Green Server, then use priority shipping.

You will create a bucket set that contains the values you will select when you create your transformation rule, then create the rule.

We strongly recommend that you get details about decision tables, bucket sets, and how to use them. For details, see *Overview of Using Business Rules With Order Management*.

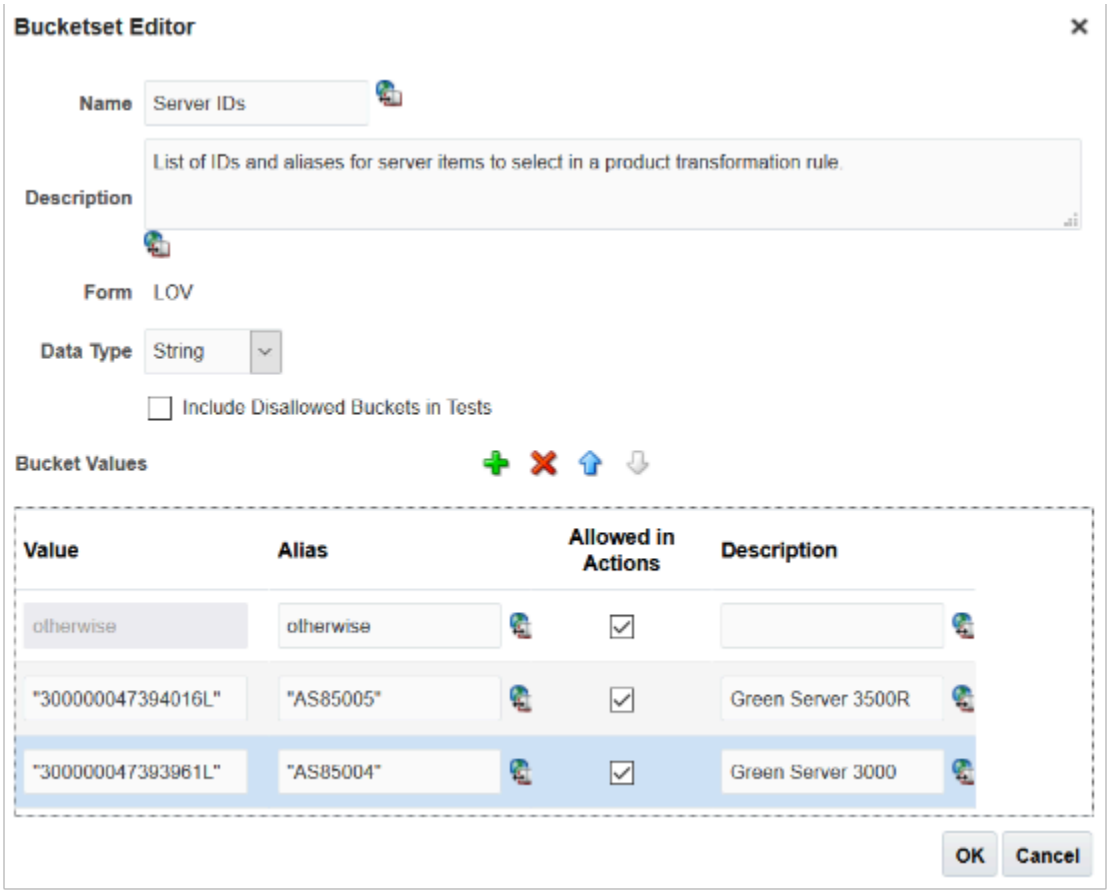
Summary of the Set Up

1. Create the bucket set list.
2. Create the decision table and add a condition.
3. Add a rule to the decision table.
4. Add the action to take if the condition is true.

This topic uses example values. You might need different values, depending on your business requirements.

Create the Bucket Set List

You will create this bucket set list.



Do it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Product Transformation Rules
2. On the Manage Product Transformation Rules page, click **Bucket sets > Add Bucket set > List of Values**.
3. In the Bucket Sets list, click **Bucket Set 1 > Edit Bucket set**.
4. In the Bucket Set Editor dialog, set the values.

Attribute	Value
Name	Server IDs
Description	List of IDs and aliases for server items to select in a product transformation rule.
Data Type	String

5. Click **Add Bucket**, then, in the Bucket Values list, in the Bucket 1 row, set the values.

Attribute	Value
Value	300000047394016L This number identifies the item ID that the database contains. The letter L indicates a long value.
Alias	"AS85005" You must include the double quotation marks.
Description	Green Server 3500R

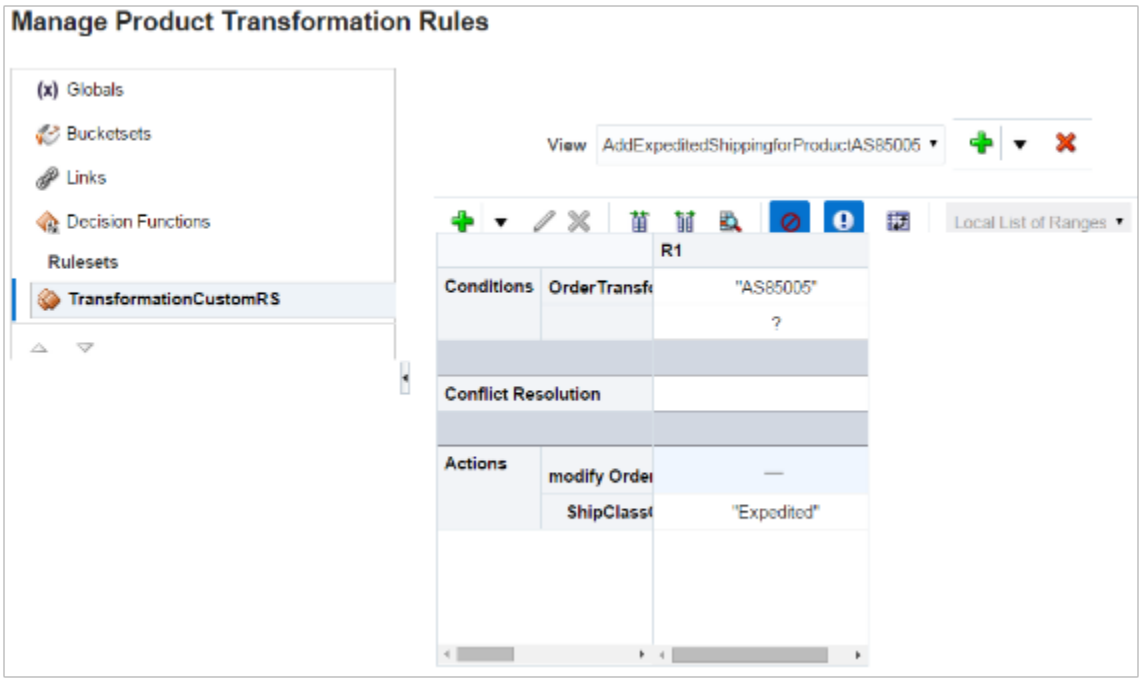
6. Click **Add Bucket**, then, in the Bucket Values list, in the Bucket 2 row, set the values.

Attribute	Value
Value	300000047393961L
Alias	"AS85004"
Description	Green Server 3000

7. Click **OK > Save**.

Create the Decision Table and Add a Condition

You will create the decision table, condition, and action.



Do it.

1. On the Manage Product Transformation Rules page, click **TransformationCustomRS**.
2. In the View list, click **IF/THEN Rules > Add > Add Decision Table**.
3. Replace Decision Table 1 with `AddExpeditedShippingforProductAS85005`, then click **Save**.
4. Immediately above the decision table, click **Add > Add Condition**.
5. In the Condition Browser dialog, expand **OrderTransformationRules > FulfillLineVO > InventoryItemId**, click **toString**, then click **OK**.

   The Condition Browser dialog displays objects from the OrderTransformationRules dictionary. You can use the FulfillLineVO object in this dictionary to specify transformation according to the value of a fulfillment line attribute, such as InventoryItemId.

   In this example, you use toString to get the value of InventoryItemId as a string so you can compare it in the rule.

Add a Rule to the Decision Table

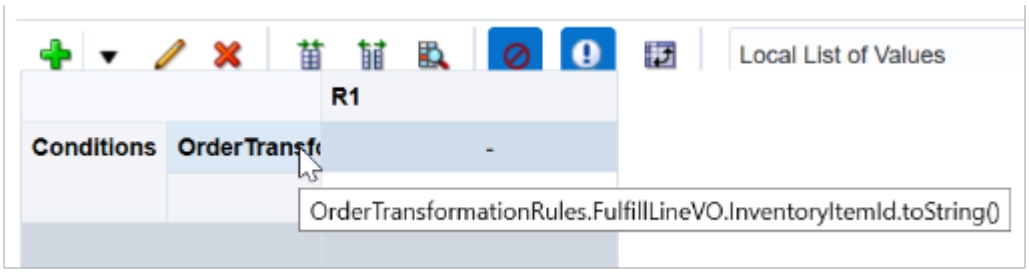
You will add this rule.

- If InventoryItemId is AS85005

Do it.

1. In the decision table, click the **cell** that displays condition OrderTransformationRules.FulfillLineVO.InventoryItemId.toString().

   For example:



2. Click **Local List of Values > Server IDs**, which is the bucket set you created earlier.
3. In row OrderTransformationRules.FulfillLineVO.InventoryItemId.toString(), double-click the **cell** in column R1, single-click the **cell** in column R1, then add a check mark to **AS85005**.

Add the Action to Take If the Condition Is True

You will add this action.

- Set the ShipClassOfService attribute to "Expedited"

At run time, order fulfillment will recognize that ShipClassOfService is Expedited, then expedite the shipment.

Do it.

1. Immediately above the decision table, click **Add > Add Action > Modify**.
2. In the Action Editor dialog, in the Target list, click **OrderTransformationRules.FulfillLineVO**.
3. In the Arguments list, locate the ShipClassOfService argument, set the value, then click **OK**. You might need to scroll or page down through the Arguments list.

Property	Value
Parameterized	Contains a check mark. The Parameterized property makes the object available to the business rule as a parameter.

4. In the decision table, in the ShipClassOfService row, double-click the cell in column R1, then enter `"Expedited"`. You must include the double quotation marks.
5. Click **Save**.

Related Topics

- Overview of Transformation Rules
- Create Advanced Transformation Rules
- Use Visual Information Builder
- Manage Pretransformation Rules

Create Advanced Transformation Rules

Create an advanced transformation rule that compares two or more lines in a source order.

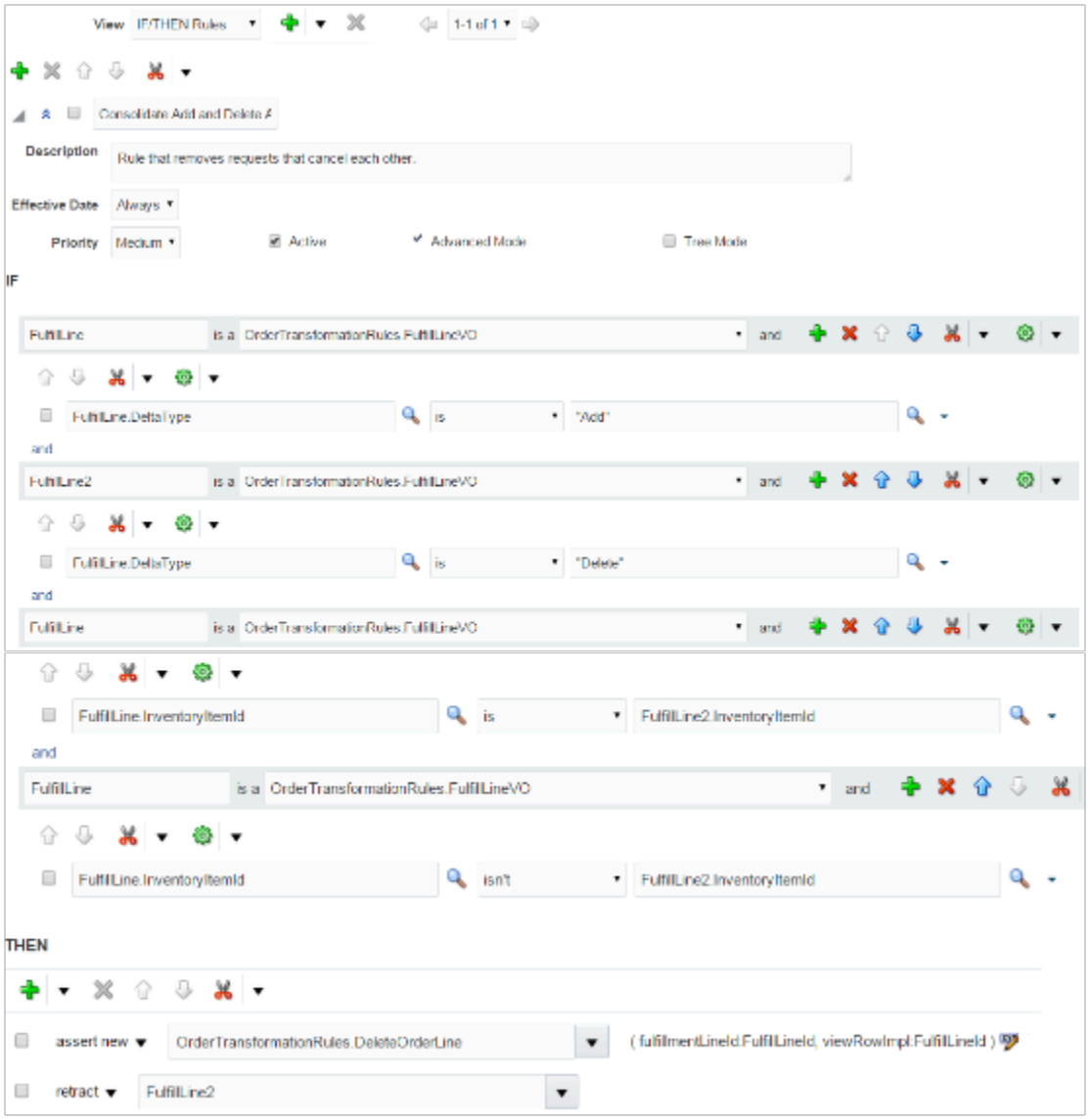
Assume a fulfillment line adds an inventory item, but then another fulfillment line deletes the same inventory item. You can create a transformation rule that avoids processing these lines.

- If fulfillment line *a* requests to add inventory item *x*, and if fulfillment line *b* requests to delete inventory item *x*, then delete fulfillment lines *a* and *b*

You will implement this logic.

Statement	Description
First IF statement	If the change on fulfillment line 1 is Add.
Second IF statement	If the change on fulfillment line 2 is Delete.
Third IF statement	If the inventory item on fulfillment line 1 is the same as the inventory item on fulfillment line 2.
Fourth IF statement	If the fulfillment line ID on fulfillment line 1 is different from the fulfillment ID on fulfillment line 2.
THEN statement	Delete fulfillment line 1 and fulfillment line 2.

You will create this transformation rule.



This topic uses Advanced Mode. We strongly recommend that you familiarize yourself with this mode and with creating business rules. For details, see *Overview of Using Business Rules With Order Management*.

Summary of the Set Up

1. Create the rule.
2. Create the first IF statement.
3. Create the second IF statement.
4. Create the third IF statement.
5. Create the fourth IF statement.
6. Create the THEN statement.

This topic uses example values. You might need different values, depending on your business requirements.

Create the Rule

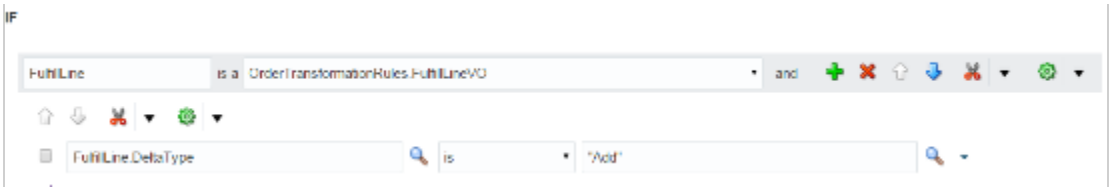
1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Product Transformation Rules
2. On the Manage Product Transformation Rules page, click **Advanced**.
3. Click **Properties**, then set the values.

Attribute	Value
Rule 1	Consolidate Add and Delete Actions
Description	Rule that removes requests that cancel each other.
Advanced Mode	Contains a check mark.

4. Click **Save > Save**.
5. Click **Add > Add Rule**.

Create the First IF Statement

You will create this IF statement. It determines whether the change in fulfillment line 1 is Add.

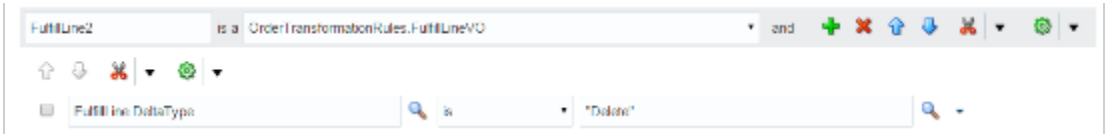


Try it.

1. In the field to the left of `Is A`, enter `FulfillLine`.
2. In the field to the right of `Is A`, click the **down arrow**, then click **OrderTransformationRules.FulfillLineVO**.
3. Click **Add Test > Simple Test**.
4. Click **Left Value**.
5. In the Condition Browser, expand **FulfillLine**, then click **DeltaType > OK**.
6. In the Right Value attribute, enter `"Add"`. You must include the double quotation marks.
7. Click **Save**.

Create the Second IF Statement

You will create this IF statement. It determines whether the change in fulfillment line 2 is Delete.



Try it.

1. Click **Add Pattern**.
2. In the window below `And`, enter `FulfillLine2`.
3. In the field to the right of `Is A`, click the **down arrow**, then click **OrderTransformationRules.FulfillLineVO**.
4. Click **Add Test > Simple Test**, then click **Left Value**.
5. In the Condition Browser, expand `FulfillLine2`, then click **DeltaType > OK**.
6. In the attribute Right Value, enter `"Delete"`. You must include the double quotation marks.

   Click **Save**.

Create the Third IF Statement

You will create this IF statement. It determines whether the inventory item in fulfillment line 1 is the same as the inventory item in fulfillment line 2

Try it.

1. Click **Add Pattern**.
2. In the window below `And`, enter `FulfillLine`.
3. In the field to the right of `Is A`, click the **down arrow**, then click **OrderTransformationRules.FulfillLineVO**.
4. Click **Add Test > Simple Test**, then click **Left Value**.
5. In the Condition Browser, expand **FulfillLine**, click **InventoryItemId**, then click **OK**.
6. Click **Right Value**.
7. In the Condition Browser, expand `FulfillLine2`, then click **InventoryItemId > OK**.

   Click **Save**.

Create the Fourth IF Statement

You will create this IF statement. It determines whether the fulfillment line ID of fulfillment line 1 is different from the fulfillment ID of fulfillment line 2.

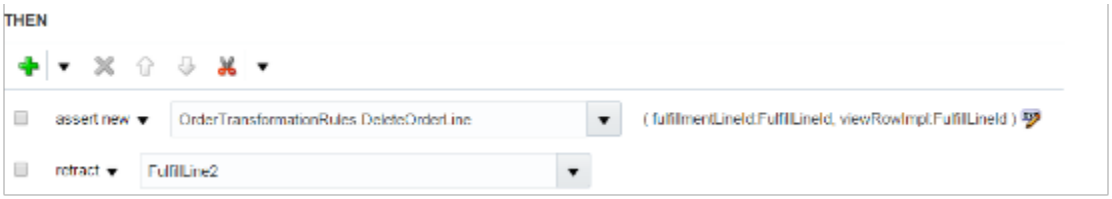
Try it.

1. Click **Add Pattern**.
2. In the window below `And`, enter `FulfillLine`.
3. In the field to the right of `Is A`, click the **down arrow**, then click **OrderTransformationRules.FulfillLineVO**.
4. Click **Add Test > Simple Test**, then click **Left Value**.
5. In the Condition Browser, expand **FulfillLine**, then click **InventoryItemId > OK**.
6. Click **Is > Isn't**.
7. Click **Right Value**.
8. In the Condition Browser, expand `FulfillLine2`, click **InventoryItemId > OK**.

Click **Save**.

Create the Then Statement

You will create this action. It deletes fulfillment line 1 and fulfillment line 2.



Try it.

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target**, then click **OrderTransformationRules.DeleteOrderLine**.
3. Click **Edit Properties**.
4. In the Properties dialog, enter values, then click **OK**

Name	Value
fulfillmentLineId	fulfillmentLineId
viewRowImpl	fulfillmentLineId

5. Click **Add Action > Retract**.
6. Click **Select a Target**, then click `FulfillLine2`.

Click **Save**.

Related Topics

- Overview of Transformation Rules
- Roadmap for Setting Up Order-to-Cash
- Create Transformation Rules

Add Promotional Items That Reward Your Customers

Add an item to a sales order to reward customers who purchase larger quantities.

For example, if your customer orders a quantity of 5 for the AS54888 desktop computer, then add a free AS9000 router on another order line on the same order.

Summary of the Setup

You can use an order management extension or a product transformation rule to add a promotional item. We recommend that you don't use an order management extension and a product transformation rule at the same time to do this.

1. Create an extension or create product transformation rule.
2. Test your setup.

Create an Extension

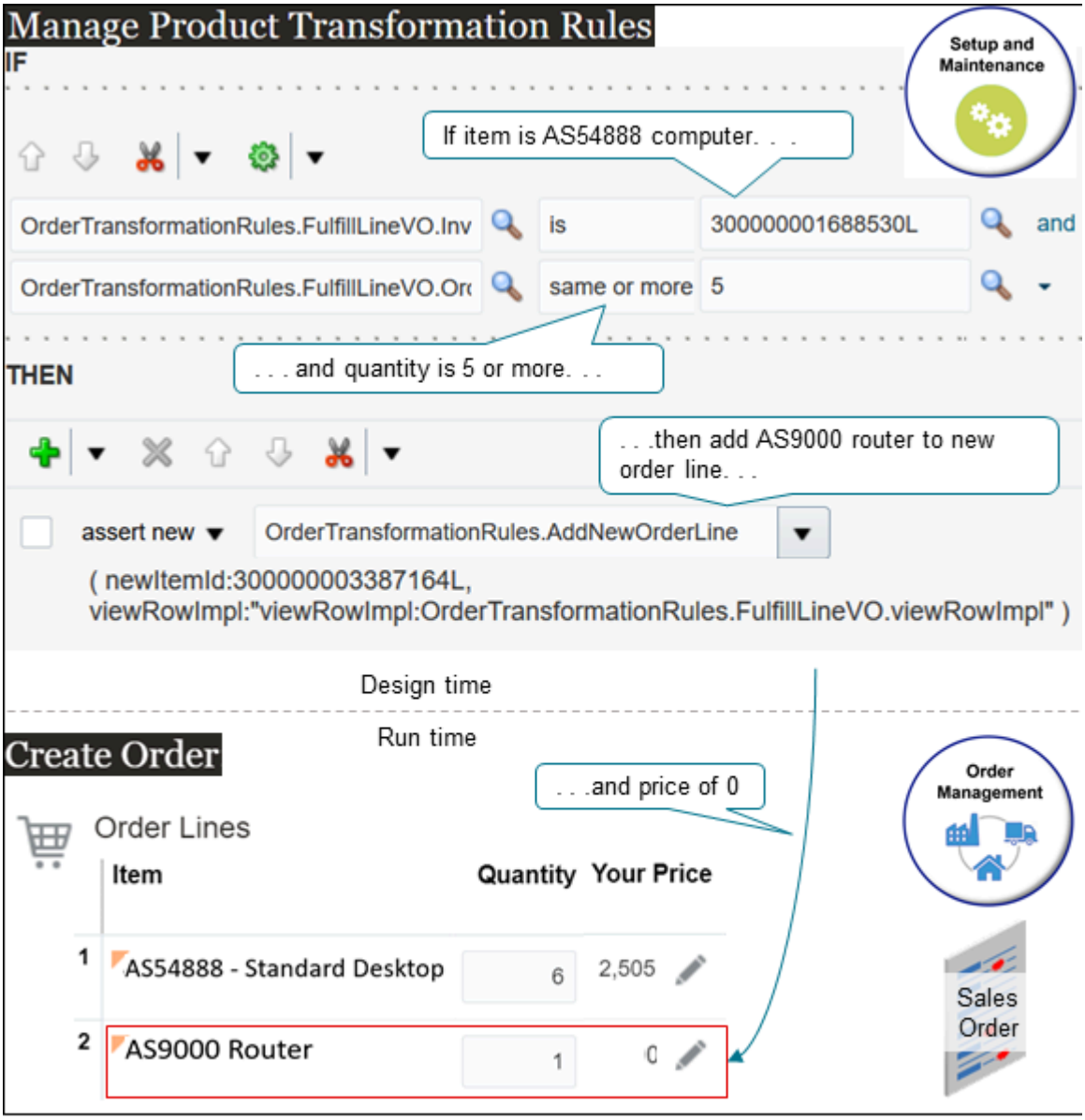
- Use the setProductNumber parameter of the createLineParams method to specify the item that you want to add.
- Use the setOrderedQuantity parameter of createLineParams to specify the quantity for the item that you want to add.
- Use an IF statement to specify the quantity that you want to use as the threshold that determines whether to add a new line.
- You can use createLineParams to set the ProductNumber, OrderedUOM, and OrderedQuantity attributes on new lines that you add. You can't use it to set any other attribute. You can use a posttransformation rule to set the value for other attributes.

For example:

```
import oracle.apps.scm.doo.common.extensions.CreateLineParams;
if (!"SUBMIT_AFTR_VALID".equals(header.getAttribute("CustomerPONumber"))) return; /* This line is only for
 testing purposes. Remove it after you successfully test this extension.*/
def createLineParams = new CreateLineParams();
createLineParams.setProductNumber("AS9000");
/* Specify the item.
that you want to add on the new line.*/
createLineParams.setOrderedUOM("Each")
createLineParams.setOrderedQuantity(1); /* Specify the quantity for the item that you're adding on the new
 line. */
def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def isClosed = line.isClosed()
 def isCanceled = line.isCanceled()
 def isTransformed = line.isTransformed()
 def transformLines = line.getTransformedLines();
 if (isClosed || isCanceled || isTransformed || transformLines.size() != 0) {
 continue;
 }
 if (line.getAttribute("ProductNumber") == AS54888 && line.getAttribute("OrderedQuantity") >= 5)
 /* Specify the quantity that you want to use as the threshold that determines when to add a new line. */
 {
 line.createNewLine(createLineParams);
 }
}
```

For details, see *Overview of Creating Order Management Extensions*.

Create a Product Transformation Rule



Try it.

1. Get the IDs for the inventory items.
    - Do an SQL.

      ```
      SELECT DISTINCT item_number,
       inventory_item_id
      FROM fusion.egp_system_items_b
      WHERE upper(item_number) LIKE '&ITEM_NUMBER%')
      ```

      For details, see *Use SQL to Query Order Management Data*.

- In the query result, locate the row that contains AS54888 in the ITEM_NUMBER column. Assume the query returns these values.

ITEM_NUMBER	INVENTORY_ITEM_ID
AS54888	300000001688530L
AS9000	300000003387164L

2. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Product Transformation Rules
3. On the Manage Product Transformation Rules page, create a new rule.
   Don't use Advanced Mode.
4. In the If area, set the conditions.

Code	Description
`OrderTransformationRules.Fulfi is 300000001688530L`	FulfillLineVO is a virtual object that contains runtime values of the attributes on the fulfillment line. Order Management populates FulfillLineVO when you create the sales order. This statement says to get the value of the InventoryItemId attribute. This attribute is in the FulfillLineVO virtual object (VO), and the OrderTransformationRules dictionary contains FulfillLineVO. If the InventoryItemId attribute contains 300000001688530L, which is the AS54888 desktop computer, then proceed to the next condition in this rule. If it doesn't, then exit the rule.
`OrderTransformationRules.Fulfi is same or more than 5`	Get the value of the OrderedQty attribute. If the OrderedQty attribute contains a value of 5 or greater, then proceed to the Then statement in this rule. If it doesn't, then exit the rule.

5. In the Then area, add a new action.

Code	Description
`Assert New OrderTransformationRules.AddNe`	Add a new order line to the sales order.
`newItemId: 300000003387164L`	Add the 300000003387164L item, which is the Vision Router, to the order line.

Code	Description
	You use Assert New to add a new object, such as a new order line.
`viewRowImpl:OrderTransformatio`	Use viewRowImpl:FLine.ViewRowImpl to specify the row that contains the runtime value.

6. Click **Save > Release**.

   For details, see *Overview of Using Business Rules With Order Management*.

Test Your Setup

1. Go back to your other browser and create a sales order.
2. Add an order line.

Attribute	Value
Item	AS54888
Quantity	5 or more

3. Click **Submit**.
4. Verify that Order Management added a new order line.

Attribute	Value
Item	AS9000 Router
Quantity	1
Your Price	0

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules
- Import Shipping Method
- Use SQL to Query Order Management Data

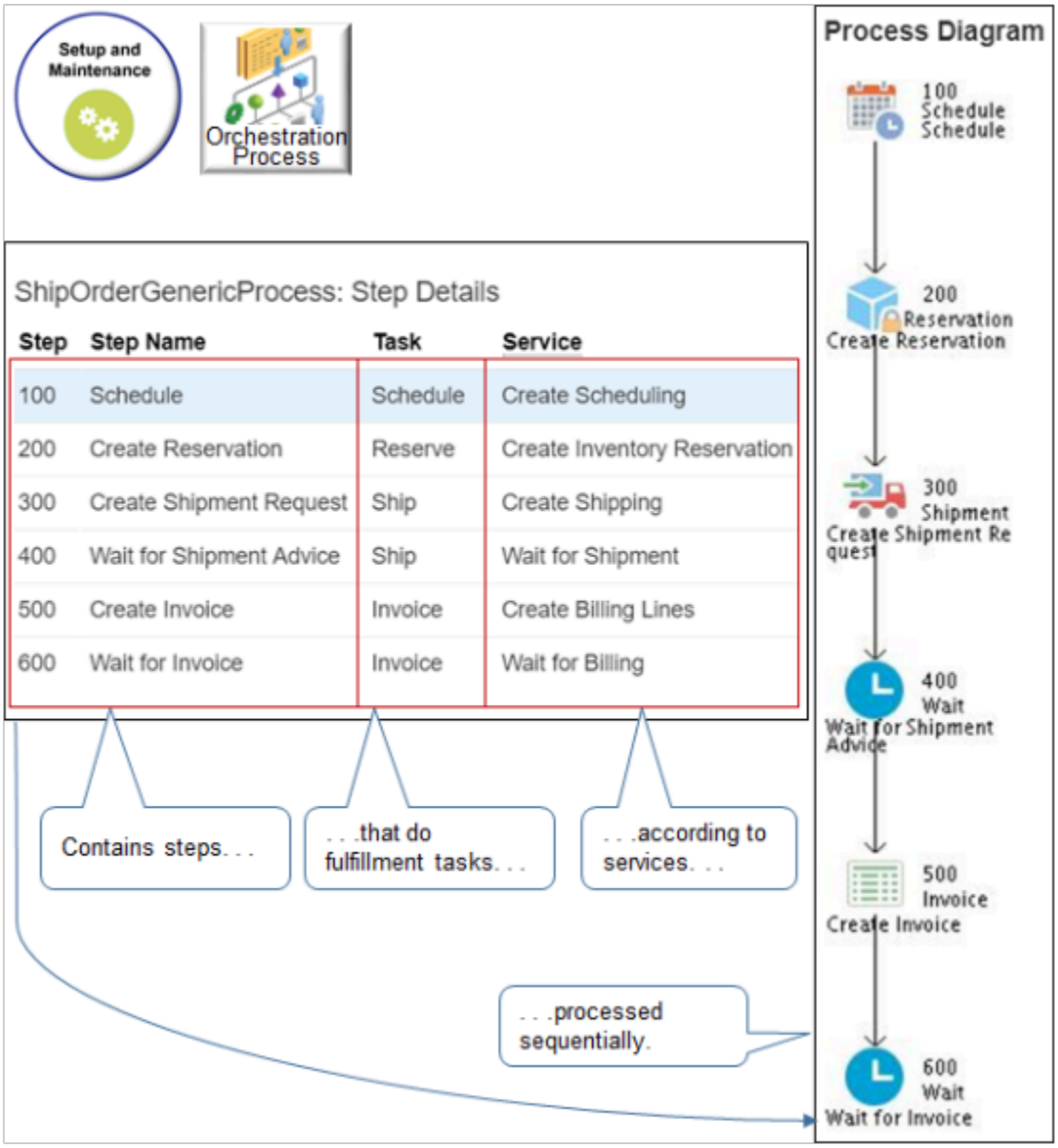
6 Orchestrate

Overview

Overview of Orchestration Processes

Set up an orchestration process so it meets the fulfillment requirements for your sales order.

An orchestration process is a sequence of steps that automate fulfilling your sales order's fulfillment lines across fulfillment systems. For example, here's the predefined ShipOrderGenericProcess orchestration process.



Note

- Each process contains steps.
- Each step calls a service that does a fulfillment task. The service communicates with the fulfillment system to do the task. For example:

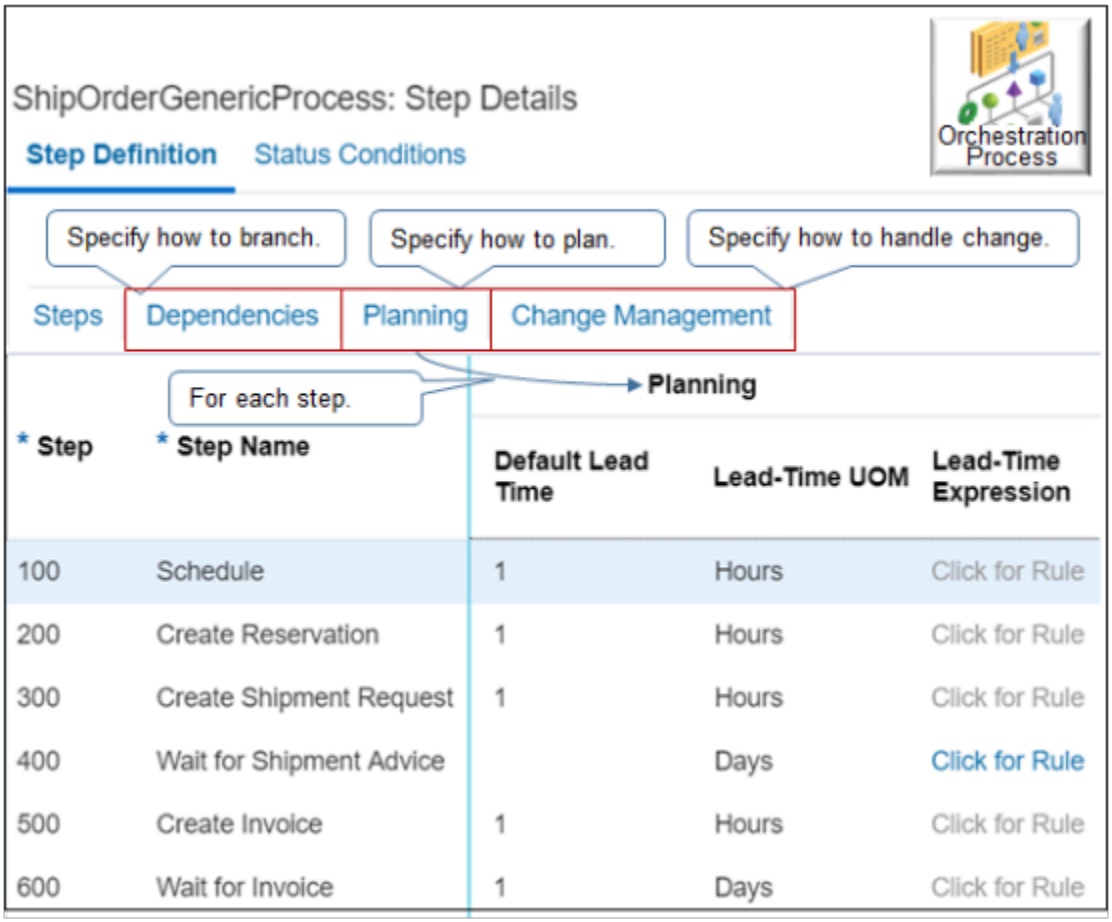
Step	Service	Description
Step 100, Schedule	Create Scheduling	Create a schedule that the process can use to fulfill the fulfillment line. It makes sure fulfillment meets the delivery dates that the sales order specifies.

Step	Service	Description
Step 200, Create Reservation	Create Inventory Reservation	Reserve inventory for the item on the fulfillment line so no other order can use the inventory.
Step 300, Create Shipment Request	Create Shipping	Create a shipping plan that makes sure the carrier delivers the item on time.
Step 400, Wait for Shipment Advice	Wait for Shipment	Wait to receive confirmation that the carrier delivered the item.
Step 500, Create Invoice	Create Billing Lines	Create billing lines that we can send to accounts receivable.
Step 600, Wait for Invoice	Wait for Billing	Wait to receive confirmation from accounts receivable that billing is done.

- The orchestration process does each step sequentially, starting with step 100 and ending with step 600.
- You use the Setup and Maintenance work area to create and manage your orchestration process.
- See how an orchestration process works in context, and get an overview of how task services interact with the fulfillment system. For details, see *How Order-to-Cash Works in Order Management*.

Set up Steps

You do almost all your set up on the steps.



Note

- **Dependencies**. If your process includes a branch, then set dependencies that affect branching.
- **Planning**. Specify how to plan fulfillment for the fulfillment line.
    - Set the default lead-time that the process needs to do the step.
    - Create a business rule that uses conditions to set lead-time.
    - Specify the status you expect from the fulfillment system during planning.
    - Specify the step that determines whether fulfillment is done.
- **Change Management**. Specify how to manage change that occurs in the fulfillment system.
    - Specify the task service to use for update messages or cancel messages that the fulfillment system sends to you.
    - Create a business rule that uses conditions to specify what to do when a change occurs.
- Specify dependency, planning, and change management for each step. For example, click **Planning** to scroll to the part of the row that contains planning attributes.

Set Up Statuses

Note

- Different fulfillment systems might use different status values.
- Set up statuses for your orchestration process so they match the values that your fulfillment system uses. This way, the fulfillment system understands values that the orchestration process sends, and the process knows how to handle values it receives.

  For example, fulfillment system x might use SCHEDULED, and system y might use FULFILLMENT_SCHEDULED. Add values to handle both systems.
- You can also set statuses for fulfillment lines.

Related Topics

- Examples of Setting Up Orchestration Processes
- Guidelines for Setting Up Orchestration Processes
- Set Up Orchestration Processes
- Fix Errors in All Sales Orders
- How Order-to-Cash Works in Order Management

Examples of Setting Up Orchestration Processes

Learn about some of the ways you can use an orchestration process.

Call Task Services

Assume you require an orchestration process that incorporates a company policy.

- If an invoice exceeds $100,000, then a representative must phone the customer.

Here's what you need to do.

1. Identify the steps your business process needs to fulfill the sales order.
2. Set up your orchestration process so it includes these steps.

You identify the sequence of calls your orchestration process must make to task services. Here's the pseudocode.

1. Plant Acknowledgment.
2. Assemble.
3. Wait for status to equal COMPLETE.
4. Ship.
5. Wait for status to equal SHIPPED.
6. Call Customer.
7. Wait for Call Customer to equal COMPLETE.
8. Test for these conditions.

   If the invoice exceeds $100,000, then Send High Value Invoice.

   - Wait for the status for High Value Invoice to equal BILLED.

   If the invoice doesn't exceed $100,000, then.

   - Invoice.
   - Wait for the status to equal BILLED.
9. End the condition.

Use Planning Details

Assume you establish lead-times that allow your sales representative to monitor order fulfillment.

Step	Lead Time
Schedule	Two days
Reserve	One day
Ship	Six days
Invoice	Two days

You add the default lead time to each orchestration process step. If a step gets delayed during fulfillment, then a process runs in the background that replans the orchestration process and resets the expected completion dates.

Set a Status

Assume you have an important customer who requires that you notify their receiving clerk one day before the shipping system ships the item. You set up an orchestration process class for the orchestration process. You determine the class must include statuses.

- SHIPPED
- RESERVED
- READY TO SHIP
- SHIPPED
- INVOICED

You use the Orchestration Process Status tab to define a status condition.

- If the status of the Create Shipment step is PRESHIP READY, then use the READY TO SHIP status to indicate the orchestration process status.

Your users can use the Order Management work area to determine whether the orchestration process status is READY TO SHIP.

Related Topics

- Set Up Orchestration Processes
- Add Lead-Times for Orchestration Process Steps
- Order Management Statuses

Guidelines for Setting Up Orchestration Processes

Create a copy of a predefined orchestration process instead of creating a new one. Using a copy of a predefined orchestration process and copying other predefined objects will help reduce your development and maintenance effort.

Predefined Orchestration Processes

Predefined Orchestration Process	Includes These Tasks
ShipOrderGenericProcess	• Schedule • Reservation • Shipment • Invoice
ReturnOrderGenericProcess	• Return Receipt • Invoice
OrderFulfillmentGenericProcess	• Schedule Conditional. Starts the branch. • Request Supply. Starts the back-to-back branch. • Pause. • Create Back to Back Shipment Request. • Wait for Back to Back Shipment Request. • Create Purchase Request. Starts the drop ship branch. • Wait for Procurement. • Create Reservation. Starts the warehouse shipment branch. • Create Shipment Request. • Wait for Shipment Advice. • Merge. Ends the branch. • Invoice. • Wait for Invoice. Use it for various fulfillment requirements, such as back-to-back shipments, drop ship, and so on.

Note

- Use a predefined orchestration process as your first course of action.

  Create a copy of a predefined orchestration process only if the predefined orchestration process doesn't meet your needs.

  Don't modify a predefined orchestration process. Instead, create a copy of it, then modify the copy.
- Create a new orchestration process only if modifying the copy of a predefined one doesn't meet your needs.

Do the Required Set Ups

Do these set ups in the same order that the table lists them

Description	Get Details
Set up the task types that arrange fulfillment tasks in groups. Each task type	*Set Up Task Types for Holds*

Description	Get Details
references services that communicate with a type of fulfillment system. For example, a billing system.	
Set up the orchestration process.	*Set Up Orchestration Processes*
Set up business rules that determine how the orchestration process handles changes to sales orders.	*How Order Management Processes Change Orders*
Set up the schedule that uses process planning to display the completion date of each task and the orchestration process.	*Guidelines for Setting Up Orchestration Process Steps*
Set up jeopardy threshold and jeopardy priority to assess the level of risk that's associated with the delay of an orchestration process task as low, medium, or high.	*Jeopardy Threshold*
Set up the status and status conditions for the sales order, task, orchestration process, fulfillment line, or order line.	*Orchestration Process Status*
Deploy your orchestration process.	*Deploy Orchestration Processes*

Plan Your Orchestration Process

Orchestration planning is the process of orchestrating and planning fulfillment for your sales order. For example, planning dates, planning shipments, and so on.

Specify how Order Management plans each orchestration process.

- Set and help meet the completion date for each orchestration process step and task in an orchestration process.
- Specify how to use the transformation rules that an orchestration process references to transform each source order, including planning for each step after it receives a source order from an order capture system.
- Use the Enable Orchestration Process Planning and Calculate Jeopardy parameter. For details, see *Manage Order Management Parameters*.

Replan Your Orchestration Process

Replanning is the process of updating the orchestration plan for the sales order to accommodate a change that happens to the sales order or in the fulfillment environment. For example, when your customer requests to change the quantity of a sales order you already submitted to fulfillment.

Specify how Order Management replans each orchestration process.

- Replan completion dates when a change happens to the sales order at any point in the orchestration process.

- Replan immediately after each orchestration process step finishes.
- Replan according to an event, such as every time the orchestration process receives an update from your order capture system. To control replanning, use the Plan Orchestration Processes scheduled process to schedule an update at regular intervals according to the frequency that your deployment requires.

  For example, if your orchestration process requires planning data that's current, then set up the scheduled process to run the orchestration process and update the planning data one time each day. See an example that uses a scheduled process. For details, see *Fix Errors in More Than One Sales Order*.

Enable the Replan Instantly option when you set up your orchestration process. It replans the orchestration process immediately after the process finishes the orchestration process step, then displays the revised order data according to the results of the replanning.

- Use Replan Instantly only for high priority sales orders, or with sales orders that include a jeopardy threshold of less than one day.
- For performance reasons, don't use Replan Instantly with an orchestration process step that's long or complex.
- If you don't enable Replan Instantly, then Order Management updates planning data only during the scheduled replanning.

What if I Need to Apply New Processing to Orders I Already Imported?

You must cancel the orders you already imported, modify the orchestration process definition or create a new one, then reimport your sales orders.

Migrate

If you use your implementation project to migrate an orchestration process instance from a development environment to a production environment, then don't modify the process name in either environment. Modifying the name might prevent Order Management from updating references to other data in the orchestration process. For details about using an implementation project, see *Guidelines for Setting Up Order-to-Cash*.

Copy and Duplicate

If you need to copy or duplicate an orchestration process, then make sure you validate it and remove all validation errors before you create the copy or duplicate. If you don't remove validation errors on the original process, then you might encounter a problem when you attempt to release the copy or duplicate. The Orchestration Process Definition page might not display any errors for the copy or duplicate but the status remains as New and doesn't update to Released when you attempt to release it.

Use the DOO_ScheduleShipInvoice Orchestration Process

Use the predefined DOO_ScheduleShipInvoice orchestration process to process a large volume of sales orders.

DOO_ScheduleShipInvoice is similar to DOO_OrderFulfillmentGenericProcess, except DOO_ScheduleShipInvoice only has the Scheduling, Shipping, and Invoice tasks.

You can modify DOO_ScheduleShipInvoice to integrate with your own fulfillment system. For example, if you need to add your own pause task or a custom task to integrate with your own fulfillment system, then you must include the statuses that your fulfillment system needs for the pause task and for the custom task.

One way to do this is to examine the orchestration process classes and see what statuses they use.

1. Go to the Setup and Maintenance work area, then search for and open the Manage Status Values task.

2. Click Orchestration Process Classes, then find a class that works for you. Make a note of the statuses that the class uses.
3. Add the statuses that you noted in the class to your task.

For details, see:

- *Use FBDI and REST API to Import Large Volumes of Sales Orders*
- *Use REST API to Import Large Volumes of Sales Orders*
- *Use Order Management Extensions to Import Large Volumes of Sales Orders*

Related Topics

- Examples of Setting Up Orchestration Processes
- Set Up Orchestration Processes
- Guidelines for Setting Up Orchestration Process Steps

Guidelines for Setting Up Orchestration Process Steps

Specify details about the step, such as branching, planning, and managing change.

Each step in your orchestration process specifies the task service that the step calls to fulfill the fulfillment line. Each step specifies how to run the step, such as the task type, task, service, dependencies, planning, change management, and so on. A step might also specify branching.

- To set up an orchestration process step, you open the Edit Orchestration Process Definition page, then use the Step Definition list in the Process Details area.
- You can't update an order line after Order Management interfaces the line to billing, so don't add a scheduling, reservation, or shipping step after a billing step.

Set Up the Orchestration Process Step

Attribute	Description
Step Type	Set the behavior for the orchestration process step. • **Conditional.** Runs a path in an orchestration process according to the results of a condition. You must specify a branching condition on a step immediately after the conditional step. • **Merge.** Identifies the point where two or more orchestration paths reunite. • **Service.** Use this step to call a service.
Task Type	Each task type includes services you can use to communicate with a fulfillment system, such as a billing system. Here are the predefined task types you can use. • Schedule • Reservation • Shipment • Invoice

Attribute	Description
	• Return
Task	Specify the task to run. A task can include more than one step. For example, the Ship task calls the Create Shipment service. It also calls the Wait for Shipment service to wait for different status values to occur in the fulfillment system updates. CAUTION: If you use your implementation project to migrate an orchestration process instance from a development environment to a production environment, then don't modify the task name in either environment. Modifying the name might prevent Order Management from updating references to other data in the orchestration process. For details, see *Guidelines for Setting Up Order-to-Cash*.
Service	Specify the task service that this step calls.
Manual	Specify whether to wait for user input. If you specify a manual task, then the orchestration process waits until the user manually finishes the task in the Order Management work area. For example, set the Schedule task as a manual task so your users can manually schedule all fulfillment lines at the end of the day. The orchestration process waits the first time the user submits the sales order. It doesn't wait if the user revises the order line. Assume you enable the Manual attribute on the Reservation step and the next step after reservation is a shipping task. At runtime, the process stops at the Reservation step and sets the status on the order line to Manual Reservation Required. The user finishes the manual task, orchestration moves to the shipping task, sets the status to Awaiting Shipping, then proceeds to fulfillment. If the user revises the order line, then orchestration will proceed to fulfillment for the revision, such as schedule, reserve, and ship, but it won't stop on the Reservation step.
Exit Criteria	Specify the task status that determines when to exit a wait step. For example, if the status of a shipment task changes to Shipped, then exit the wait step. If your task includes more than one wait step, then make sure these steps don't use the same exit criteria. Use the Manage Task Status Conditions page to make sure only the last step or wait step uses the exit criterion. If you set up more than one wait step for a task, then make sure you set the exit criteria for each wait step that occurs before the final wait step to Mark as Complete. For example: • Assume you set up wait step x, wait step y, and wait step z on the same task, and that wait step z occurs last. • Make sure you set the exit criteria for step x and step y to Mark as Complete. • If the fulfillment system doesn't reply to step x or step y, but instead replies only to step z, then the flow can continue without waiting for step x and step y to finish. • If you don't set Mark as Complete for x and y, then the orchestration process might remain at x or y and never proceed to the next task. • Make sure you specify the value Canceled as an exit criteria status to exit the wait task in your orchestration process.
Line Selection Criteria	Add a business rule that selects fulfillment lines. The orchestration process step will then process only these fulfillment lines. • The rule populates the result with the fulfillment line Ids that identify the fulfillment lines to select. • The rule runs for each fulfillment line that the orchestration process is processing.

Attribute	Description
	• If the rule doesn't select any fulfillment line, then the orchestration process skips the step. An orchestration process might not require all of the order lines or fulfillment lines when it calls the fulfillment task service. For example, assume the item on an order line is a warranty. You typically don't ship a warranty, so you can create a business rule that specifies not to ship items that don't ship. For details, see *Select Fulfillment Lines for Orchestration Process Steps*.
Pause Rule	Specify when to pause processing before calling the next step, or when to resume processing. For details, see *Pause Orchestration Processes Until Time Elapses*.

Branch Your Process

An orchestration process branch is a path in an orchestration process that the process runs when the flow meets a condition.

You can set up an orchestration process so its linear, where steps occur in a sequence with no branching, or so it contains a branch where flow travels along different paths depending on a condition.

Here's how you set up branching.

- Use a branching condition in a single orchestration process.
- Use an assignment rule that examines a set of orchestration processes, then assigns one for the branch.

The technique you use depends in part on the complexity you need. For example, you can create several simple, linear orchestration processes, then use an assignment rule to choose one. Or, you can combine these orchestration processes into a single orchestration process that uses branching conditions. A more complex set up might require assignment rules and branching conditions.

For details, see *Add Branches to Orchestration Processes*.

Here are the attributes you set for branching.

Attribute	Description
Branching Condition	Specify the criteria that the condition must meet to run the steps in a branch. • Add the condition on the first step of the branch, which is the first step immediately after the conditional step. • If you don't add a check mark to the Otherwise option on the conditional step, then you must include a branching condition.
Evaluation Sequence	Specify the sequence that the orchestration process uses when it evaluates each branch condition. Set the Evaluation Sequence only if your orchestration process uses branching. If it doesn't use branching, then you can't set the sequence.
Otherwise	If you add a check mark to the Otherwise option on the conditional step, and if the branching condition does meet the criteria you specify for the branch, then the orchestration process runs the branch that doesn't meet the branching condition.

Plan Your Process

Attribute	Description
Planning Default Branch	Specify the default path that the orchestration process uses for planning. The process uses this setting only if it includes one or more conditional branches.
Fulfillment Completion Step	Add a check mark to this attribute to indicate that the fulfillment lines are fulfilled when this step finishes. • The orchestration process uses this setting when it does planning to make sure it meets the request date. • The last step that occurs in chronological order in the process isn't necessarily the Fulfillment Completion Step. For example, to indicate the completion date, the orchestration process might use the requested ship date as the last step instead of using the shipped date.
Default Lead Time	Specify the duration that the orchestration process needs to do the step. • The process uses lead-time to plan and to predict the completion date. The lead-time is the amount of time the process needs to finish the step, including wait steps and pause steps. • If you don't specify a lead-time expression for the step, then the process uses the value you set for Default Lead Time. • If actual completion dates are available, then the process uses actual dates instead of estimated dates. • The Gantt chart in the Order Management work area displays the planned orchestration process. Order Management uses the number of days that are past the lead time when it calculates jeopardy.
Lead Time UOM	The unit of measure for the lead-time, such as days, hours, or minutes.
Lead-Time Expression	A lead-time expression is a business rule that determines the amount of time you expect the step needs to finish. • Use it to calculate planning for the orchestration process. For example, an item that uses complex packing might require a longer lead-time for shipping. • The rule populates the result with a numeric value that represents the lead time. • You must use BigDecimal with your lead-time expression. Here's an example expression that determines the difference between the current date and the scheduled ship date. `BigDecimal.valueOf((DooSeededOrchestrationRules.DOOHeader/childFLines.scheduleShipDate.time-CurrentDate.date.timeInMillis)/(1000*60*60*24))` where • BigDecimal is a public Java class that specifies an arbitrary-precision, signed, decimal number. • DooSeededOrchestrationRules is a method that contains a set of predefined business rules for an orchestration process. • DOOHeader/childFLines specifies a relationship between the parent order head and the child fulfillment lines. The forward slash (/) separates the parent from the child.

Attribute	Description
	• scheduleShipDate is a sales order attribute. • CurrentDate is a function that returns the current date. • timeInMillis converts the current date to milliseconds. • 1000 is the number of milliseconds in one second. • 60 is the number of minutes in one hour. • 60 is the number of seconds in one hour. • 24 is the number of hours in one day. For details, see *Add Lead-Times for Orchestration Process Steps*.

Manage Change

Attribute	Description
Hold on wait	Sends a message to the fulfillment system for each active step when the orchestration process receives a change order.
Use Transactional Item Attributes	If you enable transactional item attributes, then the step examines the transactional item attributes to help it determine the differences that exists between the change order and the previous version of the order.
Use Flexfield Attributes	If you enable flexfield attributes, then the step examines them to help it determine the differences that exists between the change order and the previous version of the order.
Compensation Pattern	Specify the set of rules that determine how to handle each step that ran before Order Management received the change order. For example, assume a change order requests a change from carpet to tiles. • The orchestration process must cancel a number of the previous steps that it ran. • If it already scheduled carpet for shipping, then the process must cancel the steps that scheduled the shipping. • Order Management creates a different set of fulfillment lines for tiles when it processes the change order, so it must use a different orchestration process because a tile order requires more time to fulfill and it uses a different contractor. • Order Management must cancel most of the previous steps, but it doesn't cancel the Measure step because the room dimensions are still accurate. If you don't specify a compensation pattern, then Order Management might process a step as an update, depending on the context of the operation. It might rerun some steps. If Order Management can't compensate a step for some reason, then it bypasses the step, then compensates the next step that it encounters.

Plan Your Dates

Order Management sets the required completion date for the last step of the orchestration process to the Required Fulfillment Date, then calculates the planned dates for each step and task that the orchestration process contains so they meet the Required Fulfillment Date.

Here's how Order Management does it.

- Works through the orchestration process from beginning to end, starting with the first step in chronological order, and ending with step Last Fulfillment Completion.
- Doesn't use the last step that occurs in chronological order to identify the completion date.
- Incorporates lead times in the dates that it calculates. You can set up these lead times.
- Displays the replanned schedule after it finishes the calculation.

Here's how Order Management sets the Required Fulfillment Date.

1. Sets the Required Fulfillment Date to the date that your source system provides.
2. If the source system doesn't provide a date, then Order Management sets Required Fulfillment Date to Requested Ship Date.
3. If Requested Ship Date doesn't contain a value, then Order Management sets Required Fulfillment Date to Requested Arrival Date.
4. If Requested Arrival Date doesn't contain a value, then Order Management sets Required Fulfillment Date to Order Date.
5. If Order Date doesn't contain a value, then Order Management sets Required Fulfillment Date to System Date. The operating system of the computer that runs Order Management contains the system date.

Order Management exits this sequence as soon as it successfully sets the date. For example, if your source system provides a date, then it does only step 1, and doesn't do steps 2 through 5.

Keep this sequence in mind when you set up your integration and orchestration process.

Delete a Step

If you delete a step, and if the step references a status value from the orchestration process or the fulfillment line, then you must delete the status value before you delete the step. If you don't, then you can't release the orchestration process.

Assume you copy an orchestration process, it contains Step 3 Create Reservation and Step 4 Create Shipment Request. Step 3 references the Reserved status value from the orchestration process. You know from historical demand data that there's always sufficient inventory and don't want to reserve inventory for the item. So you delete step 3, but then encounter an error when you attempt to release the orchestration process.

```
Task name Reservation does not exist in the Process definition
```

Here's how you can fix that.

1. Go to the Manage Orchestration Process Definitions task, then open your orchestration process for editing.
2. On the Edit Orchestration Process Definition page, in the Process Details area, click **Status Conditions**.
3. Click **Orchestration Process Status Values**.

   If your step referenced a fulfillment line value, then you would click Fulfillment Line Status Values > Edit Status Rule Set, then remove the status value from the rule set.
4. Click the **row** that has Reserved in the Status Value attribute, then click **Actions > Delete**.
5. Click **Step Definition**.
6. Click the **row** that has Reserve in the Step Name attribute, then click **Actions > Delete**.

Duplicate a Predefined Process

If you create a duplicate of a predefined orchestration process and replace the first step of the process with your own custom step, and if you must add a business rule on the first step, then enable the Advanced Mode option and disable

the Tree Mode option when you create your rule. If you can't set these options, then don't add a rule on the first step because your rule will fail validation.

Keep Predefined Rules

Some predefined orchestration processes come with a rule on the first step. For example, the first step of ReturnOrderGenericProcess is named Create Receiving Advice, and it comes with a line-selection criteria named Receiving.

Assume you need to create a custom process that's similar to ReturnOrderGenericProcess, except your first step must send a transportation request. You still need the Receiving line-selection criteria but you need it on the second step, not the first.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, search for ReturnOrderGenericProcess, then click **Actions > Duplicate**.
3. On the Edit Orchestration Process Definition page, set the value.

Attribute	Value
Process Display Name	My Return Order Process

4. In the Process Details area, in the first row, Create Receiving Advice, set the value.

Attribute	Value
Step Name	Temporary You can use any value. You will change it soon.

5. Click the first row, then click **Actions > Add Row**.

   The page adds a new step, 200, right after step 100.
6. Set the values for step 200. You are duplicating step 100, so set these values.

Attribute	Value
Step Name	Create Receiving Advice
Step Type	Service
Task Type	Return

Attribute	Value
Task	Return Receipt
Service	Create Expected Receipt Advice
Update Service	Update Expected Receipt Advice
Cancel Service	Cancel Expected Receipt Advice

7. Click **Save**.
8. Copy the rule in step 100.
    - On step 100, in the Line-Selection Criteria column, click **Click for Rule**.
    - In the Line-Selection Criteria dialog, notice the rule named Receiving, click **Properties**, then set the values.

Attribute	Value
Advanced Mode	Contains a check mark
Tree Mode	Doesn't contain a check mark

      You must set these values. If you don't, your copy will fail validation.
    - Add a check mark in the **option** that's right next to Receiving, click the **down arrow** next to the scissors, click **Copy**, then click **Cancel**.
9. Paste the rule into step 200.
    - On step 200, in the Line-Selection Criteria column, click **Click for Rule**.
    - In the Line-Selection Criteria dialog, click **Paste**.
    - Click Validate, then verify that the Message contains `No validation errors/warnings found!.`
    - Click **Save**.
10. On step 100, set the value.

Attribute	Value
Step Name	Transportation Request

Set other attributes to meet your requirements.

Improve Performance

The way you set up your orchestration processes and orchestration process steps can directly affect performance in your environment. Here are some tips.

- Remove each step that you don't need. Each step eats up resources and increases processing time. For example, if you use the predefined ShipOrderGenericProcess and don't need to do invoicing, then delete the Create Invoice and Wait for Invoice steps.

  Don't use line-selection criteria to skip steps you don't need because that approach increases processing time and might lead to a step that remains in a Not Started status for a long time, or a process that seems to take forever to finish.
- Use a separate orchestration process to accomplish each goal. Don't use one large, monolithic process that has a lot of branches and skipped steps to do all your processing. Instead, use several processes with fewer branches.

Related Topics

- Set Up Orchestration Processes
- Roadmap for Setting Up Order-to-Cash

Create

Set Up Orchestration Processes

Set up an orchestration process so it meets your business requirements.

In this example, you create an orchestration process that calls a scheduling service, ships, then bills the sales order.

Summary of the Set Up

1. Prepare to set up the orchestration process.
2. Create the orchestration process.
3. Add the orchestration process steps.
4. Deploy the orchestration process. For details, see *Deploy Orchestration Processes*.

This topic uses example values. You might need different values, depending on your business requirements.

Prepare to Set Up the Orchestration Process

Depending on the complexity of your orchestration process, you might need to do one or more of these steps.

1. Do tasks in the Setup and Maintenance work area that allow you to set up an orchestration process.
2. Create the task types that the orchestration process requires. Also define the tasks and services that these task types reference.

3. Create the status codes that the orchestration process requires and specify how the task types, fulfillment lines, and your orchestration process will use them.
4. Create the status catalogs that the orchestration process will use for status conditions. Create catalogs for one of:
    - Product Information Management
    - Oracle Product and Catalog Management

Create the Orchestration Process

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, click **Actions > Create**.
3. On the Create Orchestration Process Definition page, set the values, then click **Save**.

Attribute	Value
Process Name	CustomDOO_ScheduleAndShip The set up page automatically adds the CustomDOO_ prefix. You must use this prefix for any orchestration process you create. The name must not exceed 80 characters, including the prefix.
Process Display Name	CustomDOO_ScheduleAndShip Enter any value. Don't include spaces. The value you that enter will display throughout the Setup and Maintenance work area and the Order Management work area.
Process Class	Ship Order Class An orchestration process class contains a set of statuses that you can assign to the orchestration process. Select a class that includes the statuses you need for the orchestration process.
Set	Common Set Note - A set is a collection of business units. Use it to organize business units and control the business units that can access an orchestration process. - Select the set that allows the business unit to access the orchestration process. - Use the predefined Common Set to provide access to many business units.

Attribute	Value
	○ You must add your business unit to the set you specify.

Add the Orchestration Process Steps

1. On the Step Definition tab, click **Actions > Add Row**.
2. Set the values, then click **Save**.

Attribute	Description
Step Name	Enter text that describes the purpose of the step. For this example, enter `Schedule Product`.
Step Type	Select a value that indicates the type of processing that this step does. For example, if this step must do conditional branching, then select Conditional. This example calls a service, so select Service.
Task Type	Select the group of services that Order Management uses to do a fulfillment task. For this example, select Schedule.
Task	Select the task that this step must perform. For this example, select Schedule.
Service	Identify the service that this step must call. For this example, select Create Scheduling.

3. Repeat steps 1 and 2. Use these values.

Attribute	Value
Step Name	Reserve Product
Step Type	Service
Task Type	Reservation
Task	Reserve
Service	Create Inventory Reservation

Attribute	Value

Tip: To maintain the correct sequence when you add each step, click the **step** that you most recently added, then click **Add Row**. To make sure you don't lose any work, click **Save** after you add each step.

4. Repeat steps 1 and 2. Use these values.

Attribute	Value
Step Name	Ship Product
Step Type	Service
Task Type	Shipment
Task	Ship
Service	Create Shipping

5. Repeat steps 1 and 2. Use these values.

Attribute	Value
Step Name	Wait for Product Shipment
Step Type	Service
Task Type	Shipment
Task	Ship
Service	Wait for Shipment
Exit Criteria	Shipped

Note that you also specify the exit criteria in this step.

6. Repeat steps 1 and 2. Use these values.

Attribute	Value
Step Name	Create Invoice
Step Type	Service
Task Type	Invoice
Task	Invoice
Service	Create Billing Lines

7. Repeat steps 1 and 2. Use these values.

Attribute	Value
Step Name	Wait for Invoice
Step Type	Service
Task Type	Invoice
Task	Invoice
Service	Wait for Billing

Related Topics

- Guidelines for Setting Up Orchestration Process Steps
- Overview of Orchestration Processes
- Deploy Orchestration Processes

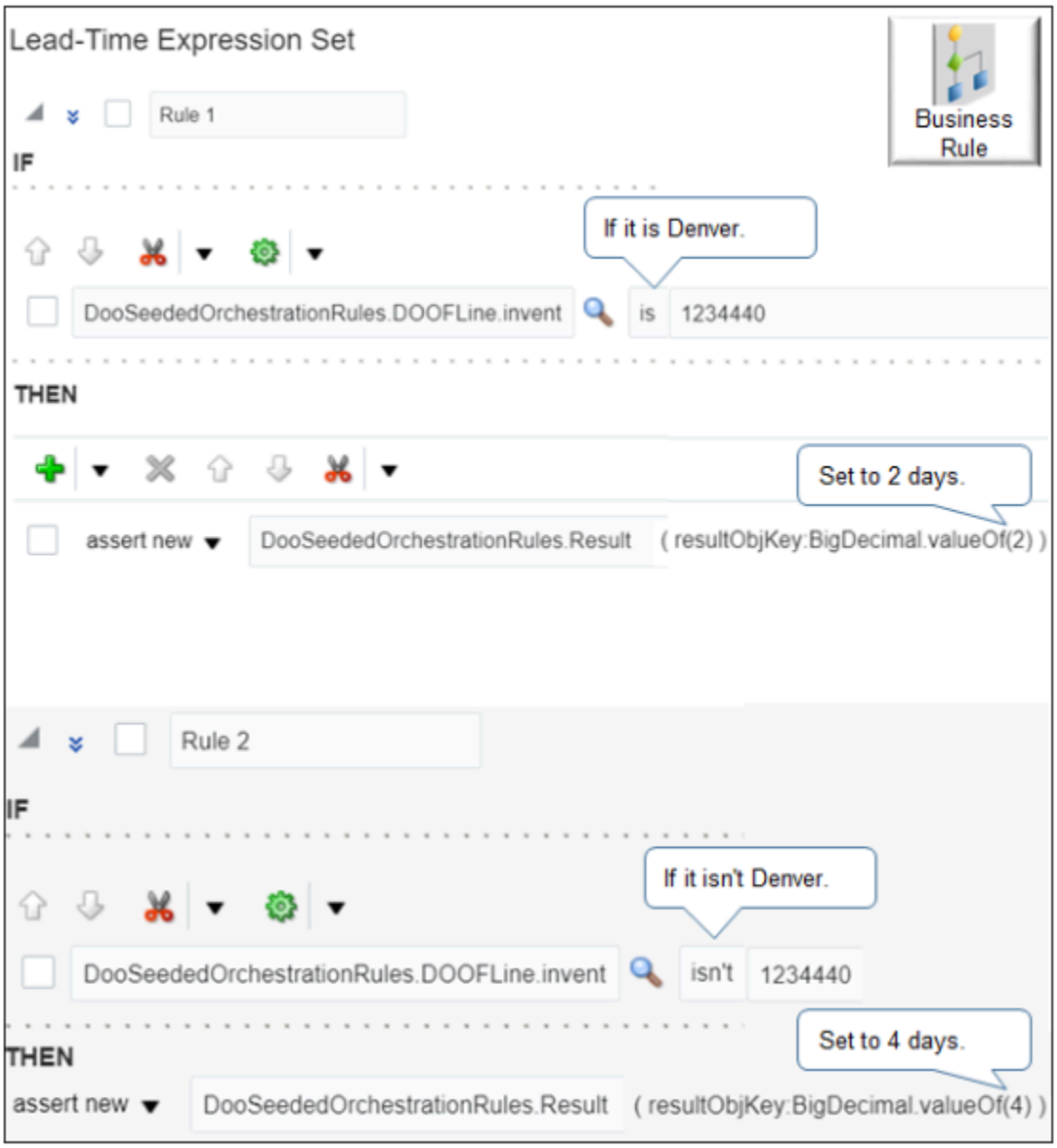
Add Lead-Times for Orchestration Process Steps

Add a rule that sets the lead-time for an orchestration process step according to a set of conditions.

Assume your lead-time to ship the item varies depending on where the inventory organization is located, so you implement these business rules:

- If the inventory organization is in Denver, then use a two day lead-time
- If the inventory organization isn't in Denver, then use a four day lead-time

Here are the rules you will create.



Summary of the Steps

1. Create the If statement for the first rule.
2. Create the Then statement for the first rule.
3. Create the If statement for the second rule.
4. Create the Then statement for the second rule.

This topic uses example values. You might need different values, depending on your business requirements.

Create the If Statement for the First Rule

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.

    Learn how to create CallCustomerWhenLargeInvoice. For details, see *Add Branches to Orchestration Processes*.
3. On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, click **Planning**.
4. In the Ship Product row, set the value.

Attribute	Value
Lead-Time UOM	Days

5. In the Lead-Time Expression column, click **Click for Rule**.
6. In the Lead-Time Expression dialog, click **Add Rule > Expand**.

    If you write a rule for an orchestration process that orchestrates more than one fulfillment line, then use Advanced Mode. However, this example treats all fulfillment lines in the same way, so Advanced Mode isn't required. To use Advanced Mode, you would click **Properties**, add a check mark to **Advanced Mode**, then edit the rule.
7. In the If area, click **Left Value**.
8. In the Condition Browser, expand **DOOSeededOrchestrationRules > DOOFline**, click **InventoryOrganizationId**, then click **OK**.

    DOOFline is an abbreviation for Distributed Order Orchestration Fulfillment line. Oracle replaced the name Distributed Order Orchestration with the name Order Management in an earlier update. Some rules still use the old name.
9. In the Right Value attribute, enter `1234440`.

    Assume 1234440 is the inventory organization ID for Denver.

Create the Then Statement for the First Rule

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Edit Properties dialog, in the ResultObjKey row, enter the value, then click **OK**.

Attribute	Value
Value	`BigDecimal.valueOf(2)`

Create the If Statement for the Second Rule

1. In the Lead-Time Expression dialog, in the Lead-Time Expression Set area, click **Add Rule**.
2. In the lower part of the dialog, in the Rule 2 area, click **Expand**.
3. In the If area of Rule 2, click **Left Value**, expand **DOOSeededOrchestrationRules > DOOFline**, select **InventoryOrganizationId**, then click **OK**.

   > **Tip:** In some attributes, you can copy the value from one attribute to another instead of using the drop-down list. For example, you can copy DooSeededOrchestrationRules.DOOFLine.inventoryOrganizationId from Left Value of the first rule, then paste it into Left Value of the second rule.

4. Click the **down arrow** for the operator, then click **isn't**.
5. In the Right Value attribute, enter `1234440`.

Create the Then Statement for the Second Rule

1. In the Then area for Rule 2, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Edit Properties dialog, in the ResultObjKey row, enter the value.

Attribute	Value
Value	`BigDecimal.valueOf(4)`

5. Click **OK > Save**.
6. On the Edit Orchestration Process Definition page, click **Save**.

Related Topics

- Add Branches to Orchestration Processes
- Guidelines for Setting Up Orchestration Process Steps
- Overview of Using Business Rules With Order Management

Select Fulfillment Lines for Orchestration Process Steps

Create a line selection rule that selects fulfillment lines, then specify whether the orchestration process step will process them.

A line selection rule is type of rule that determines whether an orchestration process step processes a fulfillment line. There are some cases where you don't want to process a line.

- The line has an item isn't shippable, such as a subscription or warranty.
- You set the Purchasable attribute to Yes for the item in the Product Information Management work area.

Use a line selection rule to prevent the orchestration process step from processing these lines.

Example

In this example, you create a line selection rule that makes sure Order Management doesn't attempt to ship a nonshippable item.

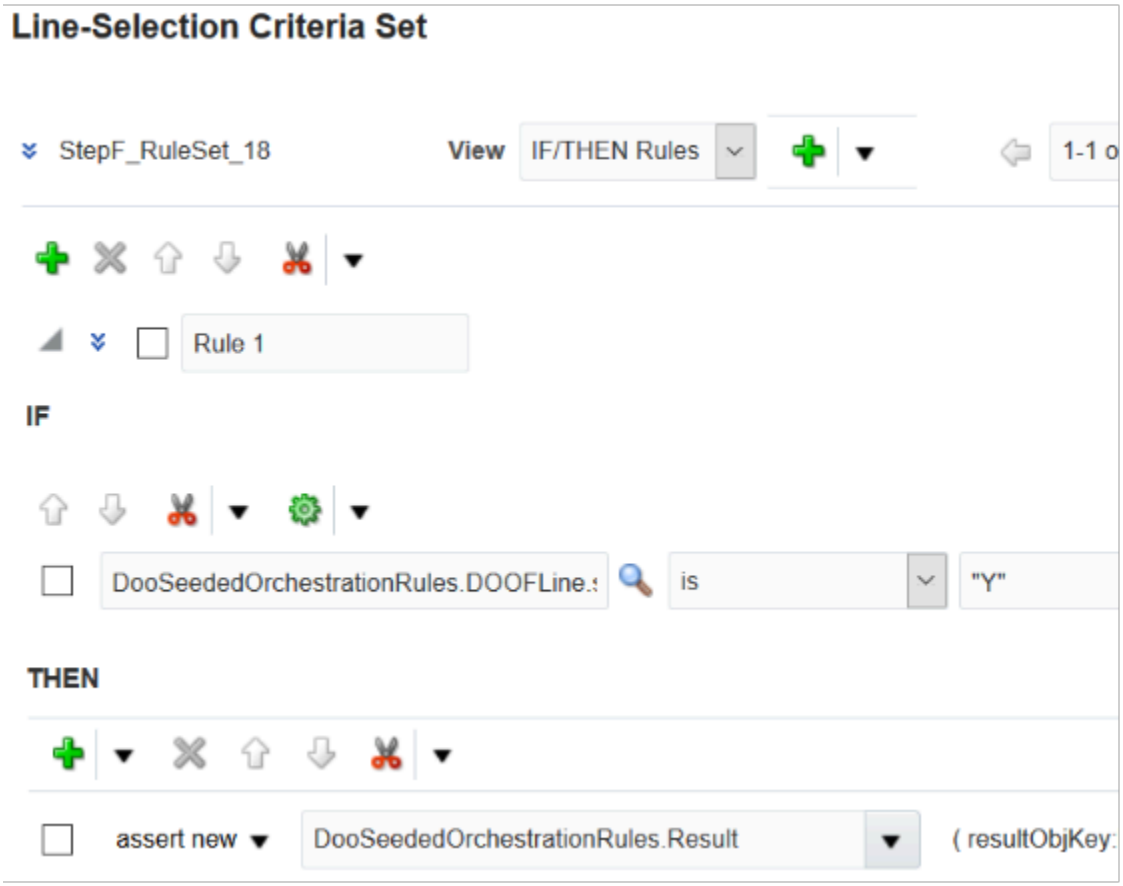
Assume you sell digital video recorders. The sales order includes more than one fulfillment line for each items.

- Digital video recorder
- Remote control
- Instruction manual
- Extended warranty

Your customers can purchase the extended warranty as a contract online, but its not a shippable item, so Order Management must not attempt to send it to the fulfillment system during the Shipment task. So, you create a rule.

- If Order Management can't ship the item, then don't attempt to ship it.

Here's the rule you will create.



where

- DooSeededOrchestrationRules is a dictionary that contains a set of predefined functions, variables, objects, and other data you can use to define behavior for an orchestration process.

- DOOFLine contains fulfillment line attributes, such as orderedQty, customerPONumber, creationDate, and so on. You can reference these attributes and use their values in your rule.
- shippableFlag is a fulfillment line attribute.
- Y is one possible value of shippableFlag.

This example includes a business rule that requires you to use a dictionary, fact, and other objects. We strongly recommend that you familiarize yourself with these objects before you proceed. For details, see the Business Rules chapter in the Implementing Order Management book.

Summary of the Steps

1. Create the If statement.
2. Create the Then statement.

This topic uses example values. You might need different values, depending on your business requirements.

Create the If Statement

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.

    Learn how to create CallCustomerWhenLargeInvoice. For details, see *Add Branches to Orchestration Processes*.
3. On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, locate the Ship Item row.
4. In the Ship Item row, in the Line Selection Criteria column, click **Click for Rule**.
5. In the Line Selection Criteria dialog, click **Add Rule > Expand**.
6. In the If area, click **Left Value**, expand **DOOSeededOrchestrationRules > DOOFline**, click **ShippableFlag**, then click **OK**.
7. In the Right Value attribute, enter `"Y"`. You must include the double quotation marks.

Create the Then Statement

You will create this Then statement.

- `assert new resultObjKey:DooSeededOrchestrationRules.DOOFLine.fulfillLineId`

where

- Assert New adds the result into the Result fact of the DooSeededOrchestrationRules dictionary.
- resultObjKey is a property of Result. It specifies the fulfillment line to examine.
- `DooSeededOrchestrationRules.DOOFLine.fulfillLineId` specifies to use the value of the `fulfillLineId` attribute in the DOOFLine fact of DooSeededOrchestrationRules.

Create the Then statement.

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.

3. Click **Edit Properties**.
4. In the Properties dialog, in the ResultObjKey row, click **Value**.
5. In the Condition Browser, enter `DOOFLine`, wait a moment, expand **DooSeededOrchestrationRules.DOOFLine**, click `FulfillLineId`, then click **OK > OK**.
6. In the Line Selection Criteria dialog, click **Validate**, make sure the Validation Log that displays doesn't contain errors, then click **Save**.
7. On the Edit Orchestration Process Definition page, click **Save**.

Related Topics

- Add Branches to Orchestration Processes
- Roadmap for Setting Up Order-to-Cash

Prevent Orchestration Process from Shipping Return Lines

Add a filter to a line selection rule that makes sure the shipping task doesn't attempt to ship a return line.

The Category Code attribute contains ORDER for an order line that's going to fulfillment, and RETURN for a return line. You can use CategoryCode to filter out return lines.

Here's the filter you add to each step.

Try it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions

2. On the Manage Orchestration Process Definitions page, search for ShipOrderGenericProcess, then click **Actions > Duplicate**.

   You can't modify the predefined process, so you create a duplicate and modify it instead. Notice the process name.

   ```
   (CopyOf)CustomDOO_ShipOrderGenericProcess
   ```

   Use this process in your deployment.
3. On the Edit Orchestration Process Definition page, locate the schedule row.

Step	Step Name
100	Schedule

4. On the step 100 row, in the Line Selection Criteria column, click **Click for Rule**.
5. In the Line Selection Criteria dialog, click **Expand**, then notice the predefined rule. Locate the last condition, immediately above the THEN area.
6. On the row you just located, click the **down arrow** at the far right of the row, then click **Simple Test**.
7. In the new row, enter `"ORDER"`, change `is` to `equals ignore case`, then enter `DooSeededOrchestrationRules.DOOHeader/childFLines.categoryCode`.

   Verify the entire row looks like.

   ```
   "ORDER" equals ignore case DooSeededOrchestrationRules.DOOHeader/childFLines.categoryCode
   ```
8. Click **Validate**, then click **Save**.
9. Repeat these steps for the Create Reservation step.
10. Repeat these steps for the Create Shipment Request step.

Add Branches to Orchestration Processes

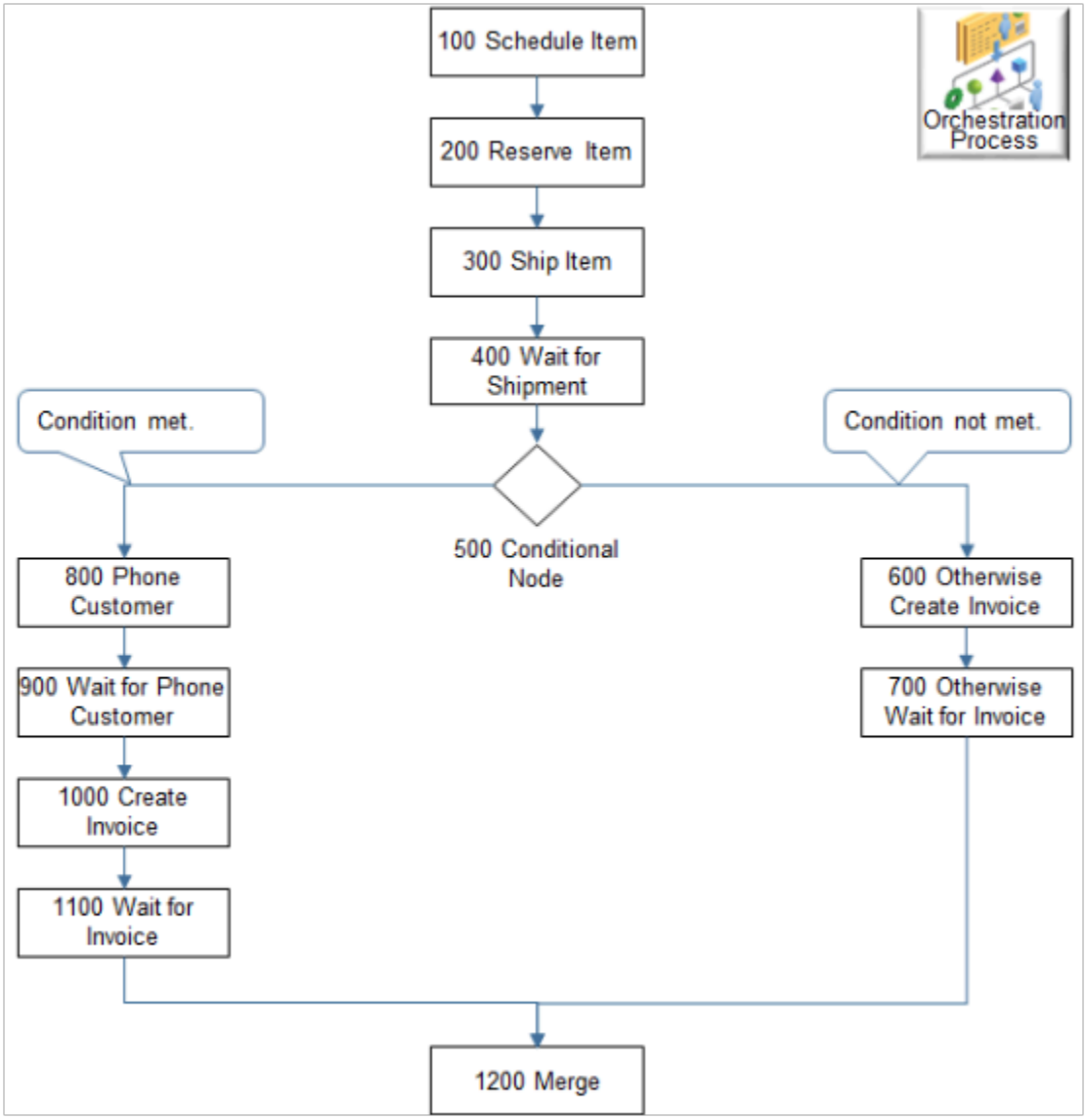
Create a branching condition that determines whether to run a branch on an orchestration process.

An orchestration process branch is a path in an orchestration process that the process runs when the flow meets a condition.

Assume you sell commercial computer systems, and you need a business rule.

- If the sales order is valued at $50,000.00 or more, then make sure a representative calls the customer before sending the invoice for the order.

Here's the flow that you will create for this example.



Note

- Each step in the flow includes the step number, task name, task type, and service name.
- The Conditional Node indicates that an orchestration process is about to branch. The first step of the branch contains the condition.
- If the flow.
    - **Meets the condition.** The orchestration process runs the steps on the branch that notifies the representative.
    - **Doesn't meet the condition.** The orchestration process runs the steps on the branch that don't notify the representative.
- Order Management adds an empty default branch when it runs the orchestration process. If the orchestration process includes only one branch, then it isn't necessary to set an Otherwise condition.

This example uses a rule that processes only one fulfillment line. You use an advanced rule to write a rule for an orchestration process that processes more than one fulfillment line. For details, see *Overview of Using Business Rules With Order Management*.

Summary of the Set Up

1. Route the notification to the representative.
2. Create the orchestration process.
3. Add the orchestration process steps.
4. Create the If statement.
5. Create the Then statement.
6. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Route the Notification to the Representative

This example sends a notification to a representative to call the customer if the invoice is valued at $50,000.00 or more. You create the routing that enables the notification. This setup is specific to this example. Other branching usages might not require this setup, or they might require some other setup.

1. Create the routing rules that enable the send. For details, see *Manage Routing Rules*.
2. Create the connector that you referenced in the routing rules in step 1.
3. Use the Manage Web Service Details page to create the connector.

Create the Orchestration Process

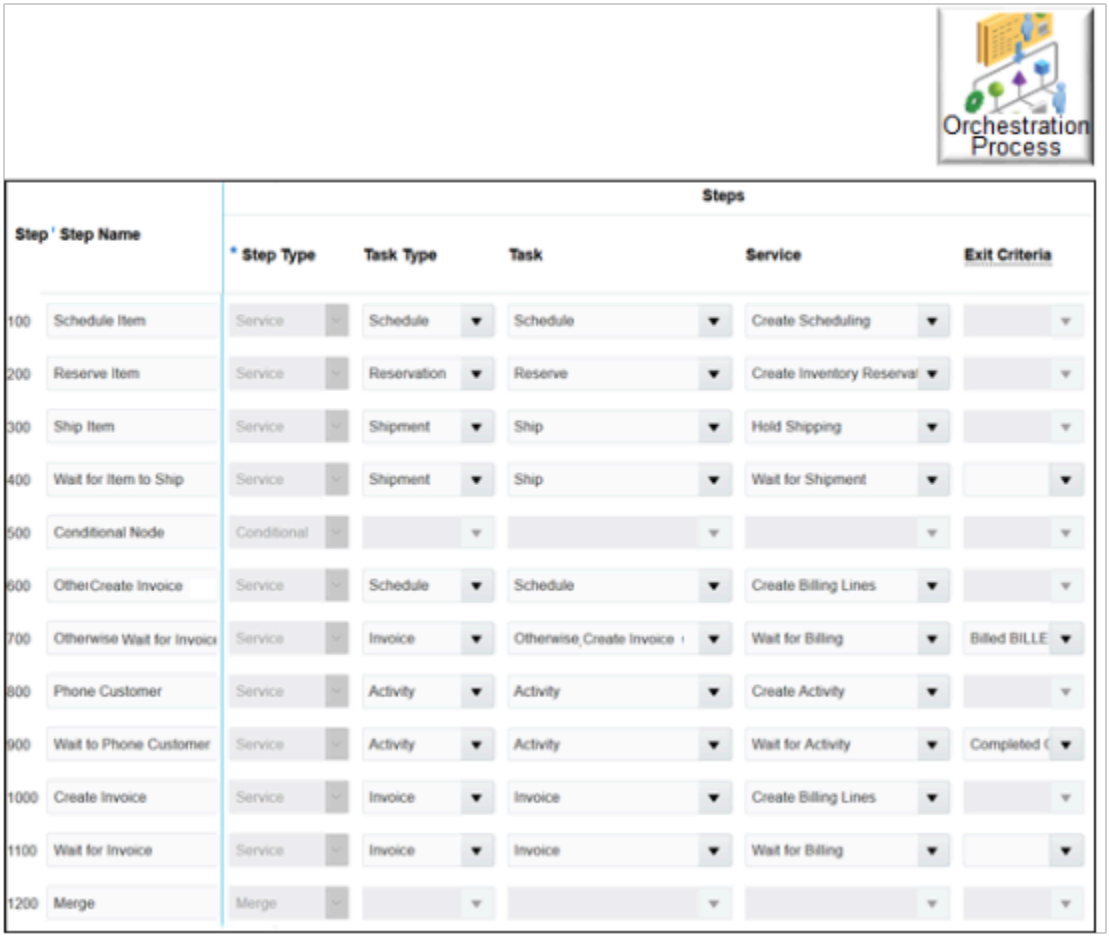
Use these values in the header of the orchestration process when you create the process.

Attribute	Value
Process Name	CallCustomerWhenLargeInvoice
Process Display Name	Call the Customer
Process Class	Ship Order Class
Set	Common Set

For details, see *Guidelines for Setting Up Orchestration Processes*.

Add the Orchestration Process Steps

You will add these steps.



1. Add a step.

Attribute	Value
Step Name	Schedule Item
Step Type	Service
Task	Schedule
Service	Create Scheduling

2. Add another step.

> **Tip:** To maintain the correct sequence when you add each step, click the step you most recently added, then click **Add Row**.

Attribute	Value
Step Name	Reserve Item
Step Type	Service
Task	Reserve
Service	Create Inventory Reservation

3. Add another step.

Attribute	Value
Step Name	Ship Item
Step Type	Service
Task	Ship
Service	Create Shipping

4. Add another step.

Attribute	Value
Step Name	Wait for Item to Ship
Step Type	Service
Task	Ship
Service	Wait for Shipment

Attribute	Value
Exit Criteria	Shipped
Fulfillment Completion Step	Contains a check mark.

5. Add another step.

Attribute	Value
Step Name	Conditional Node
Step Type	Conditional

6. Add another step.

Attribute	Value
Step Name	Phone Customer
Step Type	Service
Task	Activity
Service	Create Activity
Evaluation Sequence	1

7. Add another step.

Attribute	Value
Step Name	Wait for Phone Customer
Step Type	Service
Task	Activity
Service	Wait for Activity

Attribute	Value
Exit Criteria	Completed

8. Add another step.

Attribute	Value
Step Name	Create Invoice
Step Type	Service
Task	Invoice
Service	Create Billing Lines

9. Add another step.

Attribute	Value
Step Name	Wait for Invoice
Step Type	Service
Task	Invoice
Service	Wait for Billing
Exit Criteria	Billed

10. In the Step Definition list, click the Conditional Node step, then add another step. This step creates the Otherwise branch.

Attribute	Value
Step Name	Otherwise Create Invoice
Task Type	Invoice

Attribute	Value
Task	Otherwise Create Invoice Each task name that you use with a task type must be unique. You used the Invoice task with the Invoice task type earlier in this procedure, so you must create a new task name. To do this, In the Task attribute, click the Search **down arrow**, then click **Create**. In the Create Task Name dialog, enter values. o Code: 1 o Name: Otherwise_Create_Invoice o Display Name: Otherwise Create Invoice o Task Type: Invoice Click **Save and Close**.
Service	Create Billing Lines
Evaluation Sequence	2
Otherwise	Contains a check mark.
Planning Default Branch	Contains a check mark.

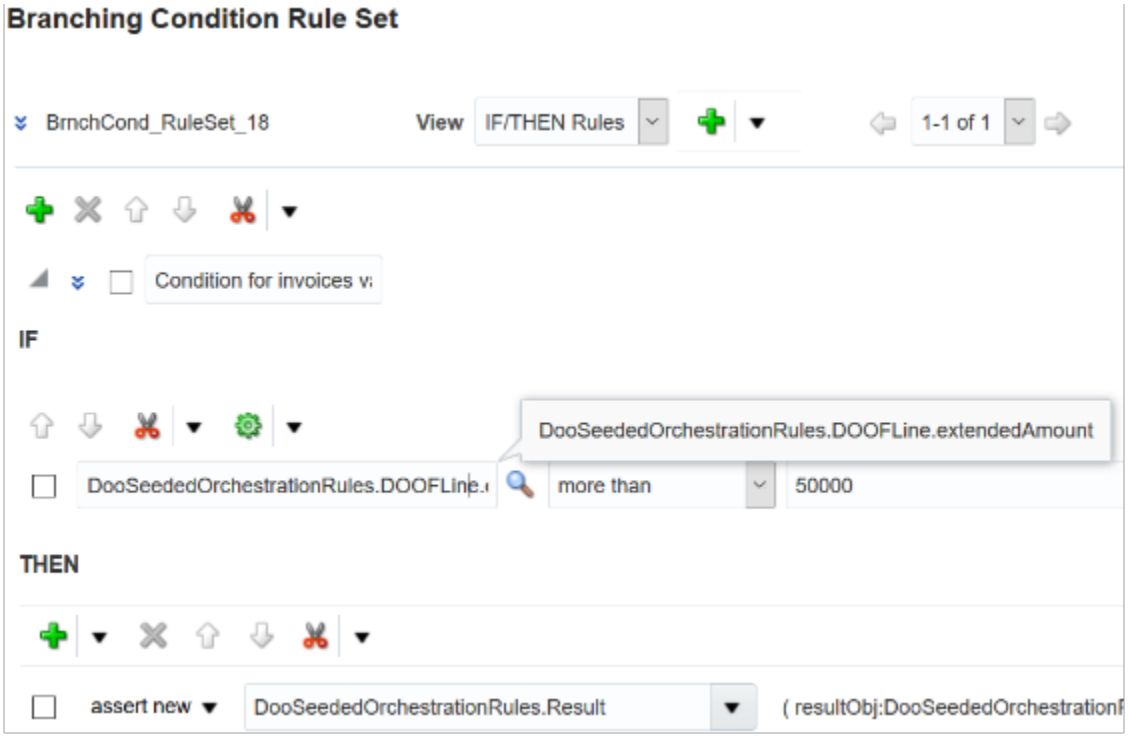
11. In the Step Definition list, click the Otherwise Step, Create Invoice step, then add another step. This step creates the Wait for Invoice Step on the Otherwise branch.

Attribute	Value
Step Name	Otherwise Wait for Invoice
Step Type	Service
Task	Otherwise Create Invoice
Service	Wait for Billing
Exit Criteria	Billed

12. In the Step Definition list, click the Wait for Invoice step, then add another step. This step merges the branch back to the main flow.

Attribute	Value
Step Name	Merge
Step Type	Merge

Create the If Statement



Start by creating the If statement.

- If the extendedAmount attribute on the fulfillment line is more than 50000

Do it.

1. In the Step Definition list, in the Phone Customer step, in the Branching Condition column, click **Click for Rule**.
2. In the Branching Condition Rules dialog, click **Add Rule**, then click **Expand**.
3. Delete the value `Rule 1` , and then enter `Condition for invoices valued at more than 50000 dollars.`
4. Click **Left Value**.

5. In the Condition Browser dialog, expand **DooSeededOrchestrationRules > DOOFLine**, click **extendedAmount**, then click **OK**.

   where

Code	Description
`DooSeededOrchestrationRules`	A dictionary that includes predefined rule sets, facts, functions, variables, bucket sets, links, and functions that you can use to orchestrate fulfillment.
`DOO`	Means distributed order orchestration, which is an earlier name for order orchestration.
`FLine`	Fulfillment line.
`extendedAmount`	A fulfillment line attribute that stores the total value of the sales order.

6. Click **Is**, and then click **More Than**.
7. Click **Right Value**.
8. In the Condition Browser dialog, enter `50000`, then click **OK**. Don't include commas in your value.

Create the Then Statement

You will create the Then statement.

- `( resultObj:DooSeededOrchestrationRules.Boolean.TRUE )`

where

Code	Description
`resultObj`	A variable in the DooSeededOrchestrationRules dictionary. You use it to store the result of the business rule.
`Boolean`	Sets the value of resultObj to TRUE.

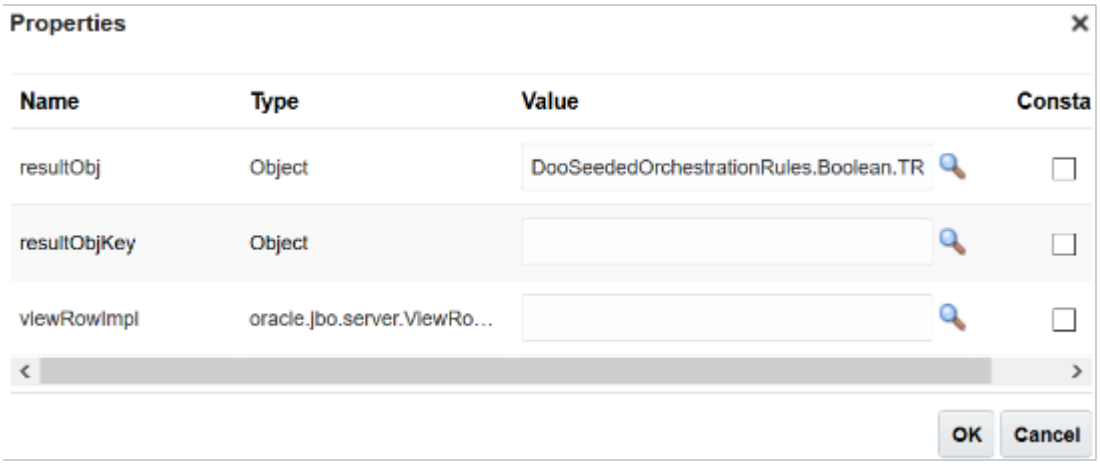
Do it.

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target**, then click **DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Properties dialog, in the ResultObj row, click **Value**.
5. In the Condition Browser dialog, expand **DooSeededOrchestrationRules > Boolean**, click **True**, then click **OK**.

6. Make sure the Properties dialog contains these values.

Name	Type	Value
resultObj	Object	DooSeededOrchestrationRules.Boolean.TRUE
resultObjKey	Object	Leave this cell empty.
viewRowImpl	oracle.jbo.server.viewRowImpl	Leave this cell empty.

For example:



7. Click **OK**.
8. In the Branching Condition Rules dialog, click **Save**.
9. On the Edit Orchestration Process Definition page, click **Save**.

Test Your Set Up

1. Verify that you correctly defined the orchestration process steps and flow.
    - In the header of the Edit Orchestration Process Definition page, click **Actions**, then click **Generate Process Diagram**.
    - Make sure the diagram includes the same steps and logic that the diagram at the beginning of this topic displays.
2. Test the nonbranching flow.
    - In the Order Management work area, create a sales order that's valued at less than $50,000.00.
    - Verify that Order Management ships the item without requesting that the user call the customer.

3. Test the branching flow.
    - Create a sales order that's valued at more than $50,000.00.
    - Verify that Order Management doesn't ship the item until the activity that the Wait for Phone Customer step references reaches a Completed state.

Related Topics

- Set Up Orchestration Processes
- Manage Routing Rules

Deploy

Deploy Orchestration Processes

Release and deploy an orchestration process that you create so its available throughout Order Management.

If you use one of these predefined orchestration processes, then you must deploy it, but its not necessary to release it.

- ShipOrderGenericProcess
- ReturnOrderGenericProcess

Summary of the Steps

1. Release the orchestration process.
2. Deploy the orchestration process.

Release the Orchestration Process

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the orchestration process definition you must release, then click **Actions > Edit**.
3. On the Edit Orchestration Process Definition page, click **Actions > Release**.

Order Management does these steps.

1. Validates orchestration process logic to make sure it correctly created the process.

Validation	What Happens Next
Doesn't find any errors.	Order Management continues the release. An orchestration process that doesn't contain errors is valid.

Validation	What Happens Next
	A valid orchestration process might include warning messages, but these messages don't stop the release from proceeding.
Finds errors.	Order Management stops the release and displays an error icon next to the orchestration process name. It keeps these errors until the next time it runs the validation. You must resolve errors before you can continue the release.

2. Finishes the validation.
3. Updates the status of the orchestration process to Released.
4. Makes the orchestration process definition read-only.
5. Creates and saves the BPEL artifacts that Order Management uses to deploy and run the orchestration process in a production environment.

You can release more than one version of an orchestration process in a single day. You must reject the previous version before you can release the next version on the same day.

Deploy the Orchestration Process

1. Click **Actions > Deploy Process**.
2. In the Deploy Process dialog, enter your password, then click **Deploy**.

If you can't access the Manage Orchestration Process Definition page to deploy the orchestration process for some reason, then you can use Oracle Middleware to deploy it. For details, see *Use Oracle Middleware to Deploy Orchestration Processes*.

Related Topics

- Set Up Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Use Oracle Middleware to Deploy Orchestration Processes

Use Oracle Middleware to Deploy Orchestration Processes

If you can't use the Manage Orchestration Process Definition page to deploy an orchestration process for some reason, then use Oracle Middleware to deploy it.

Summary of the Steps

1. Release the orchestration process. For details, see *Deploy Orchestration Processes*.
2. Download the orchestration process.
3. Modify the SOA configuration plan.
4. Deploy the JAR file.

Download the Orchestration Process

You deploy the downloaded artifacts to the server. You use Oracle Setup Manager to export the artifacts, and Oracle Middleware to deploy them.

1. On the Manage Orchestration Process Definitions page, select the orchestration process you must deploy, then click **Edit**.
2. On the Edit Orchestration Process Definition page, click **Actions > Release**.
3. In the Download Generated Process dialog, click **Download**.
4. Save the archive file that Order Management displays to a local directory.
5. Open the archive file you saved.

Modify the SOA Configuration Plan

You use an Oracle Service-Oriented Architecture (SOA) configuration plan to define the URL and properties to use for different environments. Order Management uses the plan to search the SOA project for values it must replace so the project supports each environment.

Modify this SOA configuration plan.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAConfigPlan
xmlns:jca="http://platform.integration.oracle/blocks/adapter/fw/metadata"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"
xmlns="http://schemas.oracle.com/soa/configplan">
<composite name="*">
<import>
<searchReplace>
<search/>
<replace/>
</searchReplace>
</import>
<service name="client">
<binding type="ws">
<attribute name="port">
</attribute>
</binding>
</service>
<reference name="*">
<binding type="ws">
<attribute name="location">
<searchReplace>
<search>http://localhost_am:port</search>
<replace>http://actualDOOADFserver:port</replace>
</searchReplace>
<searchReplace>
<search>http://localhost_soa:port</search>
<replace>http://actualDOOSOAserver:port</replace>
</searchReplace>
</attribute>
</binding>
</reference>
</composite>
</SOAConfigPlan>
```

Note

- Replace each host name with the Oracle Application Development Framework (ADF) server.

- Replace each port with the port that your organization uses for Order Management and the server and port that Supply Chain Management uses for SOA. Use the external-facing URLs of the servers.

Deploy the JAR File

Use one of these tools.

- Oracle Enterprise Manager Middleware Control
- Ant command line tool
- Oracle WebLogic Scripting Tool

The JAR file resides in a Deploy folder. The Deploy folder resides in a folder that uses the name of the orchestration process you downloaded.

For details about how to deploy an SOA composite application, see Oracle Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite.

Related Topics

- Deploy Orchestration Processes

Migrate Orchestration Processes in Order Management

Migrate your orchestration processes from one environment to another.

Here's a summary of how its done.

Note

- Assume you created an orchestration process named CustomDOO_MyProcess in your test environment and need to migrate it to your production environment.
- There are dependencies between the orchestration process and other objects that you import. Pausing the import for the orchestration process object allows the import to finish processing these dependencies. You then resume the import.
- Click Yes to any dialogs that display during this procedure.

Summary of the Set Up

1. Export.
2. Import.

Export

Do this in your test environment.

1. Release and deploy CustomDOO_MyProcess in your test environment.
   For details, see *Deploy Orchestration Processes*.
2. Create an implementation project.
   - Go to the Setup and Maintenance work area, then click **Tasks > Manage Implementation Projects**.
   - On the Implementation Projects page, click **Actions > Create**.
   - On the Create Implementation Project page, set the value, then click **Next**.

Attribute	Value
Name	My Implementation Project

     On the Select Offerings to Implement page, in the Order Management row, set the value, then click **Save and Open Project**.

Attribute	Value
Include	Selected

   - On the My Implementation Project page, click **Done**.
   - On the Implementation Projects page, notice that the search results displays your My Implementation Project, then click **Done**.
3. Create a configuration package.
   - On the Setup page, click **Tasks > Manage Configuration Packages**.
   - On the Manage Configuration Packages page, click **Actions > Create**.
   - On the Create Configuration Package page, set the values, then click **Next**.

Attribute	Value
Name, in the Source Implementation Project area	My Implementation Project
Setup Task List and Setup Data	Enabled
Name, in the Configuration Package Details area	My Configuration Package

- On the Select Objects for Export page, click **Deselect All**, then sort the Name column in alphabetic order.
- Scroll to the row that has Orchestration Process in the Name column, then click that **row**.
- In the row you just clicked, set the value.

Attribute	Value
Export	Contains a check mark.

- Scroll to the bottom of the page, then, in the Orchestration Process area, click **Actions > Select and Add**.
- In the dialog that displays, search the Name attribute for CustomDOO_MyProcess, then click **Apply > Save and Close**.
- On the Create Configuration Package page, click **Submit**.

4. Monitor and download the configuration package.

- On the Manage Configuration Packages page, in the search results, click the **row** that has My Configuration Package in the Name column.
- In the Export and Import Processes area, click **Actions > View Process Results**.
- On the Export and Import Process Results page, monitor the Status attribute at the top of the page until it changes to Completed Successfully. If necessary, click **Refresh** to periodically update the page. It might take a few minutes.
- Click **Done**.
- On the Manage Configuration Packages page, in the Export and Import Processes area, click **Refresh** (the circular arrow).
- In the Download column, click the **icon**, then click **Download Configuration Package**.
- Save the My Implementation Project_1.zip file to your computer.

  Notice that the download appends some text to the file name, such as My Implementation Project 1_1_20210811_2348.zip.

Import

Do this in your production environment.

1. Upload your configuration package.

- Go to the Setup and Maintenance work area, then click **Tasks > Manage Configuration Packages**.
- On the Manage Configuration Packages page, Click **Actions > Upload**.
- On the Upload Configuration Package, browse to the My Implementation Project_1.zip file that you downloaded earlier, click **Get Details**, wait until the details display, then click **Submit**.

2. Upload your orchestration process.

- In the Export and Import Processes area, click the **row** that has the value.

Attribute	Value
Name	My Configuration Package

Attribute	Value

- Click **Actions > Import Setup Data**.
- On the Enter Basic Information page, click **Next**.
- On the Select Pauses for External Import page, click **Actions > Deselect All**, then sort the Name column in alphabetic order.
- Scroll to the row that has Orchestration Process in the Name column, then click that **row**.
- In the row you just clicked, set the value.

Attribute	Value
Pause	Enabled

- Click **Next > Submit**.

3. Monitor and finish the upload.
   - In the Export and Import Processes area, click the **row** that has My Configuration Package in the Name attribute, then click **Actions > View Process Results**.
   - On the Export and Import Process Results page, monitor the Status attribute at the top of the page until it changes to User Action Required. If necessary, click **Refresh** to periodically update the page. It might take a few minutes.
   - Click **Done**.
   - On the Manage Configuration Packages page, in the Export and Import Processes area, click **Refresh** (the circular arrow), click the **row** that has your configuration package, then click **Actions > Resume**.
   - On the Resume Import Setup Data page, click **Next > Next > Submit**.
   - On the Manage Configuration Packages page, in the Export and Import Processes area, click **Refresh**, click the **row** that has your configuration package, then notice the value in the Status column. Keep refreshing until it says Completed Successfully.
   - At the top of the page, click **Refresh** until the status at the top of the page says Completed Successfully.

Related Topics

- Deploy Orchestration Processes

Assign

Guidelines for Assigning Orchestration Processes

Create an assignment rule that assigns an orchestration process to one or more fulfillment lines of a sales order.

- Assign the orchestration process according to your unique requirements.

- Order Management doesn't create different versions of an assignment rule so the changes that you release take effect immediately. You can save rules without releasing them.
- Use the Otherwise logic to assign an orchestration process that will apply by default. Use it for each orchestration group.
- Create an assignment rule that references data from a different product model. For details, see *Get Data from Product Information Management*.
- You don't need to specify a version or effective date for an assignment rule because the orchestration process controls them.
- Use the rules editor to help you filter the attributes that you can use in your assignment rule. For details, see *Use Tools and Environments to Create Business Rules*.

Note

- You must set up your orchestration processes before you create your assignment rule.
- Order Management can't reassign an existing order line to a different orchestration process when you revise a sales order. For details, see *Fix Problems That Occur When Assigning Orchestration Processes*.
- If a group of fulfillment lines are part of a shipment set, then you can't assign different orchestration processes to different lines in the set. For example, if fulfillment line w and fulfillment line x are part of a shipment set, then you can't assign line w to process y and line x to process z.
  Order Management assigns the first line that it encounters in the set to an orchestration process, and then assigns all subsequent lines in the set to the same orchestration process. If you create an assignment rule that attempts to assign a subsequent line in the set to a different orchestration process, then Order Management will ignore that rule. Order Management uses this configuration to make sure it can schedule, reserve, and ship the lines together.
- Don't create an assignment rule that assigns a fulfillment line to an orchestration process when you revise a sales order. If you need to reassign a fulfillment line to an orchestration process that's different from the one that the fulfillment line currently uses, then you must cancel the sales order or the order line and create a new one.

Examples

Assign an Orchestration Process According To	Description
Item	Each sales order for the AS54888 Desktop Computer must use the same processing steps. You write an assignment rule that assigns an orchestration process named AS54888 Desktop Computer Process to each order line that includes the AS54888 in the Item attribute of the order line.
Customer	Customer Computer Service and Rentals requires an inspection step for each sales order. You write an assignment rule that assigns an orchestration process named Computer Service and Rentals Process to each order line that includes Computer Service and Rentals in the Customer attribute of the order header.
Destination	Each sales order that the fulfillment system must ship to a country that resides outside of your current location requires different handling, such as completing customs forms. You write an assignment rule that assigns an orchestration process named International Orders Process to each order line that includes a foreign country in the Ship-To Address attribute of the order header.

Assign an Orchestration Process According To	Description

Use Orchestration Groups

An orchestration group is a collection of fulfillment lines that Order Management processes together. Each orchestration group is a subset of a sales order. You can assign an orchestration process for an orchestration group.

- You create an orchestration group as a shipment set, configured item, or set of order lines.
- Your assignment rule processes each orchestration group at runtime, so you must add a test that links each fulfillment line with the group. The predefined assignment rules already include this test.

Here are the predefined groups that you can use.

Orchestration Group	Groups All Fulfillment Lines That Fulfill
Standard	A standard item.
Configured Item or Kit	A configured item or kit.
Shipment Set	A shipment set.

Predefined Assignment Rules

Order Management comes with predefined assignment rules. To view them, go to the Manage Orchestration Process Assignment Rules page, then click **View Predefined Rules**.

Predefined Rule	Description
CreateSTDGroups	Assign the orchestration process when the fulfillment line has an item that isn't configured or that's configured but that the user has finished configuring.
AddSTDLines	Add a fulfillment line when you assign an orchestration process according to lines that have items that aren't configured or that are finished items.
CreateShipSets	Assign an orchestration process according to fulfillment lines that have a shipment set.
AddShipSetLines	Add a fulfillment line when you assign an orchestration process according to lines that have a shipment set.
CreateModels	Assign an orchestration process according to fulfillment lines that have a configured item.

To view the predefined AssignProcess rule on the Manage Orchestration Process Assignment Rules, set the View attribute to AssignProcess.

If you create your own rule, then consider the predefined rules when you set priority. If you want Order Management to apply your custom rule first, then set the priority higher on your custom rule than the value that's on the predefined rule.

Related Topics

- Use Tools and Environments to Create Business Rules
- Assign Orchestration Processes
- Return Sales Orders

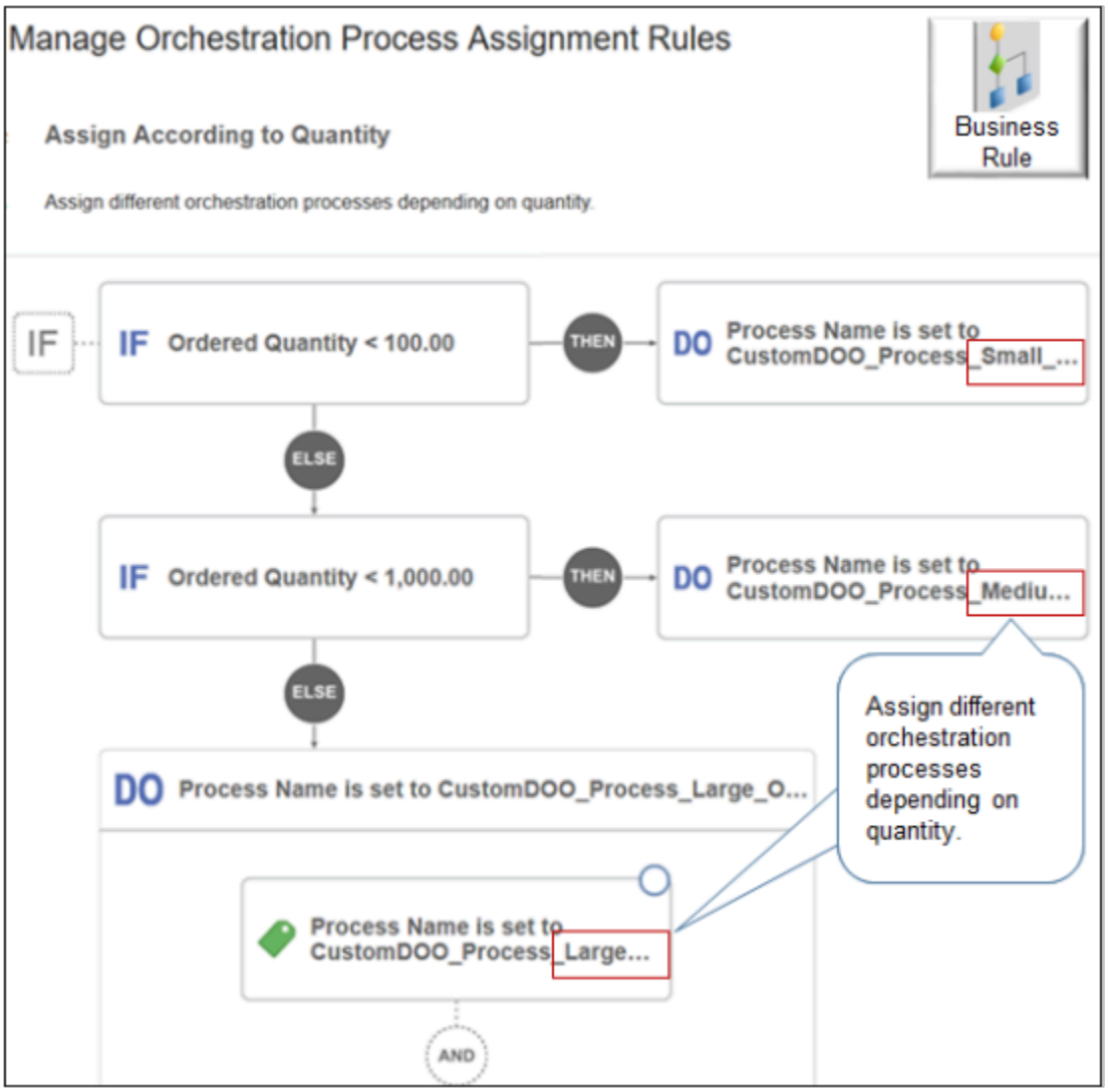
Assign Orchestration Processes

Create an assignment rule that assigns your orchestration process to a fulfillment line.

Assume you must process sales orders differently depending on quantity. Assume you already created these processes.

Orchestration Process Name	Description
Process_Small_Orders	For quantity under 100. Expedite shipping and don't measure the impact on inventory.
Process_Medium_Orders	For quantity 100 to 999. Use regular shipping and measure the impact on inventory only at the distributor, not the warehouse.
Process_Large_Orders	For quantity over 1,000. Use bulk freight. Measure the impact on inventory at the warehouse and the distributor, and call the customer to confirm the order before you ship it.

Here's the assignment rule that you will create.



Summary of the Set Up

1. Create and deploy orchestration processes.
2. Create the assignment rule.
3. Create the small order condition.
4. Create the medium order condition.
5. Create the large order condition.

Learn how to create a business rule. For details, see *Overview of Using Business Rules With Order Management*.

This topic uses example values. You might need different values, depending on your business requirements.

Create and Deploy Orchestration Processes

Assume you already created and deployed these orchestration processes.

- Process_Small_Orders
- Process_Medium_Orders

- Process_Large_Orders

For details, see *Set Up Orchestration Processes*.

Create the Assignment Rule

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Process Assignment Rules for Sales Orders
2. On the Manage Orchestration Process Assignment Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Assign According to Quantity
Description	Assign different orchestration processes depending on quantity.

Create the Small Order Condition

1. Create the If statement.
    - Click **New Condition**.
    - In the Create Condition dialog, enter `ordered`, wait a moment, then click **Ordered Quantity (Order Fulfill Line)**.

      Order Fulfill Line indicates that the attribute resides on the order fulfillment line.
    - Set the operator to `less than ( < )`.
    - Enter `100`, then click **OK**.
2. Create the Do statement.
    - Click **Then > Do > New Action**.
    - In Create Action dialog, enter `process`, wait a moment, then click **Process Name (Order Fulfill Line)**.

      The phrase `Order Fulfill Line` indicates that the orchestration process that you set will process order fulfillment lines.
3. Search for your orchestration process.
    - Click **Search > Advanced**.
    - Set Process Name to Contains.
    - Enter `Process_Small_Orders`.

      The search is case sensitive.
    - Click **Search**.

Note:

If you didn't deploy your orchestration process, then the search won't find it.

- Click the **row** in the search results.
- Click **OK**.
- In the Create Action dialog, click **OK**.

Create the Medium Order Condition

Do the same steps that you used when you created the small order condition, but with these differences.

- Click **Else > If**.
- In the If statement, enter `1000`.
- In the Do statement, set Process Name to Process_Medium_Orders.

Create the Large Order Condition

Do the same steps that you used when you created the medium order condition, but with these differences.

- Click **Else > Do**.
- Don't create an If statement.
- In the Do statement, set Process Name to Process_Medium_Orders.

Make sure you activate and publish your rule.

Related Topics

- Use Visual Information Builder
- Add Branches to Orchestration Processes
- Roadmap for Setting Up Order-to-Cash

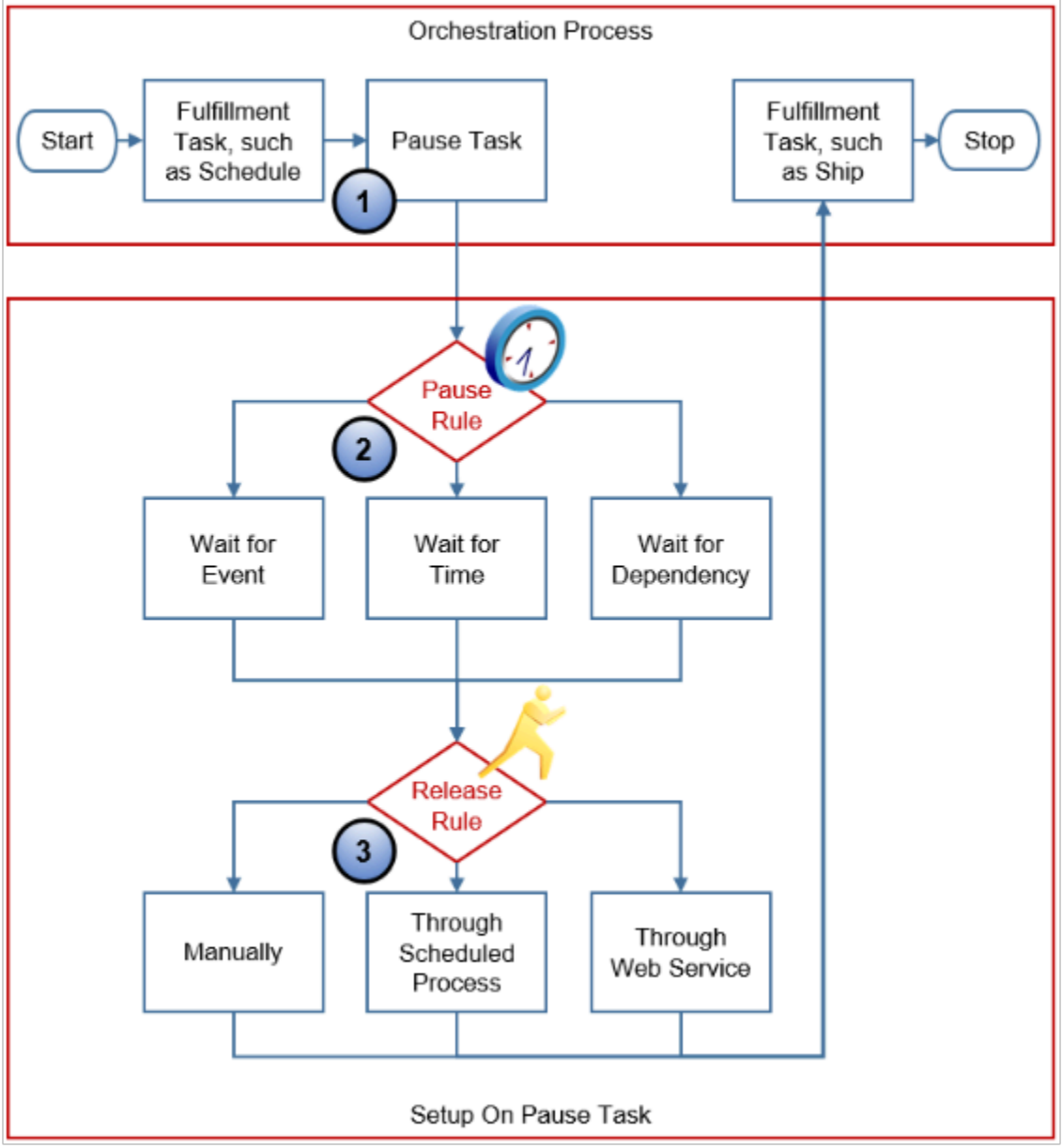
Pause

Overview

Overview of Pausing Orchestration Processes

Use a pause task to temporarily stop an orchestration process from running so it can wait to meet a condition.

When it meets the condition, the orchestration process releases the pause task, then proceeds to the next orchestration process step. You can specify a pause task to wait between tasks, or specify when to release the pause and begin the next orchestration process step.



Note

1. You add a step that references a pause task in an orchestration process. You can add a pause task step at any point in the orchestration process. This example includes a simple three step orchestration process. For example, assume the orchestration process successfully scheduled fulfillment, and now must wait to confirm payment before shipping the item to your customer.
2. You set up a rule that pauses the orchestration process to wait for an event to occur, time to elapse, or for a dependency to resolve.

3. You set up a release rule that releases the pause task so the orchestration process can continue to the next step. Here's how you release a pause task.
    - Release it automatically when the timer expires, when the flow meets the release condition, or when the flow doesn't meet any of the conditions.
    - Allow a user to release it manually in the Order Management work area.
    - Use a scheduled process that releases it automatically according to a regularly scheduled interval.
    - Use a web service that informs the orchestration process when to release.

Examples

Pause Until An Event Occurs

Assume you provide a layaway program that allows your customer to reserve an item, then pay for it in installments. Use a pause task to pause the orchestration process. When the customer pays in full, the orchestration process calls the Release Paused Tasks service, releases the pause task, then ships the item to the customer.

In another example, assume you must release each sales order to shipping only if inventory is available at the fulfillment location. Use a pause task that pauses the orchestration process until the inventory is available. When the inventory is available, the web service calls Release Paused Tasks, releases the pause task, then ships the item to the customer.

If you set up a rule that waits for an event to occur, then you typically use a web service to resume the pause task. In this context, **event** means something that occurs outside of Oracle Order Management. Its not synonymous with a business event. For details, see *Pause Orchestration Processes Until Events Happen*.

Pause Until Time Elapses

Assume you prefer not to overload the warehouse with shipment requests that aren't due for delivery for two weeks. You can pause the orchestration process to wait to send the shipment request to the warehouse until two days before the scheduled shipment date occurs.

In another example, assume you sell a video game up to seven days before the release date of the game.

- You estimate it requires seven days to schedule and process the sales order before it ships. Use this lead time so you can ship the game as soon as its available.
- You set up a pause task that pauses the orchestration process before the schedule step occurs, and continues to pause until the publisher releases the game.
- You set up a rule that releases the pause task according to an extensible flexfield that includes a value of the release date minus seven days. The orchestration process releases the pause task, schedules the sales order, then resumes processing.

For details, see *Pause Orchestration Processes Until Time Elapses*.

Pause Until a Dependency Resolves

Consider some examples.

- A customer requests to receive a shipment that includes coffee, paper cups, sugar, and creamer at the same time. One fulfillment line fulfills each item. For example, fulfillment line x fulfills the coffee, fulfillment line y fulfills the cups, and so on. You can use a pause task that pauses each order line until the orchestration process finishes scheduling each of these lines. The orchestration process periodically evaluates the sales order. It finishes scheduling all lines, then ships them to the customer at the same time.

- A customer must receive invoices for all fulfillment lines of a sales order at the same time, regardless of when the orchestration process ships each of these lines. You can use a pause task that pauses the orchestration process until it ships all items. It then sends the invoices for all lines to the customer at the same time.
- Your company policy requires people in your organization to review and approve each sales order where the scheduled date occurs after the requested date. You can use a pause task to pause the orchestration process until these folks finish review and approval.

For details, see *Pause Orchestration Processes Until Dependencies Resolve*.

Attributes You Set for the Pause Task

The pause task evaluates the condition in the IF statement of your business rule the first time the rule runs. If the condition is True, then the pause task pauses the orchestration process.

Here are the attributes you set for a pause task.

Attribute	Value
reevaluateFlag	• **Y.** Evaluate the condition every time the orchestration process finishes a step. If the condition doesn't evaluate to true, then release the pause task. For example, assume the orchestration process is processing more than one fulfillment line in a sales order, and that you must evaluate the condition after it processes each line. You can use reevaluateFlag to evaluate each line. For another example, assume you must print pictures for customers and then frame them. You set up a flow that includes another task that prints the picture. You promise same day shipping, so you periodically examine whether the picture printed so you can frame it as soon as possible. • **N.** Don't evaluate the condition every time the orchestration process finishes a task. If the IF statement is True, then the orchestration process remains paused until you explicitly release the pause task.
sacType	Specify the action to take when your If statement evaluates to True. Set sacType to one of these values. • **SAC_TYPE_EVENT.** Pause the orchestration process until an event occurs. • **SAC_TYPE_TIMER.** Pause the orchestration process until the date and time you specify occurs. • **SAC_TYPE_IMMEDIATE.** Release the pause task and resume processing. • **SAC_SYSTEM_EVENT_IPC_PAUSE**. Use an `interprocess communication` (`IPC`) to pause according to an event across orchestration processes. If you use IPC, then the rule will evaluate the condition every time the orchestration process finishes a step even if you set reevaluateFlag to N. Note that `sac` is an abbreviation for **start after condition**.

Related Topics

- Use Case to Pause for an Event
- Pause Orchestration Processes Until Time Elapses
- Manage Order Management Parameters
- Manage Order Attributes That Identify Change

Guidelines for Pausing Orchestration Processes

Use these guidelines to help make sure your pause task works correctly.

Add Step That Has a Pause Task

Add a step that references a pause task.

Note

- Use the Edit Orchestration Process Definition page in the Setup and Maintenance work area.

- Add a pause step. Set these values:

Attribute	Value
Step Type	Service
Task Type	Pause
Task	Pause
Service	Pause Process

- Click the **link** in the Pause Rule column to set up your release rule.
  You typically use line-selection criteria to filter fulfillment lines that the orchestration process step processes. However, line-selection criteria isn't available for a pause task because the pause task already examines every fulfillment line in the sales order, by default.

Don't Specify Evaluation Intervals That Are Too Short

If you set up your pause task so the evaluation interval is too short, then the orchestration process might evaluate the pause condition an excessive number of times and degrade performance. Assume you:

- Create a pause step that pauses the orchestration process until the Scheduled Ship Date happens.
- Set the reevaluateFlag to Y.
- Evaluate the condition one time every five minutes.

Assume five days typically elapse between the time the pause step runs and when the Scheduled Ship Date happens. On average, the orchestration process will evaluate the condition 1,440 times.

- 12 times for each hour, multiplied by 24 hours, multiplied by five days, equals 1,440

This processing consumes a lot of resources, can affect performance, and might result in memory problems that can cause your runtime environment to fail. So, set up your pause step so it evaluates the condition less frequently. In this example, the evaluation could probably happen one time every two hours without affecting the orchestration process in a negative way.

Use the Number of Times to Retry Pause parameter to control the maximum number of times the orchestration process evaluates the condition before it temporarily pauses the pause task and displays an error message informing your users to click Recover to manually recover the pause. For details, see *Manage Order Management Parameters*.

Evaluate Conditions Again

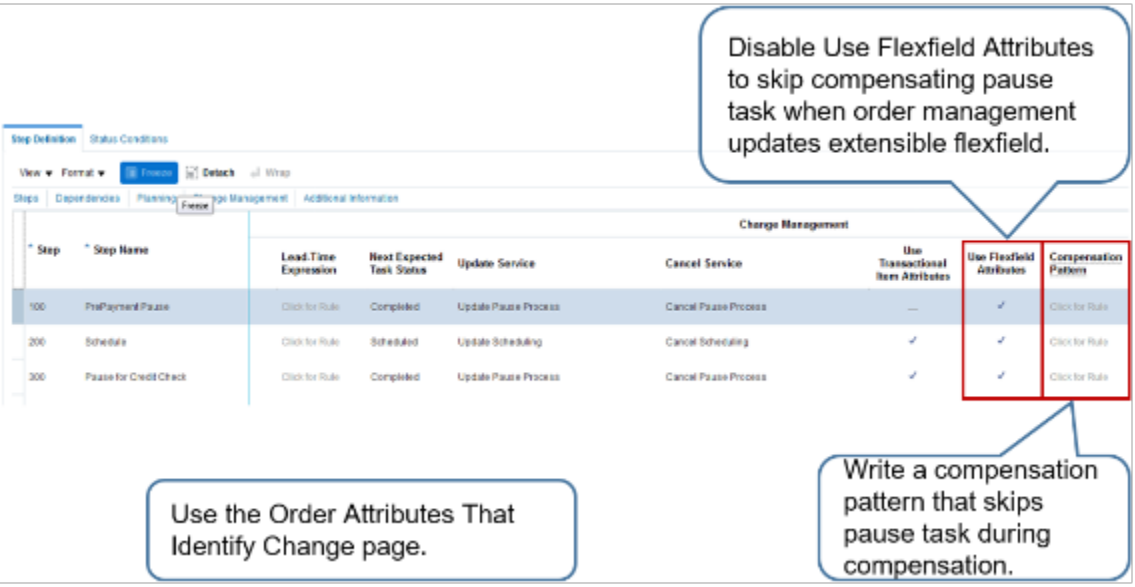
Use the *Recover Errors* scheduled process to evaluate the conditions again. For example:

1. Set up Recover Errors so it runs on the pause task.
2. Set the Number of Times to Retry Pause to 10.
3. At run time, Order Management retries 10 times.
4. If all 10 retries fail, then Order Management places the pause task on the error and displays a message that it exceeded the maximum number of retries.

5. The Order Entry Specialist uses the Recover Order action to recover the sales order.
6. Each order line that's in error retries the pause 10 times.
7. If the condition:
    - **Satisfies during retry.** Order Management releases the pause.
    - **Doesn't satisfy.** Order Management sets the pause task back to an error state.

Pause When Revising Sales Orders

Pause when the Order Entry Specialist revises a sales order. You can set attributes that control this behavior.



Note

- If your pause task doesn't reference an extensible flexfield, then disable the Use Flexfield Attributes option to instruct the orchestration process to skip the pause step when the value in an extensible flexfield changes.
- If you don't need to run your pause task when revising a sales order, then write a compensation pattern.
  Assume you write a pause step that waits for payment. The Order Entry Specialist manually indicates that the customer sent payment, and the pause ends. Some time later, the Order Entry Specialist revises the sales order in a way that doesn't affect the payment amount, such as modifying the ship-to address.
  It isn't necessary to pause to wait for payment after the revision, so you can write a compensation pattern that skips the pause on the payment step.
- The orchestration process uses the attributes that you specify on the Order Attributes That Identify Change page to determine when to compensate while revising a sales order. Use it to make sure your pause task runs only if the revision that the Order Entry Specialist makes affects an attribute that your pause task references.
  For details, see *Manage Order Attributes That Identify Change*.

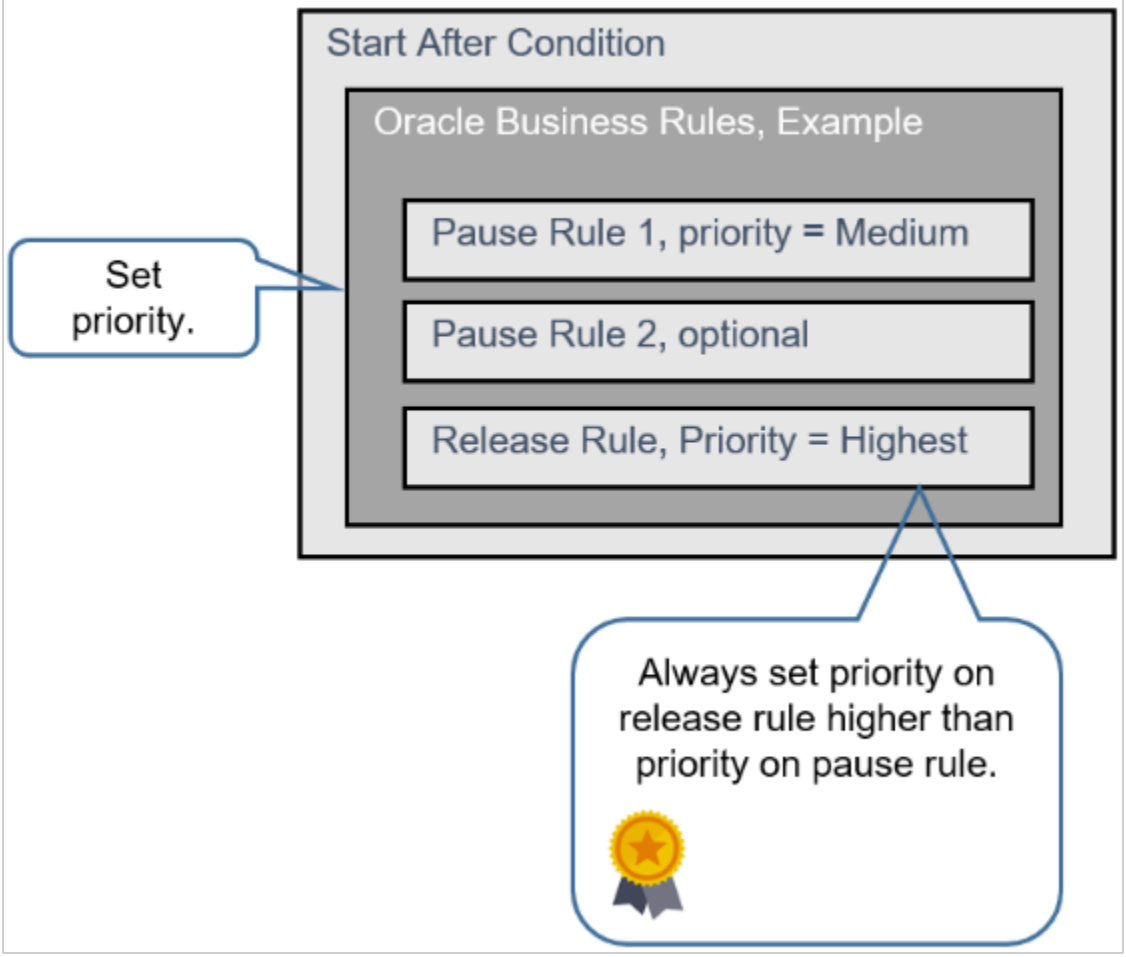
Release Your Pause

If the flow doesn't meet the conditions in the pause rule, then the release rule releases the pause. For example, if the actualShipDate attribute in the pause rule contains a value for each fulfillment line, then:

- All fulfillment lines already shipped.

- The flow hasn't met the pause rule condition that pauses the task.
- The flow is ready to proceed to the next orchestration process step.
- The flow returns to the release rule.
- The release rule sets SacResult to SAC_TYPE_IMMEDIATE.
- The release rule releases the pause.
- The flow continues to the next orchestration process step.

If you set up more than one rule that runs as a set of rules on the same start after condition, then you must set the release rule to the highest priority of all rules in the set.



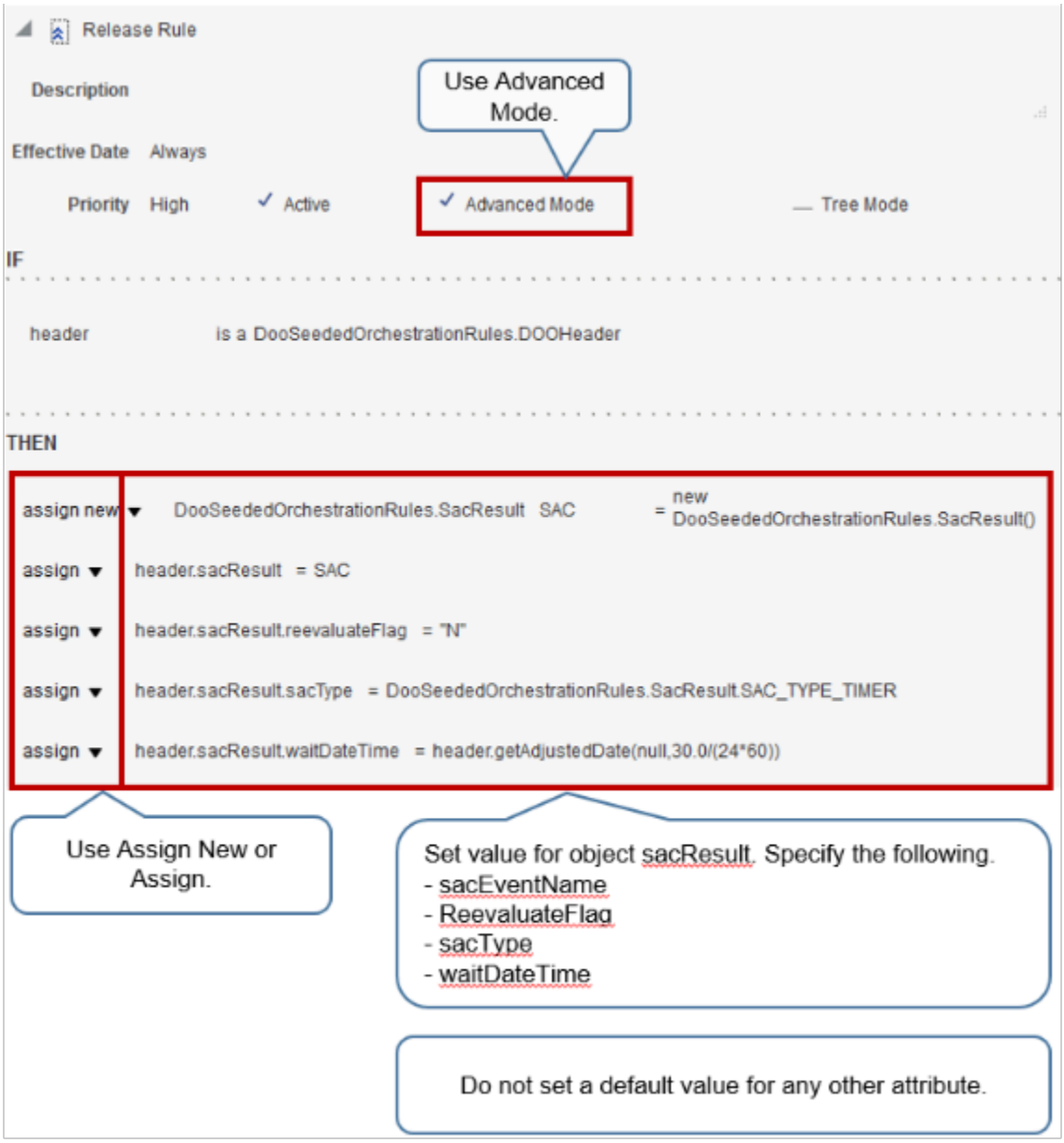
Here's an example of a release rule and a pause rule together.

Note

- Order Management runs the rule with the highest priority first.
- Set the Priority attribute to Highest on the rule where you use Assign New. This makes sure the rule runs first. It also makes sure that this rule creates the object that stores the result of the start-after condition (SAC). Other rules might set the value for the start-after condition.
- If you only have one rule in your rule set, then you don't need to worry about setting priority.
- If you don't set the priority correctly, you might encounter a null pointer error when Order Management runs the rule.
- Don't use the If statement on the release rule to define the condition that your business logic requires. Instead, set up your conditional logic on the pause rule.

- The rule editor requires that you set up a condition on the release rule so it passes validation, so set up a condition that's always true.
- Use Assign New. Assign New creates a new instance of SacResult so you can assign values to it while the rules in your rule set run.

Another Example of a Release Rule



Note

- Use advanced mode.

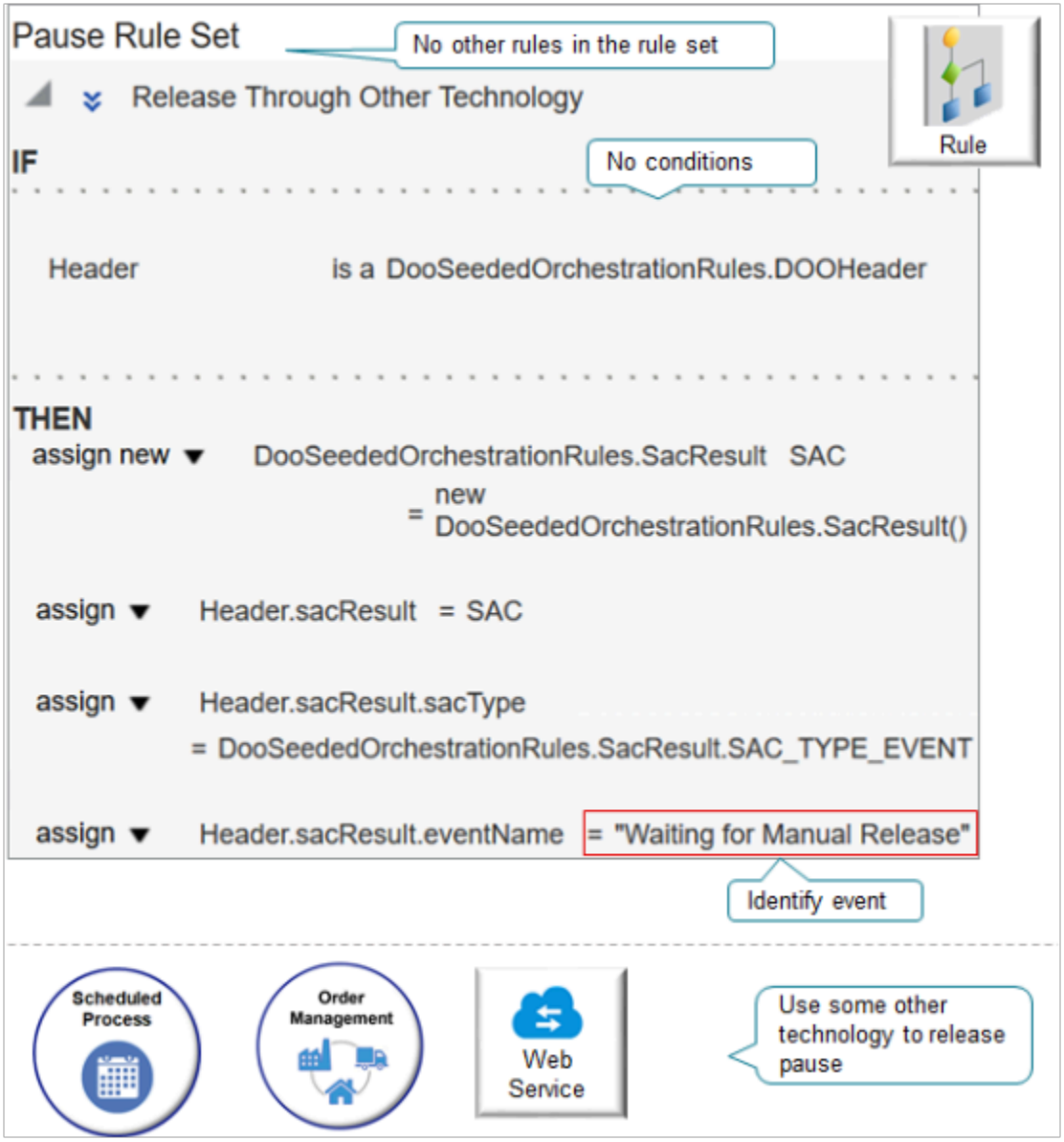
- Specify sacResult.

Attribute	Description
sacType	Required
sacEventName ReevaluateFlag waitDateTime	Optional, depending on the type of pause. For example, if the rule pauses according to a dependency to resolve, then you must include sacEventName. For details, see *Overview of Pausing Orchestration Processes*.

- Don't set default value for other attributes.

Use Some Other Technology to Release Pause

You typically use a release rule to release a pause. However, you can also use a web service, scheduled process, or the Order Management work area to release the pause.



Note

- Use only one rule in your rule set.
- Don't specify a condition.
- Use some other technology to release the pause task.

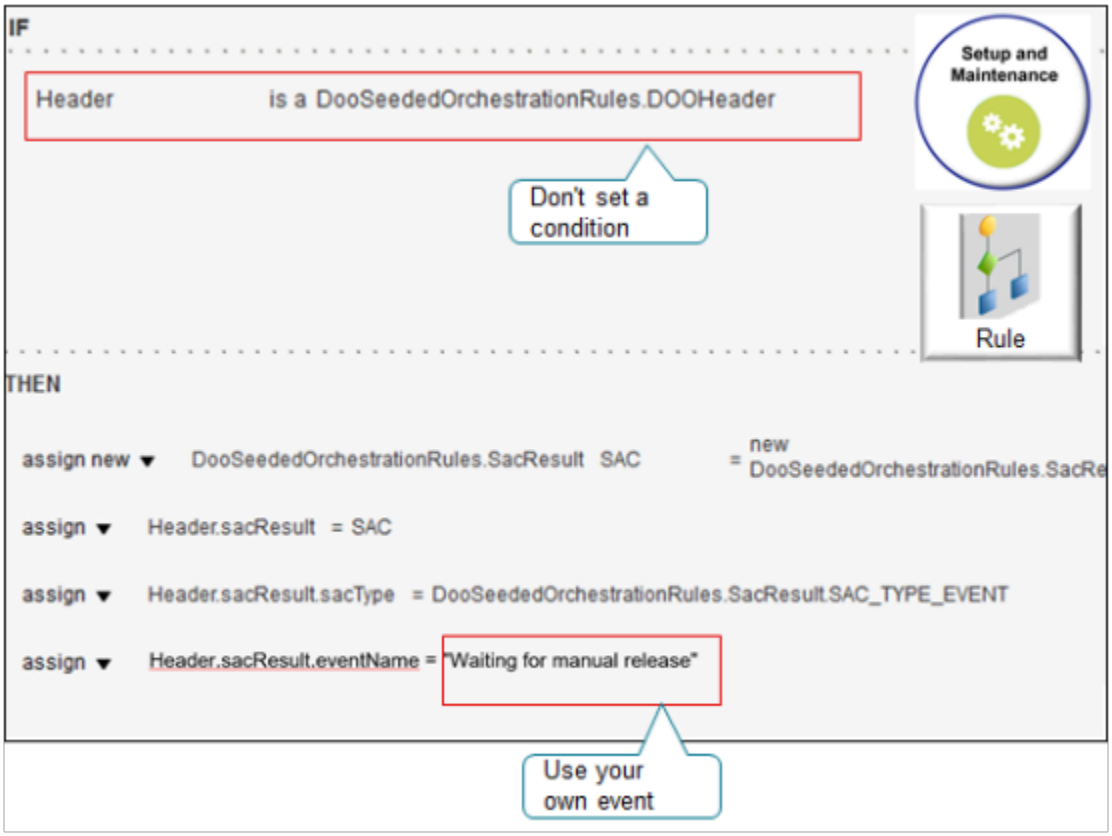
Here's the code.

Code	Description
`Header is a DOOSeededOrchestrationRules.DOOHe`	Declare the Header variable, then store attributes of the order header that the orchestration process is currently processing into this variable.

Code	Description
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.
`Assign Header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_EVENT`	Pause the orchestration process until an event happens.
`Assign Header.sacResult.eventName = "Waiting for Manual Release"`	Specify the event you use to identify the event that's tracking the pause. In this example, assume you named it Waiting for Manual Release.

Don't Include a Condition

If you manually release a pause or release it through a web service or scheduled process, then don't include a condition or default release rule. Here's an example where you send shippable and nonshippable lines together to invoicing, and you want to control when to send these lines.



You can specify your own eventName, such as `Waiting for manual release`. The Order Management work area will display this name. You can also specify it on the scheduled process that you use to release the pause.

Assign the Result

You must include these assignments as the first two actions in your Then statement:

Code	Description
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of sacResult on the order header to the value that SAC contains.

This code creates the object that stores the result of the start-after condition (SAC).

If you don't include these assignments, and if you don't create a release rule, then at least one of the rules in your rule set must use the Assign New action to create the SAC object.

Test Your Set Up

Test your set up in a development environment. Use these conditions.

- Sales order that includes one fulfillment line.
- Sales order that includes more than one fulfillment line. This test helps to make sure you set up your rules so they examine fulfillment lines that the current orchestration process instance isn't examining, but where a dependency exists between these fulfillment lines and the fulfillment line that the instance is examining.
- If your data includes a ship set, then include a ship set in your test.
- If your data includes a configured item, then include a configured item in your test.
- Attributes that don't contain a value.
- Attributes that contain values that result in condition evaluating to false.
- Attributes that contain values that result in condition evaluating to true.
- Order revision.
- Change order.
- If your rules reference them, then test extensible flexfields.

Troubleshoot

Fix Null Pointers When You Use equalsIgnoreCase

Assume you encounter an error that's similar to:

```
Order management can't pause the sales order. oracle.apps.scm.doo.common.DooJboException:
 oracle.rules.rl.exceptions.RLNullPointerException: Attempted to invoke method "equalsIgnoreCase" in
 class "java.lang.String" on a null object reference. at line 6 in /Ruleset(StartAfterCond_RuleSet_18)/
Rule(Rule1)/Pattern(v0_DOOHeader)/Test[1]. For details, see the Guidelines for Pausing Orchestration
 Processes topic on Oracle Help Center
```

This might happen because your pause rule doesn't check to see whether the order header attribute that you're using to determine whether to pause the orchestration process contains a value. Assume you're using the customerPoNumber attribute to pause the process, and you have this statement in your rule.

```
If Header is a DooSeededOrchestrationRules.DOOHeader and Header.customerPoNumber equals ignore case
 "MyValue"
```

Here's how you can fix that:

```
If Header is a DooSeededOrchestrationRules.DOOHeader and Header.customerPoNumber equals ignore case
 "MyValue" and Header.customerPoNumber isn't null
```

Fix Null Pointers When You Use longValue

Assume you encounter an error that's similar to:

```
Order management can't pause the sales order. oracle.apps.scm.doo.common.DooJboException:
 oracle.rules.rl.exceptions.RLNullPointerException: Attempted to invoke method "longValue" in class
 "java.lang.Long" on a null object reference. at line 6 in /Ruleset(StartAfterCond_RuleSet_18)/Rule(Rule1)/
Pattern(v0_DOOFLine)/Test[1]. For details, see the Guidelines for Pausing Orchestration Processes topic on
 Oracle Help Center.
```

This might happen because your pause rule doesn't check to see whether the fulfillment line attribute that you're using to determine whether to pause the orchestration process contains a value. Assume you're using the paymentTermId attribute to pause the process, and you have this statement in your rule.

```
If Fline is a DooSeededOrchestrationRules.DOOFLine and Fline.paymentTermId is 100
```

Here's how you can fix that:

```
If Fline is a DooSeededOrchestrationRules.DOOFLine and Fline.paymentTermId is 100 and Fline.paymentTermId
 isn't null
```

Fix Your Start After Condition

Assume you encounter an error that's similar to:

```
Order management can't pause the sales order. oracle.apps.scm.doo.common.DooJboException:
 oracle.rules.rl.exceptions.RLNullPointerException: Attempted to access property "sacType" in class
 "oracle.apps.scm.doo.common.businessRules.model.obr.facts.SacResult" with a null object reference. at line
 7 in /Ruleset(StartAfterCond_RuleSet_18)/Rule(Rule1)/Action[1]. For details, see the Guidelines for Pausing
 Orchestration Processes topic on Oracle Help Center.
```

This might happen because you haven't correctly created an instance of your start after condition. To fix it, make sure you add an Assign New action that creates an instance of the start after condition first, and then assign that instance to sacResult. For example:

```
assign new DooSeededOrchestrationRules.SacResult SAC = new DooSeededOrchestrationRules.SacResult()

assign Header.SacResult = SAC
```

Next, set the priority to Highest on the rule that has this Assign New action. This priority makes sure Order Management runs the Assign New rule before it runs any other rules. The Assign New action creates an instance of the start after condition that the subsequent rules use at run time. For example, the subsequent rules set the values in that instance. If the instance doesn't exist, then you'll encounter the error.

For details, see the Set the Priority subtopic in *Apply Logic in Business Rules*.

Commonly Used Expressions

ExpressionCode	Possible Values
DooSeededOrchestrationRules.SacResult	SAC new DooSeededOrchestrationRules.SacResult()
Header.sacResult	SAC

ExpressionCode	Possible Values
Header.sacResult.sacType	DooSeededOrchestrationRules.SacResult.SAC_TYPE_IMMEDIATE DooSeededOrchestrationRules.SacResult.SAC_TYPE_EVENT DooSeededOrchestrationRules.SacResult.SAC_TYPE_TIMER
Header.sacResult.eventName	DooSeededOrchestrationRules.SacResult.SAC_SYSTEM_EVENT_IPC_PAUSE You can reference the name of any event that you set up.
Header.sacResult.WaitDateTime	header.getAdjustedDate(null,number of days)

Commonly Used Date Functions

Function	Parameters	Description
`Header.getAdjustedDate`	`Timestamp`, double	Add a number of days, specified in the double format, to the date that your provide in the first parameter. The return value is in the datetime format.
`Header.subtractFromDate`	`Timestamp,Timestamp`	Subtract the second parameter from the first parameter. The return value is the number of days.
`Header.getConvertedDate`	Object	Convert various types of date objects in the datetime format.
`Header.current_date`	Not applicable	Return the system date in the datetime format.

Related Topics

- Use Case to Pause for an Event
- Pause Orchestration Processes Until Time Elapses
- Manage Order Management Parameters
- Manage Order Attributes That Identify Change
- Use Business Rules in Orchestration Processes

Resume Paused Orchestration Processes

You must resume the orchestration process that your pause task paused.

Use a scheduled process or a web service to release a pause task so your orchestration process can automatically resume processing sales orders, depending on how you set sacType.

Value of sacType	Description
SAC_TYPE_TIMER or SAC_SYSTEM_EVENT_IPC_PAUSE	Let the condition evaluate and release the process, or use one of the set ups described in this topic.
SAC_TYPE_EVENT	You must use one of the set ups described in this topic.

Note: You must release a paused orchestration process. If you don't, then the orchestration process will remain in a paused state, which might result in failed sales orders, failed order fulfillment, and so on.

Use a Scheduled Process to Automatically Release a Pause Task

1. Get these privileges.

Privilege Name	Privilege Code
Release Paused Orchestration Order Tasks	DOO_RELEASE_PAUSED_EVENT_TASKS_PRIV
Manage Orchestration Order Modification	DOO_MANAGE_ORCHESTRATION_ORDER_MODIFICATION_PRIV

   You need these privileges to run the *Release Pause Tasks* scheduled process. For details, see *Privileges That You Need to Implement Order Management*.
2. Sign into Oracle applications.
3. Go to the Scheduled Processes work area.
4. On the Scheduled Processes page, click **Schedule New Process**.
5. In the Schedule New Process dialog, set the value, and then click **OK**.

Attribute	Value
Name	*Release Pause Tasks* This scheduled process releases a pause task that's waiting for a business event to finish.

6. Use one of the parameters to specify the orchestration process you must resume, such as Item, Customer, Task Name, and so on.

   For example, if you set up a pause task on an orchestration process that's waiting for the Gold Preorders event, then set the event parameter to Gold Preorders.

   You can also set the Manual Release Type parameter.

Value	Description
Tasks That Exceed Maximum Retries	Use this value for most situations.
Tasks That Can't Establish Wait	Release a pause task that fails to release because of some problem that's internal to Order Management. Set this value only in consultation with Oracle Support.

7. Click **Submit**.

Examine an example that uses this scheduled process. For details, see *Pause Orchestration Processes Until Events Happen*.

Manually Release a Pause Task

1. Make sure you have the privileges that you need to recover errors. For details, see the Order Orchestration Error Recovery Manager chapter in *Security Reference for Order Management*.
2. Go to the Order Management work area, then navigate to the Orchestration Process page.
3. Click **Release Pause Task**.

For details, see *Resume Paused Sales Orders*.

Use a Web Service to Automatically Release the Pause Task

You can use the ReleasePausedTasks operation of the Receive Order Request Service to release a pause task.

Here's your WSDL.

```
https://server.port/soa-infra/services/default/DooDecompReceiveOrderExternalComposite/
ReceiveOrderRequestService?WSDL
```

You can use some of the same parameters that you use with the Release Pause Tasks scheduled process.

- BuyingPartyName
- BuyingPartyNumber
- EventName
- FromOrderDate
- FromPauseWaitUntilDate
- FromScheduledShipDate
- FulfillLineNumber
- FulfillmentOrganizationCode

- FulfillmentOrganizationName
- FulfillOrgId
- InventoryItemId
- LineNumber
- ManualReleaseType
- ManualReleaseTypeName
- OrderNumber
- PausedTaskName
- PauseTaskId
- ProductNumber
- SoldToCustomerId
- SourceOrderNumber
- SourceOrderSystem
- TaskInstanceId
- ToOrderDate
- ToPauseWaitUntilDate
- ToScheduledShipDate

Here's an example payload that releases pauses where the name of the pause task is PauseAfterReservation1. Look at the step in your orchestration process definition that has the pause task to determine the name of the pause task that you need to release.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:dood="http://
xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/receiveSalesOrder/
DooDecompReceiveOrderExternalComposite">
 <soapenv:Header/>
 <soapenv:Body>
 <dood:ReleasePausedEventTasksRequest>
 <dood:SourceSystem>OPS</dood:SourceSystem>
 <dood:PausedTaskName>PauseAfterReservation1</dood:PausedTaskName>
 </dood:ReleasePausedEventTasksRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

You can release a pause for a single order line. Here's a payload that releases the pause task on order line 1 of sales order 506608.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:dood="http://
xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/receiveSalesOrder/
DooDecompReceiveOrderExternalComposite">
 <soapenv:Header/>
 <soapenv:Body>
 <dood:ReleasePausedEventTasksRequest>
 <dood:SourceSystem>OPS</dood:SourceSystem>
 <dood:OrderNumber>506608</dood:OrderNumber>
 <dood:LineNumber>1</dood:LineNumber>
 </dood:ReleasePausedEventTasksRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

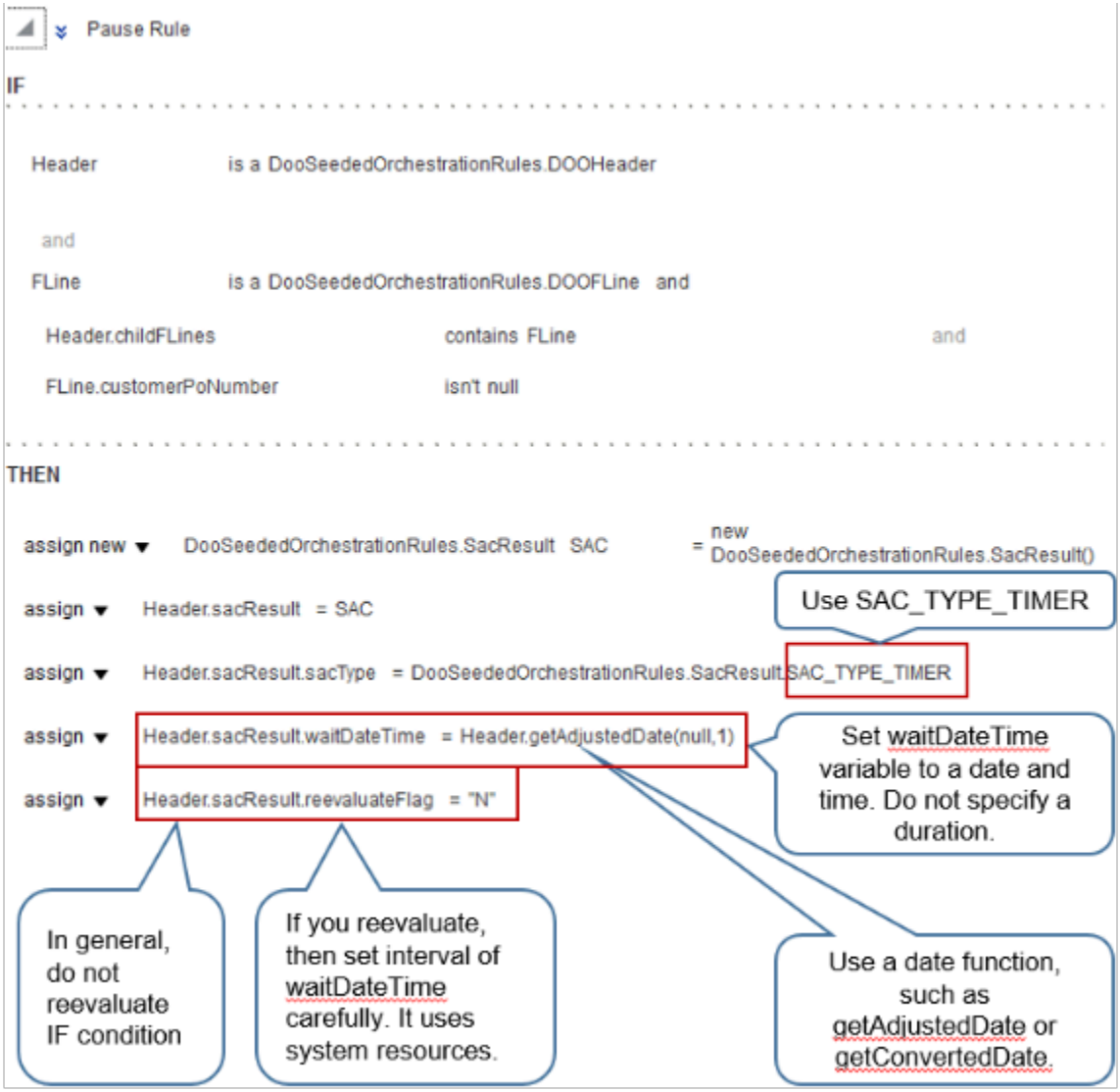
Related Topics

- Use Case to Pause for an Event
- Operations and Attributes You Can Use with the Receive Order Request Service
- Privileges That You Need to Implement Order Management
- Resume Paused Sales Orders

Pause for Time

Pause Orchestration Processes Until Time Elapses

Set up a pause task to temporarily stop an orchestration process from running until date and time elapses.



Note

- Use the Assign action to set the waitDateTime variable.
- Don't specify a duration for waitDateTime. For example, if you specify a duration of 5 hours, then the rule might convert it to a date and time that already occurred, and the rule will never pause the task because the date that it uses to determine whether to pause already occurred. Instead, specify a date and time.

  For example, `Header.getAdjustedDate(null, 1)` specifies to get the current system date, and then wait for `current system date plus one day` before releasing the pause.

  Use dateTime for the first parameter in the function. For example, to wait until two days before the schedule ship date, use `Header.getAdjustedDate(FLine.ScheduleShipDate,-2).`
- reevaluateFlag evaluates the conditions in the IF part of the rule to determine whether the condition changed. Set reevaluateFlag to Y only if you must periodically evaluate your rule. If you set it to Y, then carefully consider the interval you use for waitDateTime. For example, if you set getAdjustedDate to refresh every minute, then reevaluateFlag will cause the rule to run the IF statement every minute until it evaluates the condition in the IF statement to True.

For details, see *Guidelines for Pausing Orchestration Processes*.

Pause Lines Before You Reserve Them

Here's an example where you pause fulfillment lines that Order Management has scheduled, but where the schedule ship date isn't the same as the requested ship date. Assume you set up Global Order Promising so it promises according to on-hand quantity but you don't want to reserve lines when on-hand quantity isn't yet available for them.

Here's the code that determines whether to pause.

Note

Code	What it Means
`Fline.requestShipDate.date less than Fline.scheduleShipDate.date`	If the requestShipDate is less than the scheduleShipDate on the fulfillment line, then continue to pause.
`Fline.requestArrivalDate.date less than Fline.scheduleArrivalDate.date`	If the requestArrivalDate is less than the scheduleArrivalDate on the fulfillment line, then continue to pause.

You can also add a condition to handle the situation where the warehouse is in a different time zone, which might result in the schedule ship date occurring one day later than the requested ship date.

Related Topics

- Overview of Pausing Orchestration Processes
- Use Case to Pause for an Event
- Deploy Orchestration Processes
- Overview of Using Business Rules With Order Management

Pause According to the Current Time

Set up a pause task to temporarily stop an orchestration process from running according to the time on the system clock.

If the current time is.

- **Before 1:10:10 PM**. Pause all fulfillment lines until 1:10:10 PM of the current day, then release them.
- **On or after 1:10:10 PM**. Pause all fulfillment lines until 1:10:10 PM of the next day, then release them.

Here's the rule.

Note

Code	Description
`Header is a DOOSeededOrchestrationRules.DOOHe`	Declare the Header variable, then store attributes of the order header that the orchestration process is currently processing into this variable.
`currentdate is a DOOSeededOrchestrationRules.Times`	Declare the `currentdate` variable, then store the current time stamp from the Oracle server into this variable.
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.

Code	Description
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.
`Assign Header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_TIMER`	Specify the type of pause as a timer.
`Assign new DooSeededOrchestrationRules.Times cdate = Header.current_date`	Declare the **cdate** variable and set its value to the current system time stamp of the Oracle server.
`Assign cdate.hours = 13`	Set the hours of the **cdate** variable to 1PM in the time zone of the Oracle server. Note that the rule uses a 24 hour clock.
`Assign cdate.minutes = 10`	Set the minutes of the **cdate** variable to 10.
`Assign cdate.seconds = 10`	Set the seconds of the **cdate** variable to 10. **cdate** now contains 13:10:10.
`Assign Header.sacResult.waitDateTime = cdate`	Store the value of the **cdate** variable in the waitDateTime parameter of the sacResult object on the order header.
`If cdate.before(currentdate) is true`	If the value in the **cdate** attribute happens before the value of the **currentdate** attribute.
`Assign cdate = Header.getAdjustedDate(cdate,1)`	set the value of **cdate** to **cdate** plus 1 day. The getAdjustedDate function adds the value of the second parameter as the number of days to the first parameter, which is **cdate**.
`Assign Header.sacResult.waitDateTime = cdate`	Store the value of the **cdate** variable in the waitDateTime parameter of the sacResult object on the order header.

Let's consider an example. Assume the current time is 12:00:00, noon.

Code	Example Value
`currentdate is a DOOSeededOrchestrationRules.Times`	Timestamp equals 12:00:00, using format HH:MM:SS on a 24 hour clock. **currentdate** equals 12:00:00.

Code	Example Value
`Assign new DooSeededOrchestrationRules.Times cdate = Header.current_date`	**cdate** equals 12:00:00.
`Assign cdate.hours = 13`	**cdate** equals 13:00:00.
`Assign cdate.minutes = 10`	**cdate** equals 13:10:00.
`Assign cdate.seconds = 10`	**cdate** equals 13:10:10.
`Assign Header.sacResult.waitDateTime = cdate`	waitDateTime equals 13:10:10.
`If cdate.before(currentdate) is true`	**cdate** contains 13:10:10 and **currentdate** contains 12:00:00, so the condition evaluates to false. The rule doesn't proceed to the Then clause of this If statement. Instead, the rule pauses all fulfillment lines until 13:10:10 PM of the current day, then releases them.

Assume the current time is 14:00:00, or 2 PM.

Code	Example Value
`currentdate is a DOOSeededOrchestrationRules.Times`	Timestamp equals 14:00:00, using format HH:MM:SS on a 24 hour clock. **currentdate** equals 14:00:00.
`Assign new DooSeededOrchestrationRules.Times cdate = Header.current_date`	**cdate** equals 14:00:00.
`Assign cdate.hours = 13`	**cdate** equals 13:00:00.
`Assign cdate.minutes = 10`	**cdate** equals 13:10:00.
`Assign cdate.seconds = 10`	**cdate** equals 13:10:10.
`Assign Header.sacResult.waitDateTime = cdate`	waitDateTime equals 13:10:10.
`If cdate.before(currentdate) is true`	**cdate** contains 13:10:10 and **currentdate** contains 14:00:00, so the condition evaluates to True. The rule proceeds to the Then clause of this If statement.

Code	Example Value
`Assign cdate = Header.getAdjustedDate(cdate,1)`	`cdate` equals 13:10:10 plus 1 day.
`Assign Header.sacResult.waitDateTime = cdate`	waitDateTime equals 13:10:10 plus 1 day. The rule pauses all fulfillment lines until 13:10:10 PM of the next day, then releases them.

Related Topics

- Overview of Pausing Orchestration Processes
- Use Case to Pause for an Event
- Deploy Orchestration Processes
- Overview of Using Business Rules With Order Management

Pause for Time with Flexfields

Assume you must limit the time you allow your business partners to provide quotes to 90 days.

You already set up an extensible flexfield with a context named Quote and a segment named quoteexpiration that allows the Order Entry Specialist to use the Order Management work area to specify the expiration date for the quote.

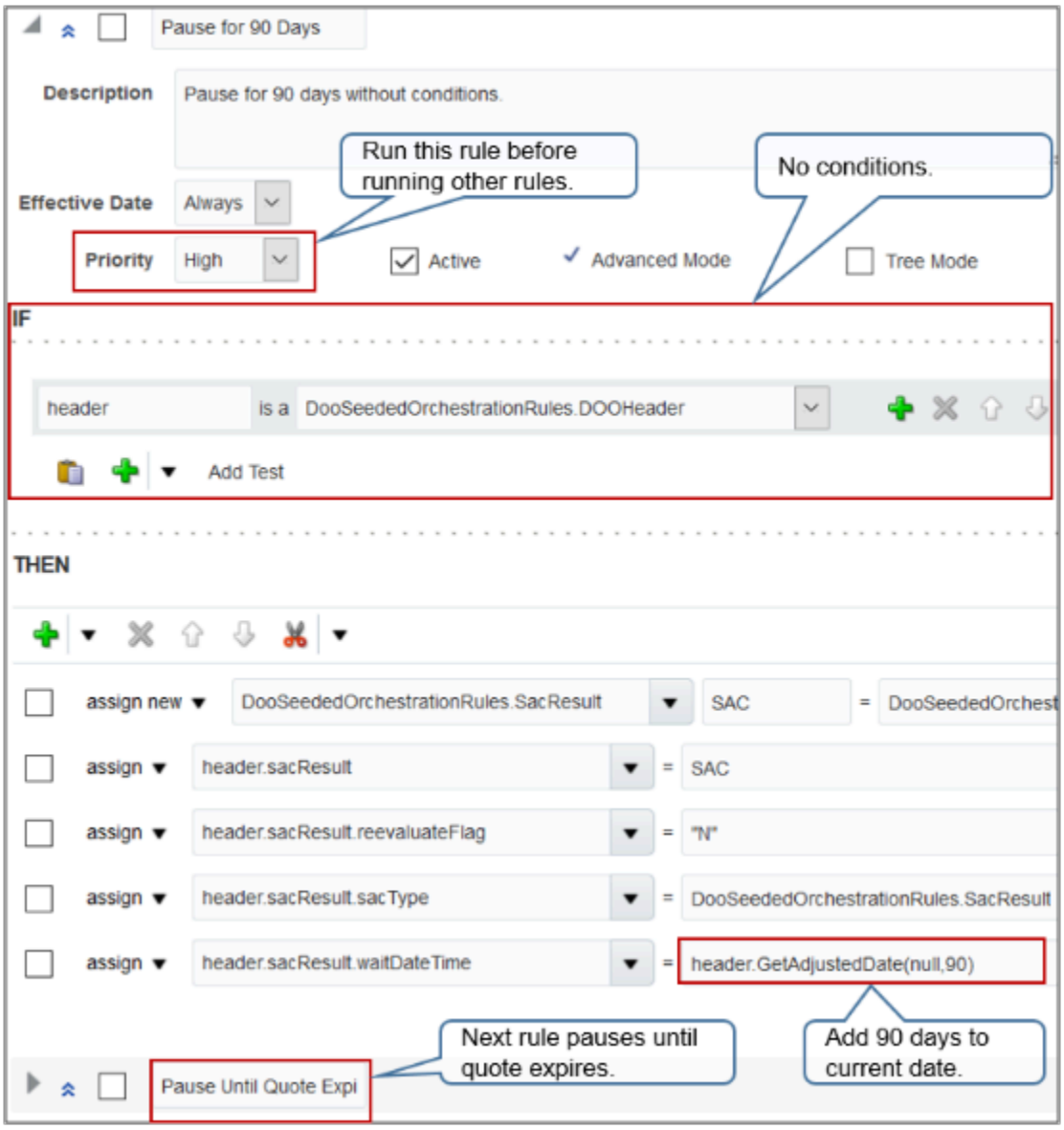
You can create a pause task that expires on the date and time that the expire quote segment specifies. This segment resides in the Quote context of an extensible flexfield.

Here are the rules that you create.

Rule	Description
Pause for 90 Days	Pause for 90 days without conditions.
Pause Until Quote Expires	Pause task that expires on the date and time that the expire quote segment specifies. This segment resides in the Quote context of an extensible flexfield. If Pause Until Quote Expires expires, then it will override Pause for 90 Days.

First Rule

Here's the first rule.



Note

Code	Description
Priority equals High	Make sure you run this rule before you run any other rule.
`Header is a DooSeededOrchestrationRules.DOOHe and`	Declare the Header variable, then load values into Header from the DOOHeader object of the DooSeededOrchestrationRules method. DOOHeader contains order header attributes and their values. This If statement doesn't include conditional logic because we must run the rule in all conditions.
`DooSeededOrchestrationRules.SacRe SAC new DooSeededOrchestrationRules.SacRe`	Create a new instance of the SAC variable, set it to the SacResult type in the DooSeededOrchestrationRules method.

Code	Description
`header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.
`header.sacResult.reevaluateFlag = "N"`	The IF statement doesn't contain a condition that we must reevaluate.
`header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_TIMER`	Specify the type of pause as a timer.
`header.sacResult.waitDateTime = header.GetAdjustedDate(null,90)`	Get the current date, add 90 days to it, and store the result in the waitDateTime parameter of the sacResult object on the order header. The rule won't release the pause until waitDateTime happens.

GetAdjustedDate Function

Here are the parameters that you can use with the GetAdjustedDate function.

- **First parameter**. A date, a date variable, or leave it empty for current date.
- **Second parameter**. Number of days. Use a whole number or a fraction, such as 0.5 for 12 hours.

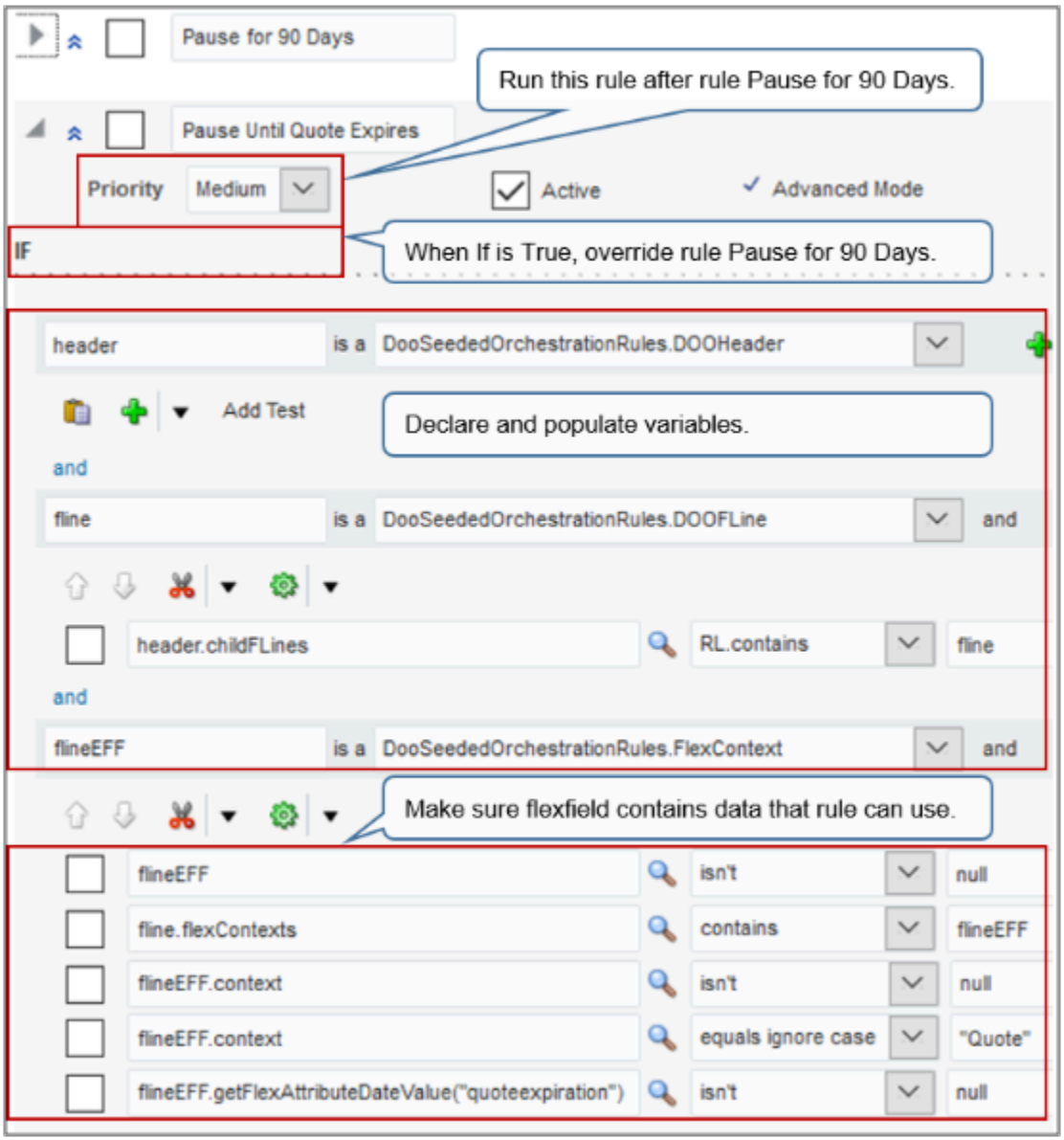
GetAdjustedDate adds the number of days to the first parameter, then returns the result as date and time.

To add minutes or hours, express the second parameter as a fraction.

Format	Example
Number of minutes * 1 / (24 * 60).	To add 30 minutes, use `30 * 1 / (24 * 60)`
Number of hours * 1 / 24.	To add 5 hours, use `5 * 1 / 24`

Second Rule

Here's the If statement for the second rule.

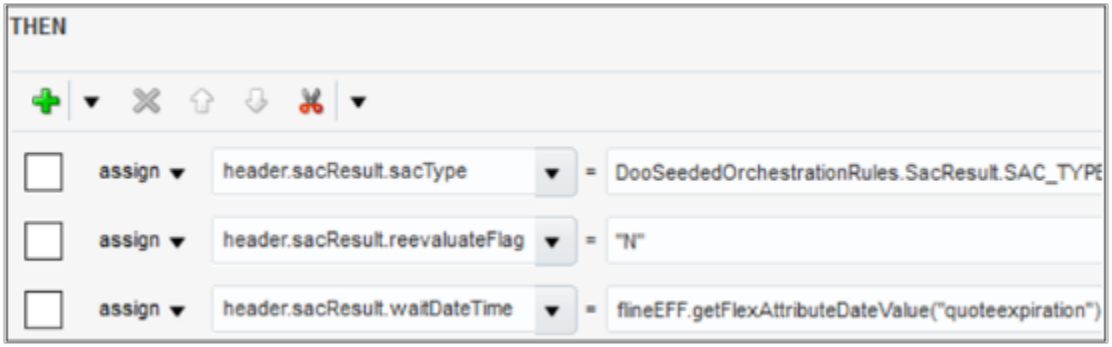


Note

Code	Description
`Priority equals Medium`	Make sure you run this rule after you run the Pause for 90 Days rule.
`If equals True`	If all conditions in the If statement are true, then override the Pause for 90 Days rule.
`Header is a DooSeededOrchestrationRules.DOOHe and`	Declare the Header variable, then load values into Header from the DOOHeader object of the DooSeededOrchestrationRules method. DOOHeader contains order header attributes and their values.

Code	Description
`Fline is a DooSeededOrchestrationRules.DOOFl and`	Declare the Fline (fulfillment line) variable, then load values into Fline from the DOOFline object of the DooSeededOrchestrationRules method. DOOFline (Distributed Order Orchestration Fulfillment Line) contains fulfillment line attributes and their values.
`header.childFLines RL.contains fline`	Declare the fline variable into the rules language (RL) dictionary, then set the value of fline to the value that childFLines contains. This condition makes sure that the fline variable contains a child fulfillment line of the parent sales order that the orchestration process instance is processing. It also makes sure you correctly declare the variable into the dictionary.
`flineEFF is a DOOSeededOrchestrationRules.FlexC`	Declare the flineEFF variable, then store the value of object FlexContext into this variable. Note that the DooSeededOrchestrationRules method contains FlexContext.
`flineEFF isn't null`	Proceed to the next AND only if the flineEFF variable contains a value.
`fline.flexContexts contains flineEFF`	Proceed to the next AND only if flineEFF references an extensible flexfield on the fulfillment line. To reference an extensible flexfield on the order header, use header.flexContexts instead of fline.flexContexts.
`flineEFF.context isn't null`	Proceed to the next AND only if the context contains a value.
`flineEFF.context equals ignore case "Quote"`	Proceed to the next AND only if context equals the string Quote.
`flineEFF.getFlexAttributeDateValu isn't null`	Proceed to the THEN statement only if the Order Entry Specialist enters a value in the flexfield.

Here's the Then statement for the second rule.



Note

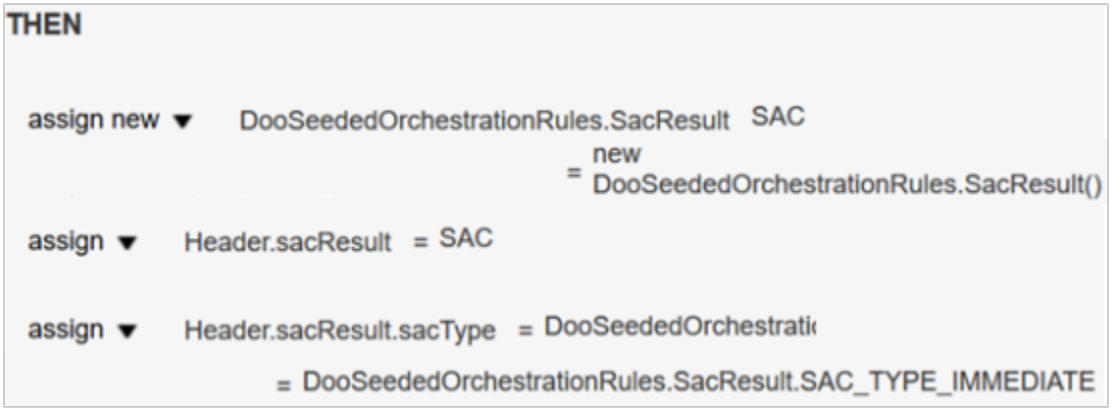
Code	Description
`header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_TIMER`	Set SacResult so it uses the timer.
`header.sacResult.reevaluateFlag = "N"`	Don't reevaluate the condition when the timer expires. This rule doesn't evaluate according to a timer. Instead, it waits until the quote expiration date happens, so reevaluate isn't necessary. If the user submits a revision that includes a new value for the date, then the rule reevaluates the date and time, and sets this new time as part of compensation.
`header.sacResult.waitDateTime = flineEFF.getFlexAttributeDateValu`	Set the waitDateTime variable of the sacResult rule set on the order header to the value that the Order Entry Specialist entered into the flexfield.

Learn how to use business rules with extensible flexfields. For details, see *Overview of Setting Up Extensible Flexfields in Order Management*.

Another Example That Uses Flexfields

Here's an example that pauses according to the value in a flexfield.

```
IF
  Header                is a DooSeededOrchestrationRules.DOOHeader
   and
  FLine                 is a DooSeededOrchestrationRules.DOOFLine  and
    Header.childFLines                          contains FLine
   and
  FLineEFF              is a DooSeededOrchestrationRules.FlexContext  and
    FLine.flexContexts                          contains FLineEFF                 and
    FLineEFF                                    isn't null                        and
    FLineEFF.context                            isn't null                        and
    "FulfillLineContext1"                       equals ignore case FLineEFF.context and
    FLineEFF.getFlexAttributeValue("FL1AttributeChar1")  isn't null               and
    FLineEFF.getFlexAttributeValue("FL1AttributeChar1")  isn't "PauseForMe"
```



Related Topics

- Overview of Pausing Orchestration Processes
- Use Case to Pause for an Event
- Overview of Using Business Rules With Order Management
- Overview of Setting Up Extensible Flexfields in Order Management
- Identify Flexfield Contexts and Category Codes for Your Business Rules

Use Case to Pause for Time

Read this topic to examine a detailed example that describes when to pause for time.

Assume your supply chain analysis determines it typically requires two days to ship goods from warehouse to customer when your customer orders through your legacy system. Your supply chain uses just-in-time fulfillment. You don't want to send the order to shipping too early to avoid the cost of reserving inventory any time longer than necessary to meet the requested delivery date. So you set up a pause task that pauses the orchestration process until two days before the scheduled ship date happens.

You create a business rule.

- If source system is Legacy, then release pause on ShipDate minus two.

Summary of the Set Up

1. Set up orchestration process.
2. Create If statement.
3. Create Then statement.
4. Deploy the orchestration process. For details, see *Deploy Orchestration Processes*.

This topic uses example values. You might need different values, depending on your business requirements.

For background details about.

- Properties you set for the pause task, see *Overview of Pausing Orchestration Processes*.
- Business rules, see *Overview of Using Business Rules With Order Management*.

Set Up Orchestration Process

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management

- Functional Area: Orders
- Task: Manage Orchestration Process Definitions

2. On the Manage Orchestration Process Definitions page, click **Actions > Create**.
3. On the Create Orchestration Process Definition page, set the values, then click **Save**.

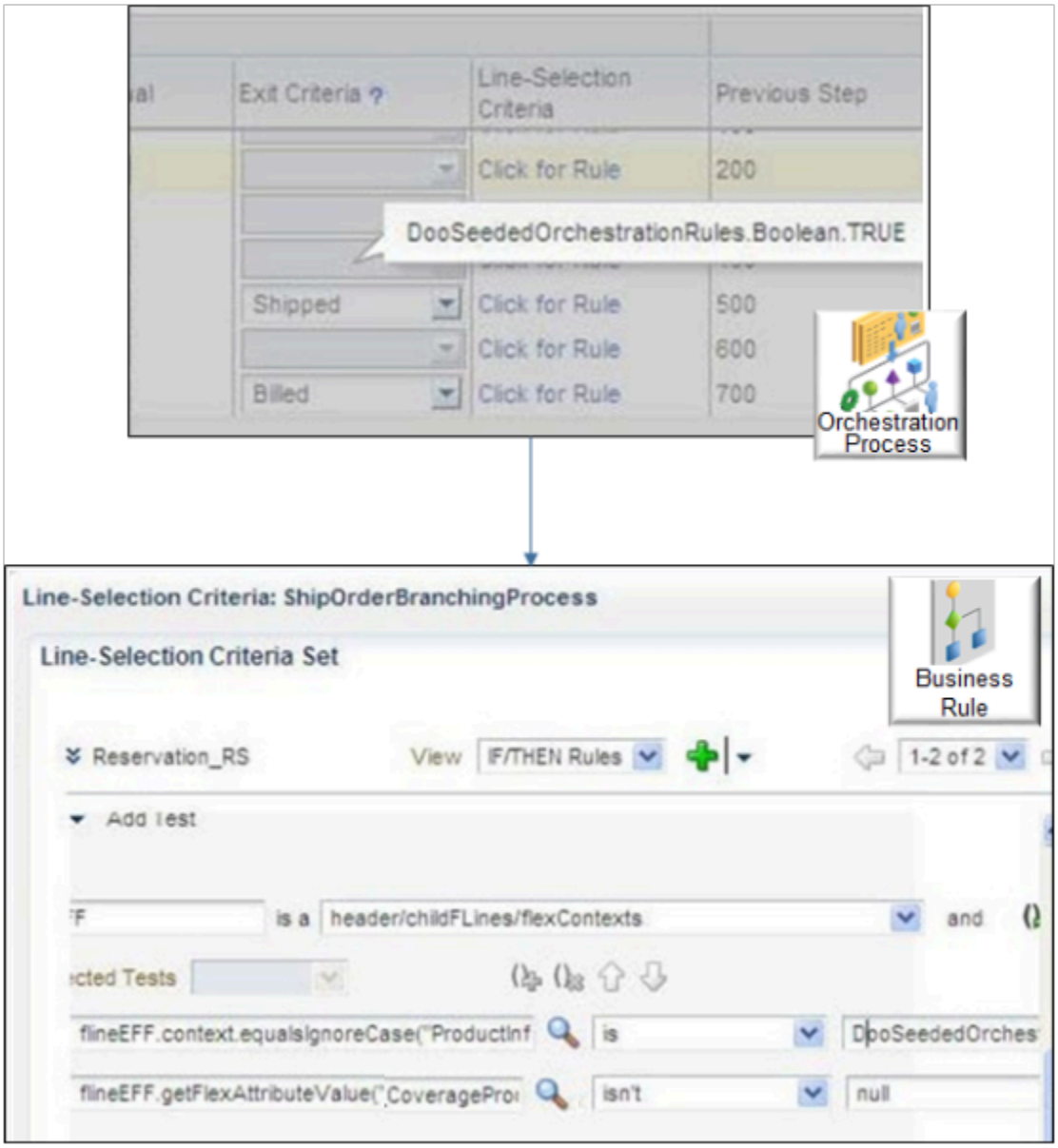
Attribute	Value
Process Name	Pause_for_Time
Process Display Name	Pause_for_Time
Process Class	Ship Order Class
Set	Common Set

4. Set up orchestration process steps.

   Do the stuff that's described in the Define the Orchestration Process Steps subtopic. For details, see *Pause Orchestration Processes Until Events Happen*.

Create If Statement

Create the If statement that determines whether the source system is Legacy.



Create the If statement.

1. Create the rule.
    - On the Pause step, in the Pause Rule column, click **Click for Rule**.
    - In the Start After Condition dialog, click **Properties**, then set the values.

Attribute	Value
Name	Rule That Pauses Until Ready to Ship
Description	If source system is Legacy, then release pause on ShipDate minus two.

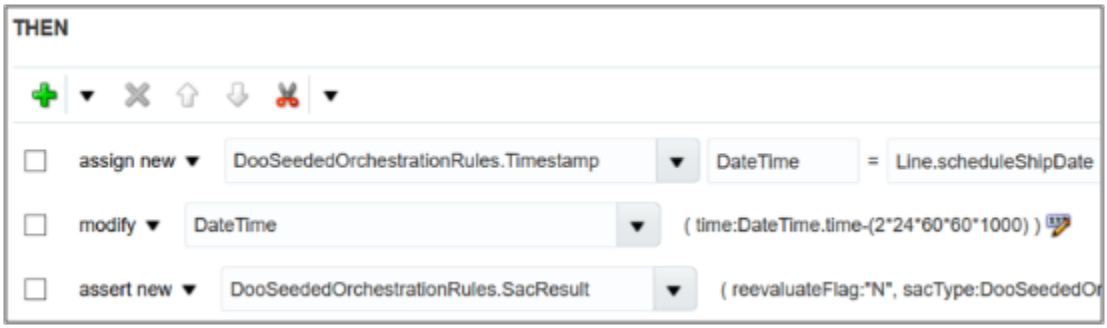
Attribute	Value
Advanced Mode	Contains a check mark.
Tree Mode	Doesn't contain a check mark.

2. Specify the header.
    - In the IF area, in the Left Value attribute, enter `Header`.
    - Set the field that's located to the right of **Is A** to DooSeededOrchestrationRules.DOOHeader.
    - In the IF area, click the **down arrow** next to the magnifying glass, then click **Delete Test**.
3. Specify the line.
    - Click **Add Pattern**.
    - In the left field, under Add Test, enter `Line`.
    - Set the field that's located to the right of **Is A** to DooSeededOrchestrationRules.DOOFLine.
4. Add the test.
    - Immediately under the left field that contains the value Line, click **down arrow > Simple Test**.
    - Click **Left Value**.
    - In the Condition Browser, expand **Header**.

      Notice that the list displays order header attributes that you can use to specify which attribute to examine. For this example, this If statement creates a filter that causes the rule to only consider source orders where the Source Order System attribute contains a value of LEG, where LEG is an abbreviation for the term legacy.
    - Click **sourceOrderSystem > OK**.
    - In the Right Value attribute, enter `"LEG"`.

Create the Then Statement

This statement adds the pause and specifies to release it two days before the scheduled ship date happens.



Create the Then statement.

1. Assign the result.

Code	Description
`Assign new DooSeededOrchestrationRules.Sa SAC = new DooSeededOrchestrationRules.Sa`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.

   For brevity, the screen print doesn't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.
2. Use Assign New to create a variable named DateTime. You will use this variable to calculate the amount of time to delay shipment.
   - In the Then area, click **Add Action > Assign New**.
   - Click **Select a Target > DooSeededOrchestrationRules.Timestamp**.
   - In the empty attribute immediately to the right of Select a Target, enter `DateTime`.

     Don't include quotation marks. Adding DateTime in this attribute creates the variable that this example uses for the date calculation.
3. Select the fulfillment line attribute that you will use to calculate the date.
   - Click **Expression Value**.
   - In the Condition Browser, expand **Line**.

     The dialog displays a list of fulfillment line attributes. You can set up you rule so it considers the value of one of these attributes.
   - Click **scheduleShipDate > OK**.
   - Make sure the attribute where you added DateTime still contains DateTime.
   - Click **Add Action > Modify**.
   - Click **Select a Target > DateTime**.

4. Calculate the amount of time to delay shipment.
    - Click **Edit Properties**.
    - In the Properties dialog, in the Time row, click **Value**.
    - In the Condition Browser, expand **DateTime**, then click **Time**.

      Note that the Condition Browser adds a value of `DateTime.time` to attribute Expression.
    - In the Expression attribute, append `-(2*24*60*60*1000)` to the end of `DateTime.time`.

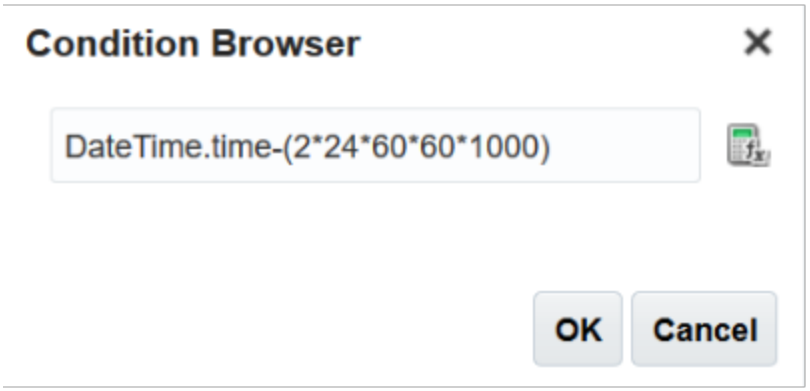
      where

        - 2 is the number of days.
        - 24 is the number of hours in one day.
        - 60 is the number of minutes in one hour.
        - 60 is the number of seconds in one hour.
        - 1000 is the number of milliseconds in one second.

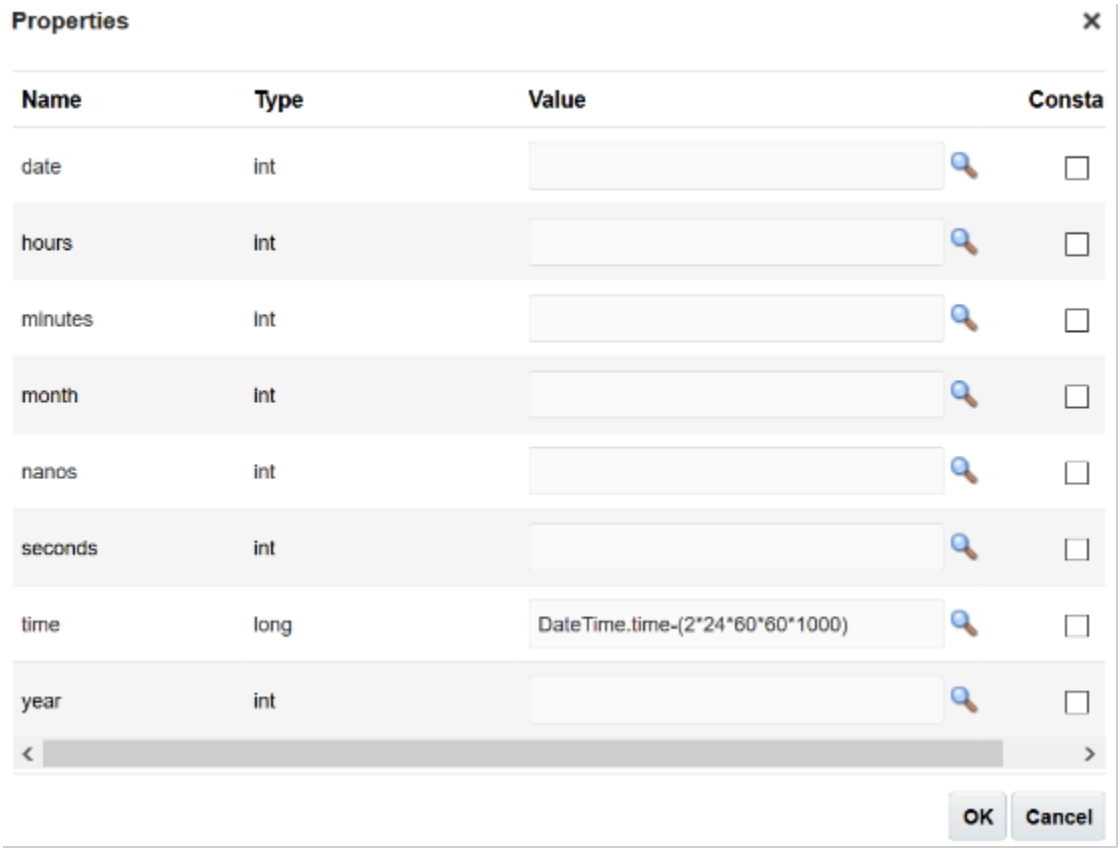
      This value calculates the number of milliseconds so the pause task ends two days before the orchestration process schedules the order lines for shipping.

      You can use the getAdjustedDate instead function of DateTime.time.
    - Make sure the Expression attribute includes `DateTime.time-(2*24*60*60*1000)`, then click **OK**.

      For example:

    - Make sure the Properties dialog includes the value in the time property.


Properties
Name
Type
Value
Consta
date
int
hours
int
minutes
int
month
int
nanos
int
seconds
int
time
long
DateTime.time-(2*24*60*60*1000)
year
int
OK
Cancel

- Click **OK**.

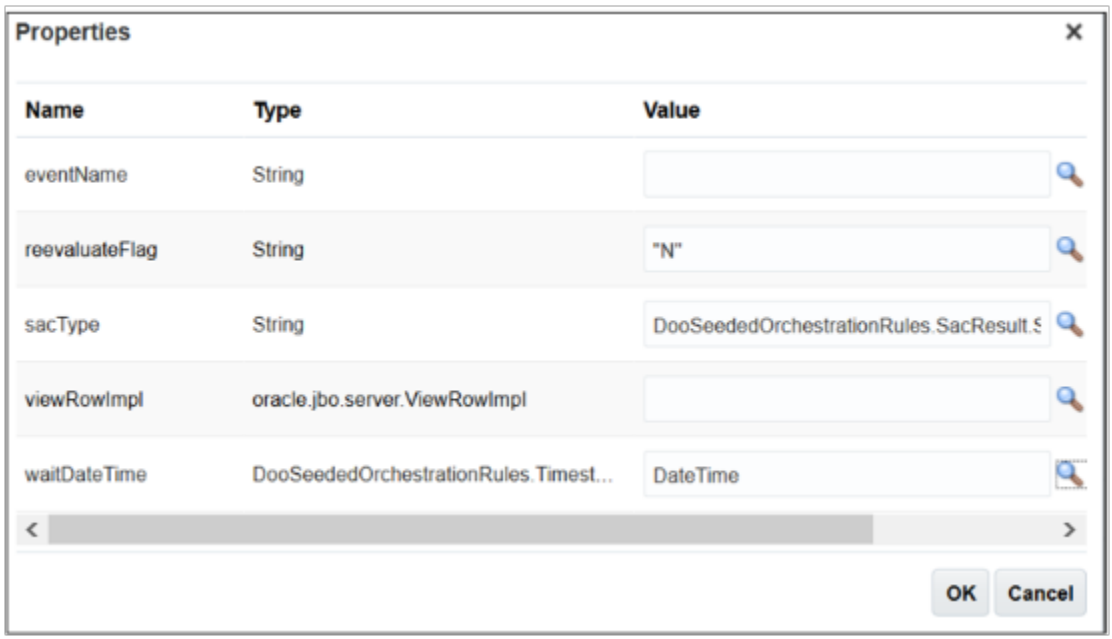
5. Pause the orchestration process.

   You will set these properties.

   ```
   ( reevaluateFlag:"N", sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_TIMER,
    waitDateTime:DateTime )
   ```

   - Click **Add Action > Assert New**.
   - Click **Select a Target > DooSeededOrchestrationRules.SacResult**.
   - Click **Edit Properties**.
   - In the Properties dialog, in the reevaluateFlag row, in the Value attribute, enter `"N"`. You must include the double quotation marks.
   - In the sacType row, click **Value**.
   - In the Condition Browser dialog, expand **DooSeededOrchestrationRules > SacResult**, click **SAC_TYPE_TIMER**, then click .

     A pause type of TIMER causes the orchestration process to pause until time elapses.
   - In the Properties dialog, in the waitDateTime row, click **Value**.
   - In the Condition Browser, click **DateTime**, then click **OK**.
   - Make sure the Properties dialog contains these values.



6. Click **OK**, then, in the Pause Rule dialog, click **Save**.
7. On the Edit Orchestration Process Definition page, click **Save and Close**.

Related Topics

- Overview of Pausing Orchestration Processes
- Use Case to Pause for an Event
- Deploy Orchestration Processes
- Overview of Using Business Rules With Order Management

Pause for Events

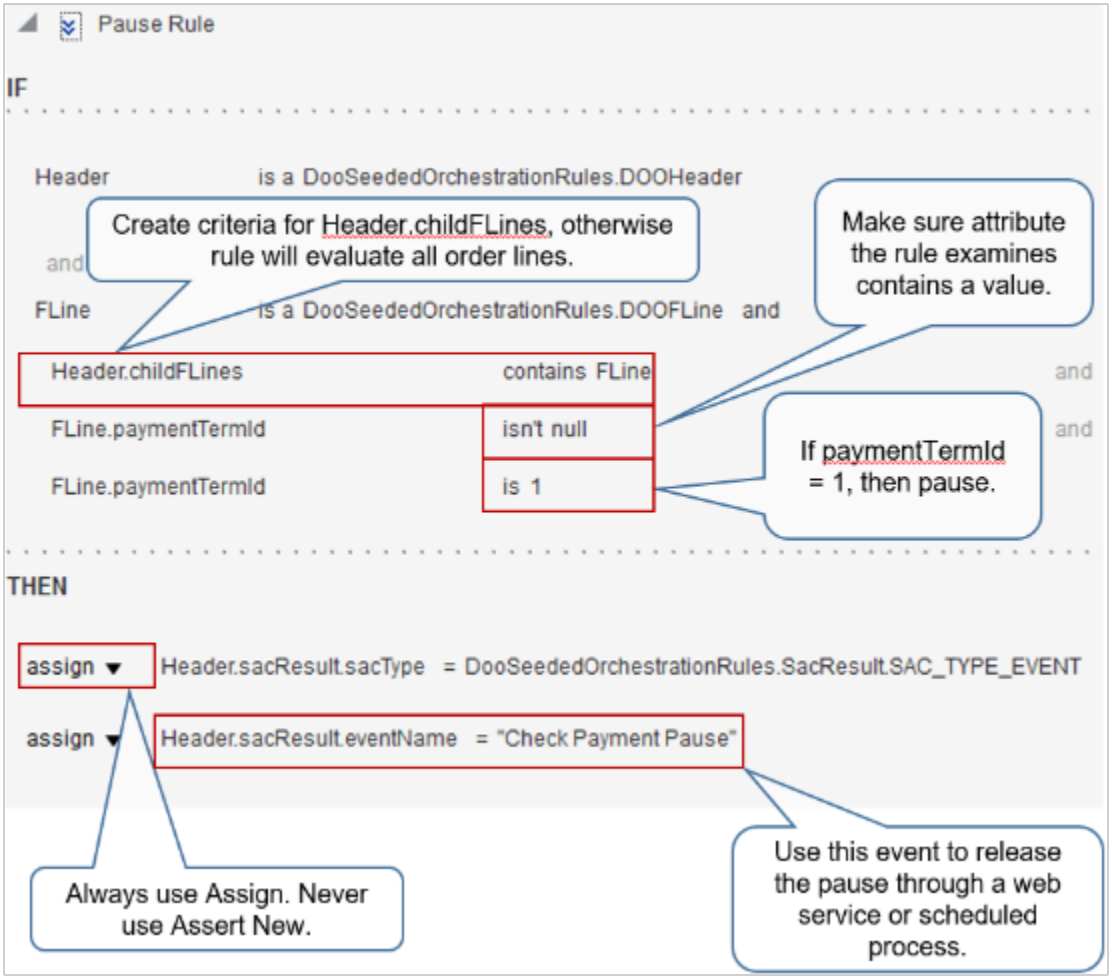
Pause Orchestration Processes Until Events Happen

Use a pause task to temporarily stop an orchestration process from running so it can wait for an event to happen.

Here are some examples.

Wait for a Check Payment Event

Pause processing when a web service sends an event that requests a payment check.



Note

- Assume the application sets paymentTermId to 1 when it requires the orchestration process to pause and makes sure you received payment from your customer.

- Most orchestration processes process one fulfillment line at a time when they pause according to time or an event. If you pause according to time or event, then, to filter fulfillment lines, use one of these codes.

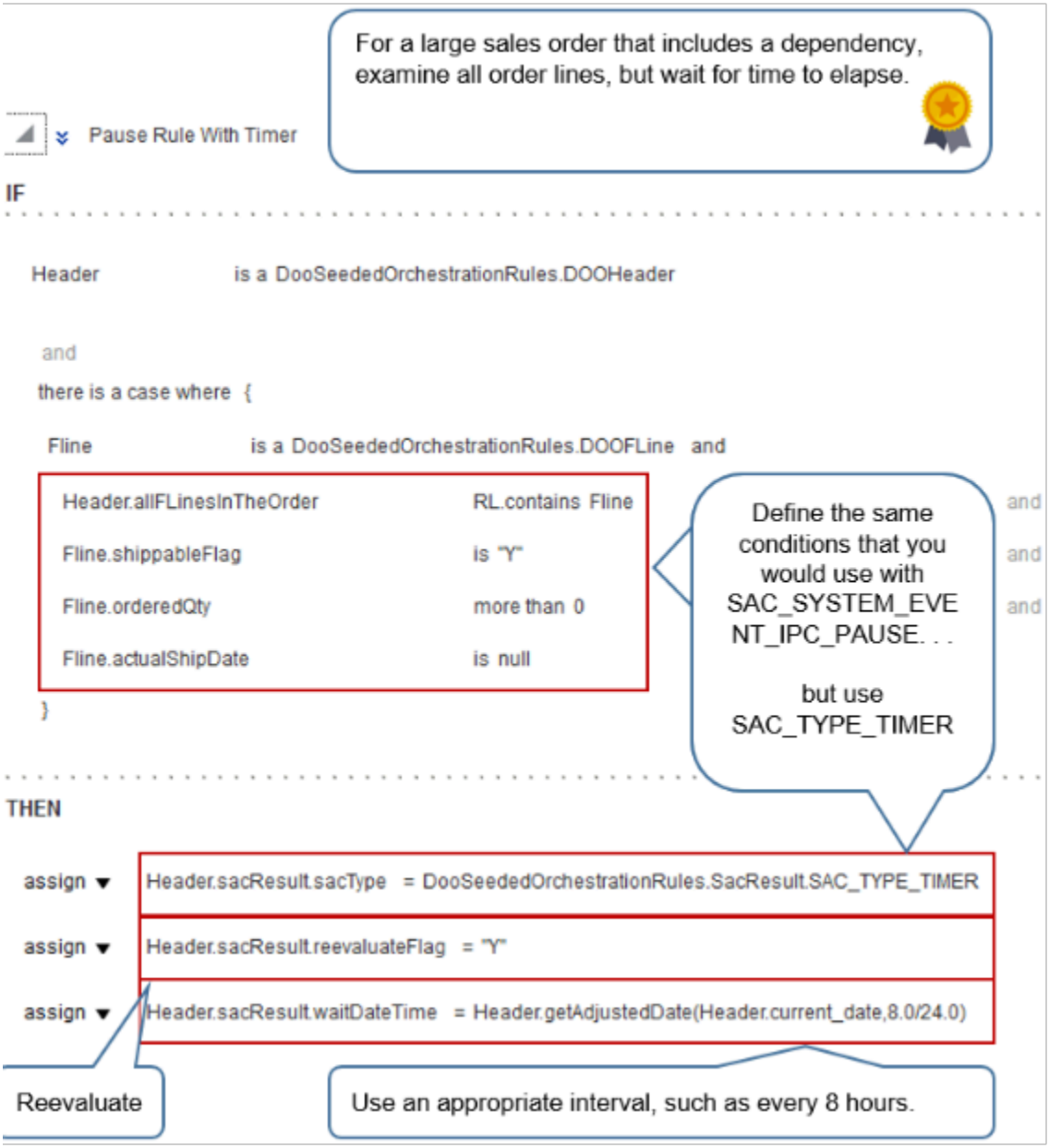
Code	Description
`Header.childFLines`	Reference the fulfillment line that the orchestration process is processing.
`Header.allFLinesInTheOrder`	References all fulfillment lines in the sales order that the orchestration process is processing.

- Use `isn't null`. If your rule examines an attribute that doesn't contain a value, then your rule might fail with results you can't predict.
- Use Assign. If you use Assert New, then the rule creates more than one instance, and it might fail with results you can't predict.
- If you use Assign New for a rule in a pause rule set, then you must set priority for the Assign New rule to Highest, and make sure no other rule in the set is Highest.
- For brevity, this screen print doesn't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.

Wait for Large Sales Orders to Finish Processing

Use SAC_SYSTEM_EVENT_IPC_PAUSE and set reevaluateFlag to Y to cause the rule to examine every orchestration process instance that's in a wait state for the sales order each time an orchestration process step finishes. Assume a sales order includes 100 fulfillment lines, and 75 of them are in a wait state according to SAC_SYSTEM_EVENT_IPC_PAUSE. The rule will reevaluate each of the 75 lines every time it runs.

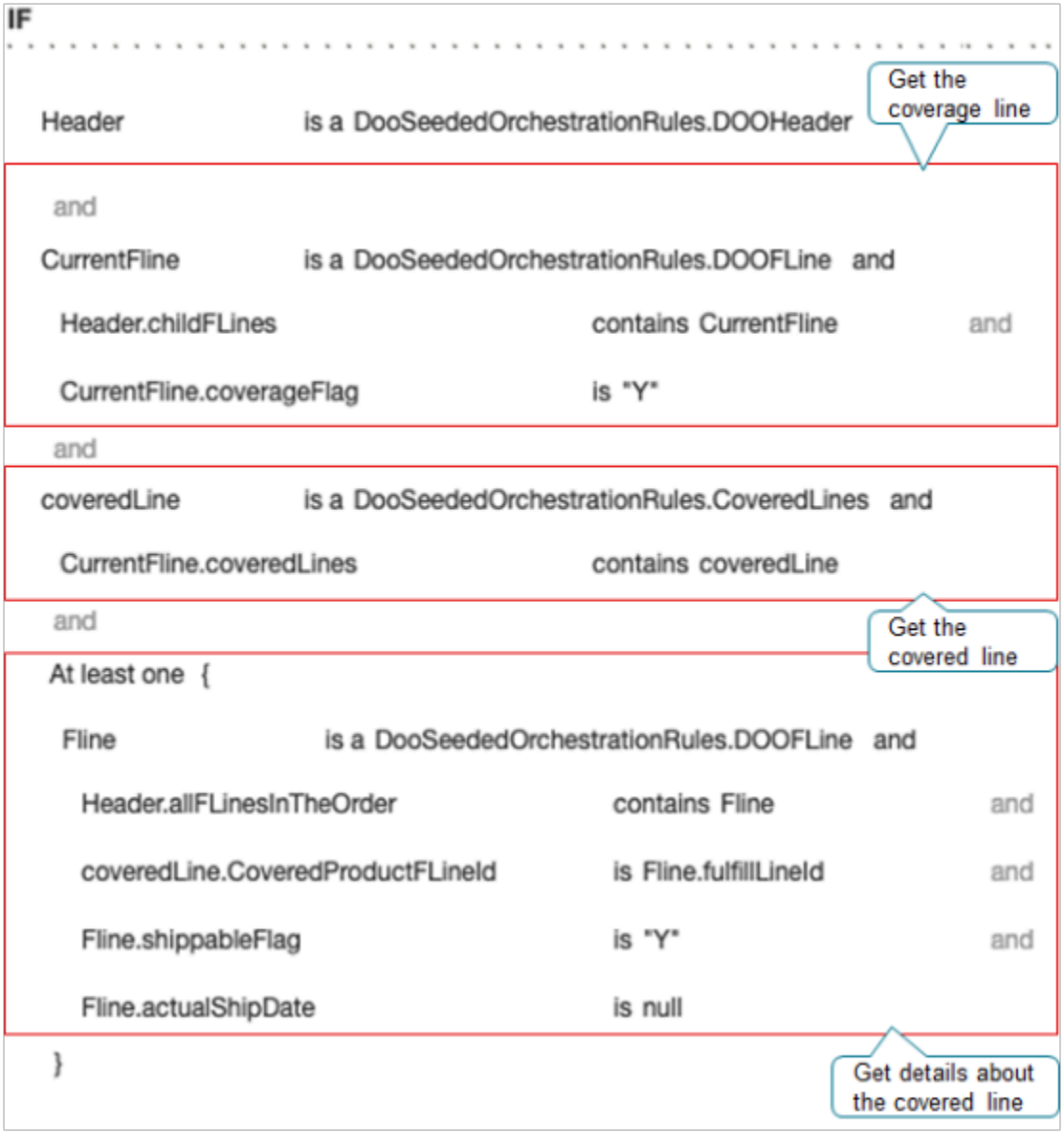
If a sales order includes a large number of fulfillment lines, and if you must pause according to a dependency, then don't use SAC_SYSTEM_EVENT_IPC_PAUSE. Instead, set up a rule that uses SAC_TYPE_TIMER, and that examines all fulfillment lines but only after some time elapses. Specify an appropriate interval. For example, if you determine that most fulfillment lines on a large sales order ship within six hours, and you only need to invoice one time every 12 hours, then use an interval of eight hours.



Send Covered Items and Coverage Items to Invoicing

Send the covered item and the coverage item that covers the covered item to invoicing together. You typically ship the covered item but not the coverage item. For example, you ship the AS54888 Desktop Computer covered item but not the 2 Year Warranty, which is the coverage item for the AS54888, so you pause sending the warranty to billing until order fulfillment finishes shipping the AS54888.

- Use this set up in an orchestration process that does billing for your coverages where one orchestration process instance processes the covered item and another instance processes the coverage item.
- Add this pause task on an orchestration process step before the step that does the invoicing task.
- This rule automatically releases the pause for the coverage item when fulfillment ships the covered item.



Here's the Then statement.

Related Topics

- Overview of Pausing Orchestration Processes
- Guidelines for Pausing Orchestration Processes

Pause for Shipping

Have a look at some examples that you can use to pause an orchestration process while shipping.

For these examples:

- Add your pause task so it happens on a step that happens after the shipping task and before the invoicing task in your orchestration process.
- Use the same Then statement for all examples, except where noted.

Pause Until A Task Finishes for All Lines

Assume you must wait until Shipping delivers all lines of the sales order before you send the sales order to invoicing, and then send all lines together to invoicing. So you create an orchestration process that has these steps. You use step 5 to add the pause.

Step	Task Type	Task Name
1	Schedule	Schedule
2	Reservation	Reserve
3	Shipment	Ship Goods
4	Pause	Wait for Delivery

Step	Task Type	Task Name
5	Pause	Wait for Consolidated Billing
6	Invoice	Invoice

Here's the pause rule that you add on step 5.

Note

Code	Description
`At least one`	Use this test to determine whether at least one fulfillment line in the sales order hasn't shipped. It asks the question.

Code	Description
	• Is there any order line in the sales order that's still on the Wait for Delivery task? If an order line is still doing the Wait for Delivery task, then it means there's at least one order line that Shipping hasn't delivered, and we must continue to pause before proceeding to the next orchestration process step. Wait for Delivery is an example value. You can use whatever type of pause you need while waiting for an action to happen, then move to the next the pause while you wait for billing.
`taskInstance is a DOOSeededOrchestrationRules.DOOTa`	Declare the taskInstance variable, then store attributes of the task instance that the orchestration process is currently processing into this variable. DOOTaskInstance gets all the task instances for all instances of the orchestration process that are currently running for all order lines on the sales order. For example, if three task instances are currently running for three order lines, then DOOTaskInstance gets all three task instances.
`taskInstance.taskName equals ignore case "Wait for Delivery"`	taskName identifies the value that you enter for the Task Name attribute on the orchestration process step. In this example, you entered Wait for Delivery as the Task Name on step 4. Step 4 is the pause step that determines whether any order lines are still waiting for delivery.
`taskInstance.actualCompletionDate is null`	Examine the value of the actualCompletionDate attribute in the taskInstance variable. If it doesn't contain a value, then it means Shipping hasn't delivered the item, and we must continue to pause. Fulfillment enters a value in actualCompletionDate to indicate when Shipping delivered the item.
`Header is a DOOSeededOrchestrationRules.DOOHe`	Declare the Header variable, then store attributes of the order header that the orchestration process is currently processing into this variable.
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.
`Assign Header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_EVENT`	Pause the orchestration process until an interprocess communication event happens.
`Assign Header.sacResult.eventName = DooSeededOrchestrationRules.SacRe SYSTEM_EVENT_IPC_PAUSE`	Use an interprocess communication (IPC) to pause according to an event across orchestration processes. Use it to evaluate the pause every time any task finishes for any line in the sales order.
`Assign Header.sacResult.reevaluateFlag = "Y"`	Yes, we want to reevaluate the pause.

For brevity, the screen prints in this topic don't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.

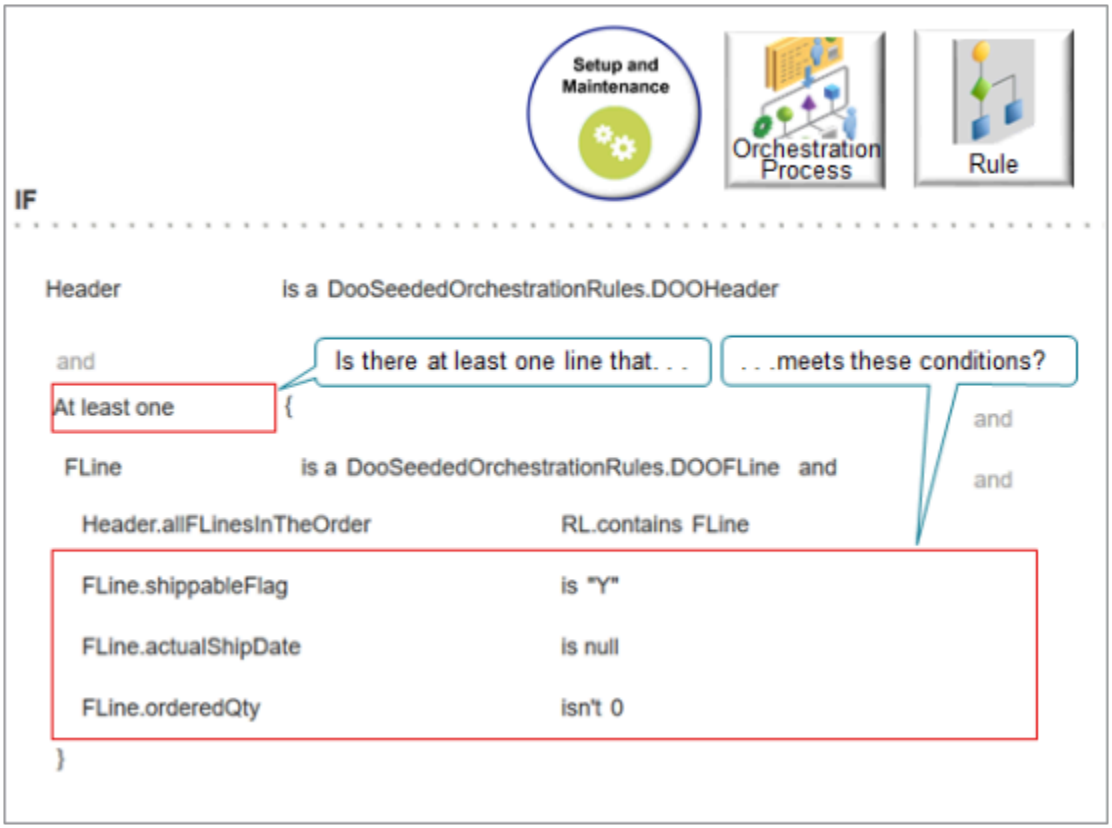
Pause Nonshippable Lines Until Shipping Ships All Shippable Lines

Here's what the rule does.

- Pause the nonshippable lines in a sales order until Shipping finishes shipping all shippable lines in the order.
- Automatically release the pause for the nonshippable lines, then proceed to the next step in the orchestration process.

  A subsequent step in the orchestration process can now send all order lines of the sales order to invoicing together.

Add your pause task after the shipping task and before the invoicing task in your orchestration process. For example, orchestration waits until Shipping ships all the sales order's shippable lines before proceeding to invoicing.



Note

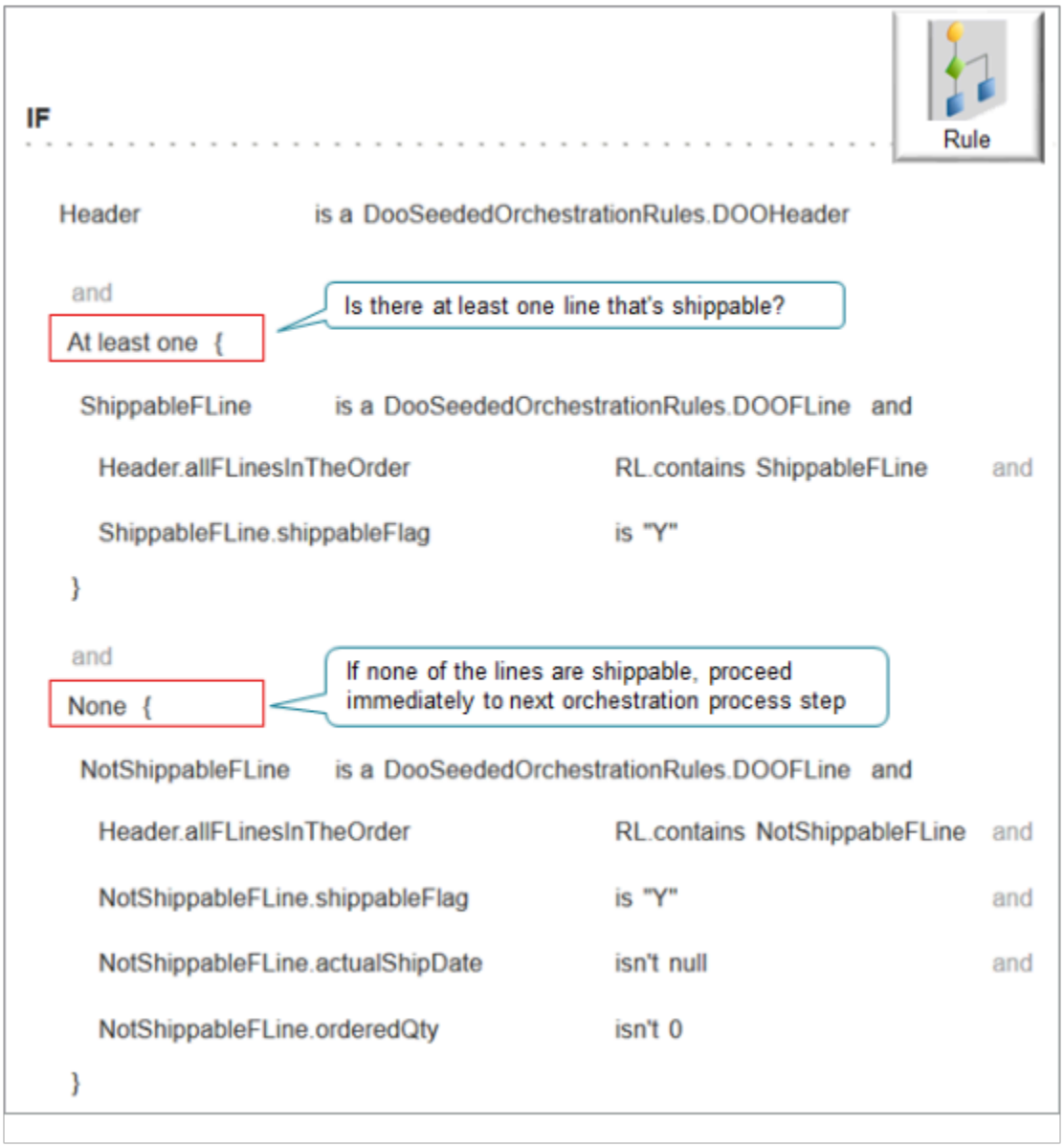
Code	Description
`At least one`	Use this test to determine whether at least one fulfillment line in the sales order hasn't shipped. It asks the question. • Is there any shippable line in the sales order that has an actual ship date that's empty? If the actual ship date is empty, it means there's at least one shippable line that hasn't shipped, and we must continue to pause before proceeding to the next orchestration process step. As an alternative, use **`There is no case where`** to pause nonshippable lines until Shipping ships at least one shippable line.
`FLine is a DOOSeededOrchestrationRules.DOOFI`	Declare the FLine variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable.
`header.allFLinesInTheOrder RL.contains FLine`	Declare allFLinesInTheOrder into the rules language (RL) dictionary, then set the value of allFLinesInTheOrder to the value that FLine contains. This condition makes sure allFLinesInTheOrder references a fulfillment line that the orchestration process is processing. It also makes sure you correctly declare the variable into the dictionary. You use it to examine all lines in the sales order, not only the line the current orchestration process instance is processing.
`FLine.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable.
`FLine.actualShipDate is null`	Skip lines that don't have an actual ship date.
`FLine.orderedQty isn't 0`	Skip lines that have a zero quantity.
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.

Code	Description
`Assign Header.sacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_EVENT`	Pause the orchestration process until an interprocess communication event happens.
`Assign Header.sacResult.eventName = DooSeededOrchestrationRules.SacRe SYSTEM_EVENT_IPC_PAUSE`	Use an interprocess communication (IPC) to pause according to an event across orchestration processes. Use it to evaluate the pause every time any task finishes for any line in the sales order.
`Assign Header.sacResult.reevaluateFlag = "Y"`	Yes, we want to reevaluate.

Pause Nonshippable Lines Until Shipping Ships At Least One Shippable Line

Here's what the rule does.

- Pause nonshippable lines until Shipping ships at least one of the shippable lines in the sales order.
- Send all nonshippable lines to invoicing as soon Shipping ships the first shippable line.
- If the sales order doesn't have any shippable lines, then proceed immediately to the next orchestration process step. There's no need to pause if no lines are shippable.


IF
Rule
Header is a DooSeededOrchestrationRules.DOOHeader
and
At least one {
Is there at least one line that's shippable?
ShippableFLine is a DooSeededOrchestrationRules.DOOFLine and
Header.allFLinesInTheOrder RL.contains ShippableFLine and
ShippableFLine.shippableFlag is "Y"
}
and
None {
If none of the lines are shippable, proceed immediately to next orchestration process step
NotShippableFLine is a DooSeededOrchestrationRules.DOOFLine and
Header.allFLinesInTheOrder RL.contains NotShippableFLine and
NotShippableFLine.shippableFlag is "Y" and
NotShippableFLine.actualShipDate isn't null and
NotShippableFLine.orderedQty isn't 0
}
THEN
assign Header.sacResult.sacType
= DooSeededOrchestrationRules.SacResult.SAC TYPE EVENT
assign Header.sacResult.eventName
= DooSeededOrchestrationRules.SacResult.SAC_SYSTEM_EVENT_IPC_PAUSE
assign Header.sacResult.reevaluateFlag = "Y"

Note

Code	Description
`At least one`	Use this test to determine whether at least one fulfillment line in the sales order is shippable.
`ShippableFLine is a DOOSeededOrchestrationRules.DOOFI`	Declare the ShippableFLine variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable.
`header.allFLinesInTheOrder RL.contains ShippableFLine`	Declare allFLinesInTheOrder into the rules language (RL) dictionary, then set the value of allFLinesInTheOrder to the value that ShippableFLine contains. This condition makes sure allFLinesInTheOrder references a fulfillment line that the orchestration process is processing. It also makes sure you correctly declare the variable into the dictionary. You use it to examine all lines in the sales order, not only the line the current orchestration process instance is processing.
`ShippableFLine.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable.
`None`	If none of the lines are shippable, proceed immediately to next orchestration process step. You create a separate variable, NotShippableFLine, for the None clause so its independent of the ShippableFLine variable from the At Lease One clause.
`NotShippableFLine is a DooSeededOrchestrationRules.DOOFI`	Declare NotShippableFLine into the rules language (RL) dictionary.
`Header.allFLinesInTheOrder RL.contains NotShippableFLine`	Make sure allFLinesInTheOrder references a fulfillment line that the orchestration process is processing. Examine all lines in the sales order, not only the line the current orchestration process instance is processing.
`NotShippableFLine.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable. This statement is in a None clause, which asks the question. • If none of the lines contain "Y" for the shippableFlag, then the test evaluates to true.
`NotShippableFLine.actualShipDate isn't null`	Skip lines that don't have an actual ship date.
`NotShippableFLine.orderedQty isn't 0`	Skip lines that have a zero quantity.

Send Nonshippable Lines to Invoicing Without Waiting for Shipping to Ship Shippable Lines

Assume you create a pause task that pauses all order lines until shipping has shipped all lines. You submit a sales order, then notice these fulfillment lines.

Item	Status
AS54888 Desktop Computer	Paused for Invoice
Freight Charge for the AS54888	Paused for Invoice

But you want to send the freight charge to invoicing without waiting for shipping to finish shipping the AS54888. Here's how you can do that.

Add a pause task on the orchestration step that happens after a step that does the shipping task and before the step that does an invoicing task.

where

Code	Description
`FLine.inventoryItemId.longValue() is 149`	The value 149 in the InventoryItemId attribute on the fulfillment line uniquely identifies the AS54888 Desktop Computer.
`header.childFLines RL.contains FLine`	Declare the childFLines attribute into the rules language (RL) dictionary, then set the value of childFLines to the value that FLine contains.
`FLine.extendedAmount same or less than 3000`	-
`Assign new DooSeededOrchestrationRules.SacRe SAC = new DooSeededOrchestrationRules.SacRe`	Create a new variable named SAC and set it to the SacResult type in the DooSeededOrchestrationRules method.
`Assign Header.sacResult = SAC`	Set the value of the sacResult object on the order header to the value that SAC contains.
`assert new DooSeededOrchestrationRules.SacRe ( eventName:"NO FOB" reevaluateFlag = "Y", sacType:DooSeededOrchestrationRul TYPE_EVENT)`	-

Send Nonshippable Lines to Invoicing When There Aren't Any Shippable Lines

Pause sending the nonshippable lines to invoicing until shipping ships at least one shippable line, then send the nonshippable lines and the shippable lines to invoicing together, but with a condition.

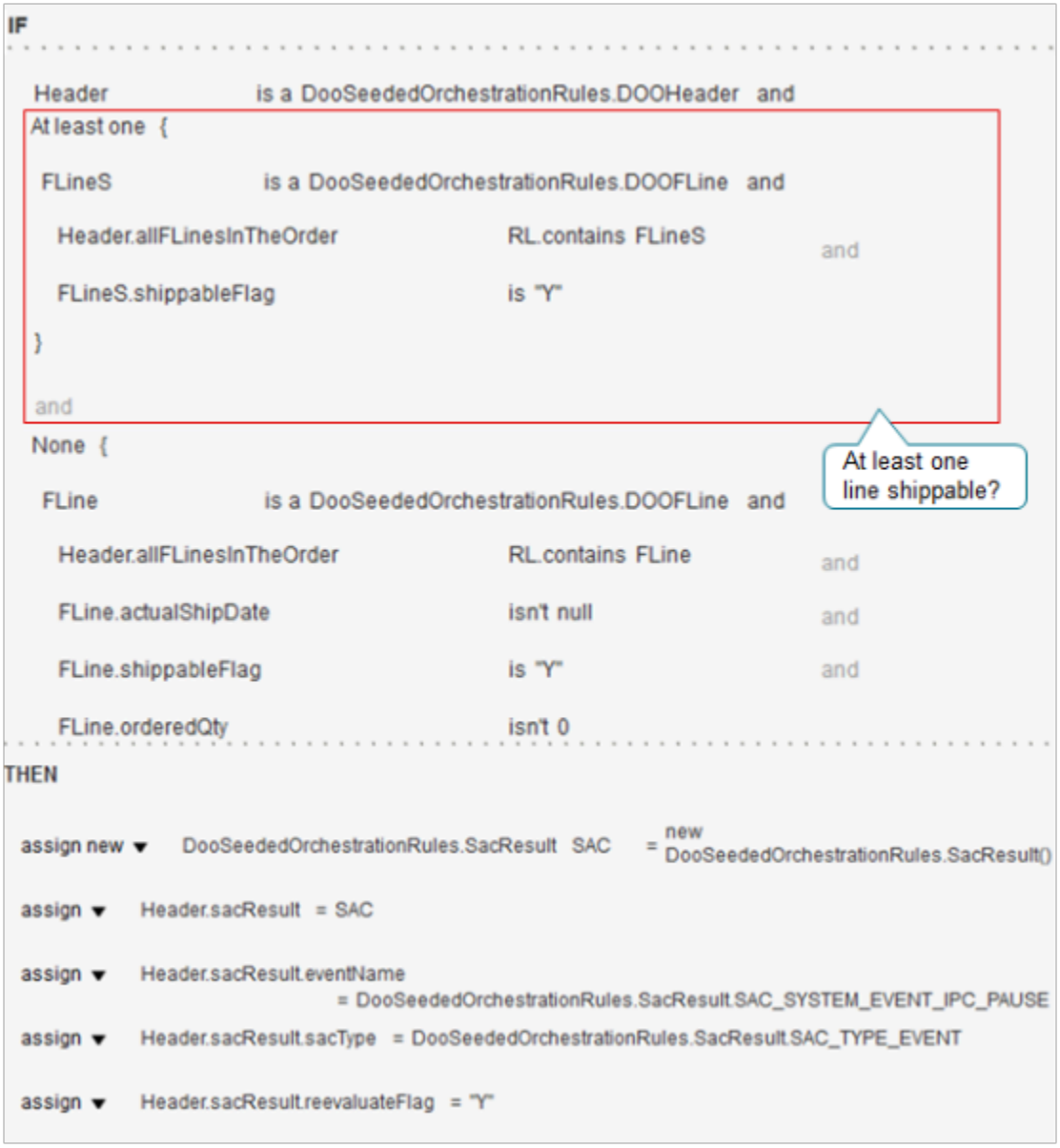
```
if the sales order doesn't have any shippable lines, then send the nonshippable lines to invoicing without waiting.
```

To account for the condition, you add the pause task on an orchestration step that happens after a step that does a shipping task and before a step that does an invoicing task.

Here's what the rule does.

- Pause nonshippable lines until Inventory Management receives at least one shippable line in the sales orders.
- Automatically release the paused, nonshippable lines, then proceed to the next orchestration process step.

  The orchestration process can now send the nonshippable lines and the shippable lines on the order to invoicing together.



Note

Code	Description
`At least one`	Use this test to determine whether at least one fulfillment line in the sales order is shippable.
`FLineS is a DOOSeededOrchestrationRules.DOOFL`	Declare the FLineS variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable. You can use whatever name you like for the variable. We're using FLineS to distinguish it from the FLine variable that use in other examples in this topic.
`header.allFLinesInTheOrder RL.contains FLineS`	Declare allFLinesInTheOrder into the rules language (RL) dictionary, then set the value of FLineS to the value that FLineS contains.

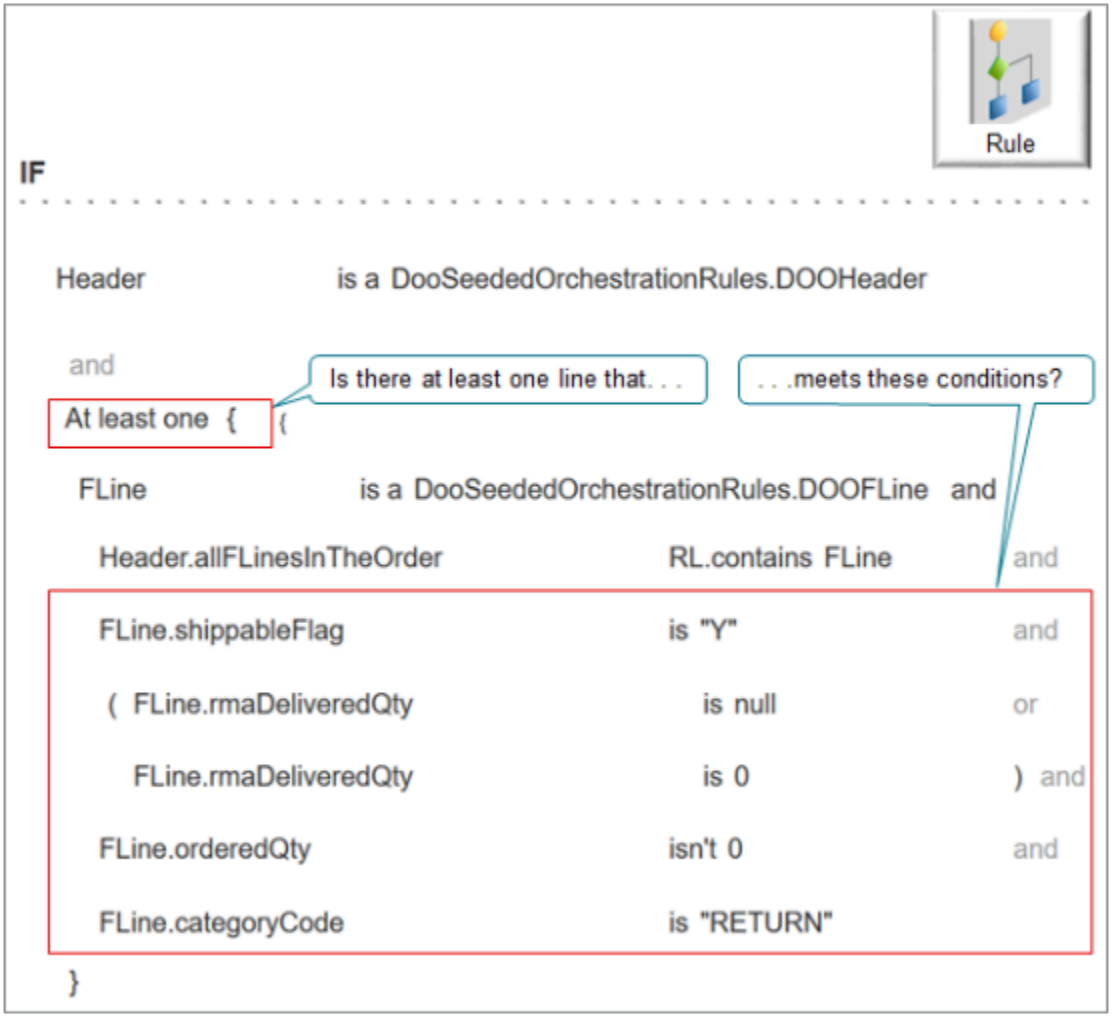
Code	Description
	This condition makes sure allFLinesInTheOrder references a fulfillment line that the orchestration process is processing. It also makes sure you correctly declare the variable into the dictionary. You use it to examine all lines in the sales order, not only the line the current orchestration process instance is processing.
`FLineS.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable.
`None`	If none of the lines are shippable, proceed immediately to the next orchestration process step. You create a separate variable, FLine, for the None clause so its independent of the FLineS variable from the At Lease One clause.
`FLine is a DooSeededOrchestrationRules.DOOFI`	Declare FLine into the rules language (RL) dictionary.
`Header.allFLinesInTheOrder RL.contains FLine`	Make sure allFLinesInTheOrder references a fulfillment line that the orchestration process is processing. Examine all lines in the sales order, not only the line the current orchestration process instance is processing.
`FLine.actualShipDate isn't null`	Skip lines that don't have an actual ship date.
`FLine.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable. This statement is in a None clause, which asks the question. • If none of the lines contain "Y" for the shippableFlag, then the test evaluates to true.
`FLine.orderedQty isn't 0`	Skip lines that have a zero quantity.

Send Nonshippable Lines and Return Lines to Invoicing

Here's what the rule does.

- Pause nonshippable lines until Inventory Management receives all shippable lines in the sales orders.
- Wait for Inventory Management to receive all shippable lines.
- Automatically release the paused, nonshippable lines, then proceed to the next orchestration process step.

  The orchestration process can now send the nonshippable lines and the return lines on the order to invoicing together.



Note

Code	Description
`At least one`	Use this test to determine whether at least one fulfillment line in the sales order is shippable and whether fulfillment has successfully returned the line.

Code	Description
`FLine is a DOOSeededOrchestrationRules.DOOFI`	Declare the FLine variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable.
`header.allFLinesInTheOrder RL.contains ShippableFLine`	Declare allFLinesInTheOrder into the rules language (RL) dictionary, then set the value of allFLinesInTheOrder to the value that FLine contains.
`FLine.shippableFlag is "Y"`	We are only looking at fulfillment lines that are shippable.

`FLine.rmaDeliveredQty is null`
 	
`FLine.rmaDeliveredQty is 0`
	rmaDeliveredQty is the quantity on the return line that fulfillment has delivered to inventory. If it has no value or is zero, then it means fulfillment hasn't successfully returned the line into inventory.

`FLine.orderedQty isn't 0`	Fulfillment sets orderedQty to zero when it cancels the line. This condition makes sure we only consider lines that aren't canceled.
`FLine.CategoryCode is "RETURN"`	Only consider return lines. Skip lines that aren't return lines.

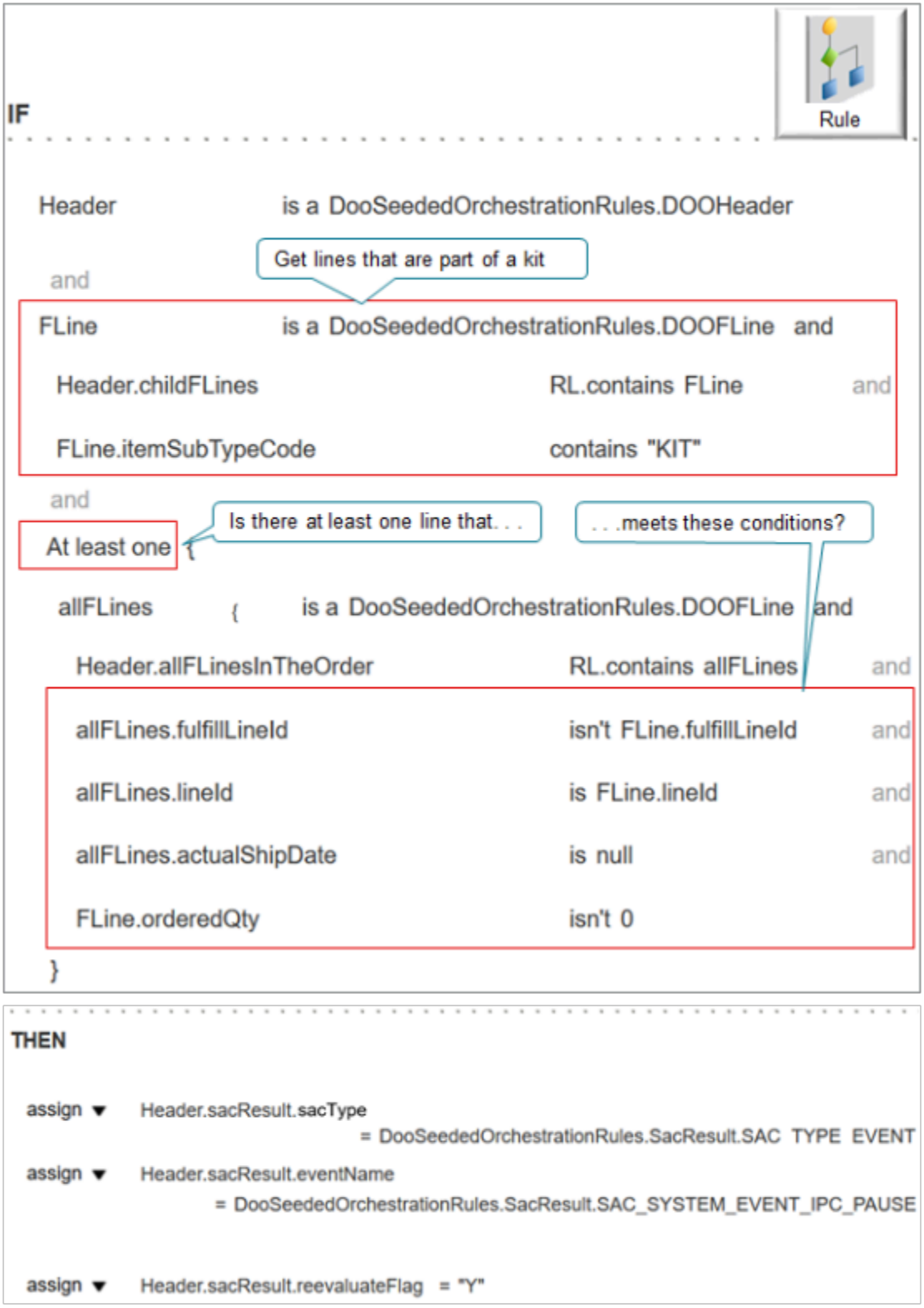
Pause Shipped Lines of a Kit Until Shipping Ships All of the Lines

Assume you order a kit that has a quantity of more than one, such as 10. All of the lines aren't available in inventory, so fulfillment splits some lines and ships part of the kit while it waits for the factory to replenish inventory. You don't want to send the shipped lines to invoicing by themselves. You need to wait until fulfillment ships all of the quantity for the kit.

Here's what the rule does.

- Pause the shipped lines until fulfillment successfully fulfills all lines for the kit.
- Automatically release the paused lines, then proceed to the next orchestration process step.

  The orchestration process can now send all lines for the kit to invoicing together.


Rule
IF
Header is a DooSeededOrchestrationRules.DOOHeader
and
Get lines that are part of a kit
FLine is a DooSeededOrchestrationRules.DOOFLine and
Header.childFLines RL.contains FLine and
FLine.itemSubTypeCode contains "KIT"
and
At least one {
Is there at least one line that. . .
. . .meets these conditions?
allFLines { is a DooSeededOrchestrationRules.DOOFLine and
Header.allFLinesInTheOrder RL.contains allFLines and
allFLines.fulfillLineId isn't FLine.fulfillLineId and
allFLines.lineId is FLine.lineId and
allFLines.actualShipDate is null and
FLine.orderedQty isn't 0
}
THEN
assign Header.sacResult.sacType
= DooSeededOrchestrationRules.SacResult.SAC TYPE EVENT
assign Header.sacResult.eventName
= DooSeededOrchestrationRules.SacResult.SAC_SYSTEM_EVENT_IPC_PAUSE
assign Header.sacResult.reevaluateFlag = "Y"

Note

Code	Description
`FLine is a DOOSeededOrchestrationRules.DOOFI`	Declare the FLine variable, then store attributes of the fulfillment lines that the orchestration process is currently processing into this variable.
`header.childFLines RL.contains FLine`	Declare the childFLines attribute into the rules language (RL) dictionary, then set the value of childFLines to the value that FLine contains.
`FLine.itemSubTypeCode contains "KIT"`	Only get fulfillment lines that are part of a kit. Skip all other lines.
`At least one`	Use this test to determine whether fulfillment split at least one fulfillment line in the kit.
`allFLines is a DOOSeededOrchestrationRules.DOOFI`	Declare the allFLines variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable. We will cycle through all the lines.
`header.allFLinesInTheOrder RL.contains allFLines`	Declare allFLinesInTheOrder into the rules language (RL) dictionary, then set the value of allFLinesInTheOrder to the value that allFLines contains.

`allFLines.fulfillLineId isn't FLine.fulfillLineId`
 	
`allFLines.lineId is FLine.lineId`
	If fulfillment ships part of a kit, then it splits the lines in the kit, ships some lines and doesn't ship others. Let's see whether fulfillment split the lines in the kit.
	
	Notice that.
	
	• The allFLines variable contains the fulfillLineId values for all fulfillment lines in the sales order.
	• The FLine variable contains only the fulfillLineId for the line that we are currently examining.
	
	If the fulfillLineId attribute in the allFLines attribute doesn't equal the fulfillLineId attribute in the FLine variable, and if the lineId attribute in the allFLines attribute does equal the lineId attribute in the FLine variable, then we can see that fulfillment didn't split the line that we are currently examining.
	
	How do we know? The fulfillLineId identifies the fulfillment line that contains the kit. Fulfillment must maintain the relationship between the kit's fulfillment line and new lines that it creates when it splits a kit. So fulfillment assigns the lineId of each new line that it creates to the value of the fulfillLineId.

`FLine.actualShipDate is null`	Only consider lines that fulfillment hasn't shipped.
`FLine.orderedQty isn't 0`	Fulfillment sets orderedQty to zero when it cancels the line. This condition makes sure we only consider lines that aren't canceled.

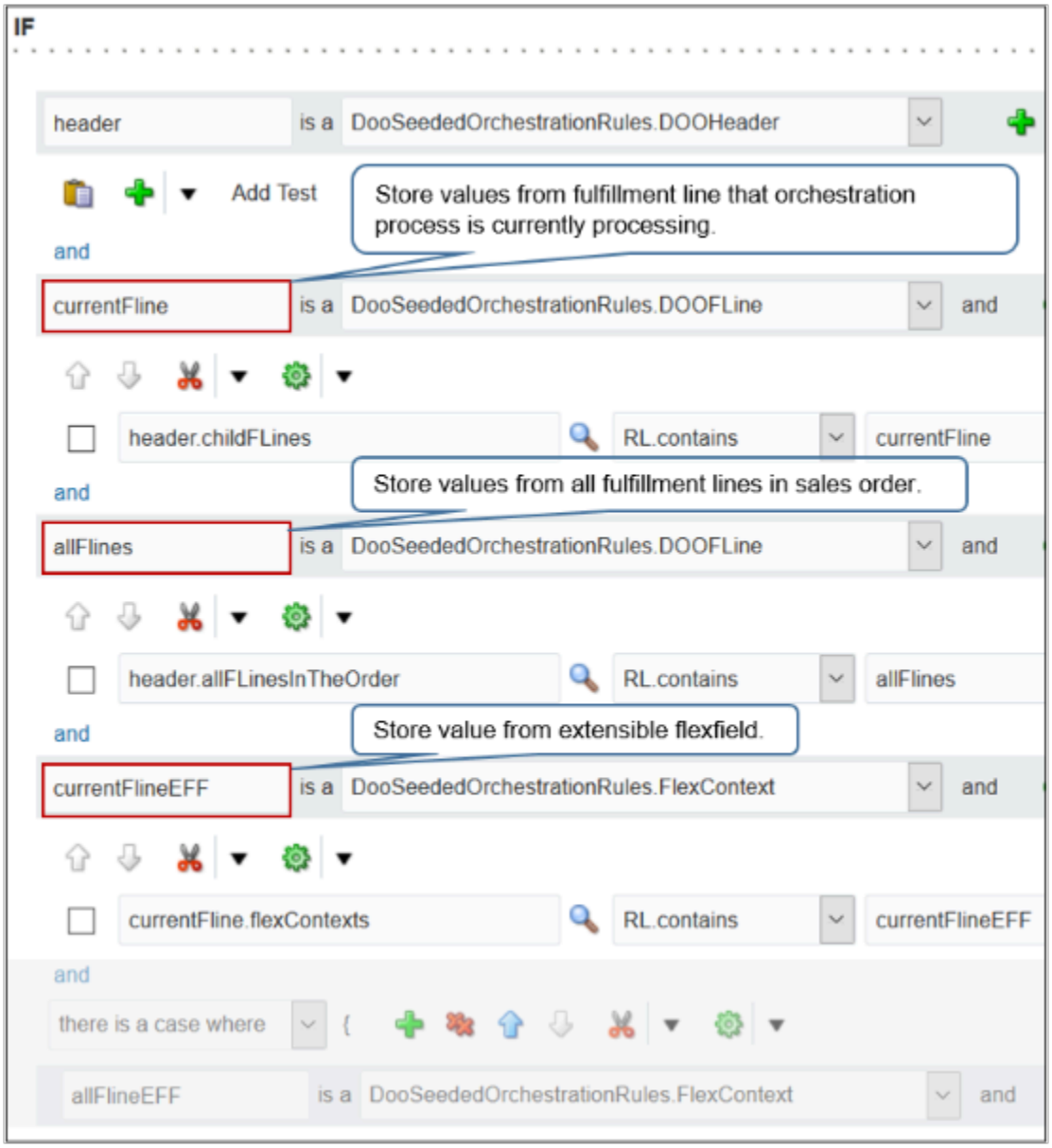
Send Shippable and Nonshippable Lines to Invoicing in Groups

Assume you must modify the pause task so the orchestration process sends all fulfillment lines to invoicing as a single group of lines, including all the lines of a sales order that fulfillment can ship and can't ship instead of only lines that fulfillment can ship. Here's what the rule does.

- Automatically releases the pause for the nonshippable lines that are part of a group as soon as Shipping finishes shipping the shippable lines.
- Uses an extensible flexfield to allow the Order Entry Specialist to specify which lines are part of the group.

You can use the same Then statement that you use in the Pause Nonshippable Lines Until Shipping Ships Shippable Lines example, but the If statement is different. Add the pause task so it happens between the shipping task and invoicing task in your orchestration process.

Here's the first part of the If statement.

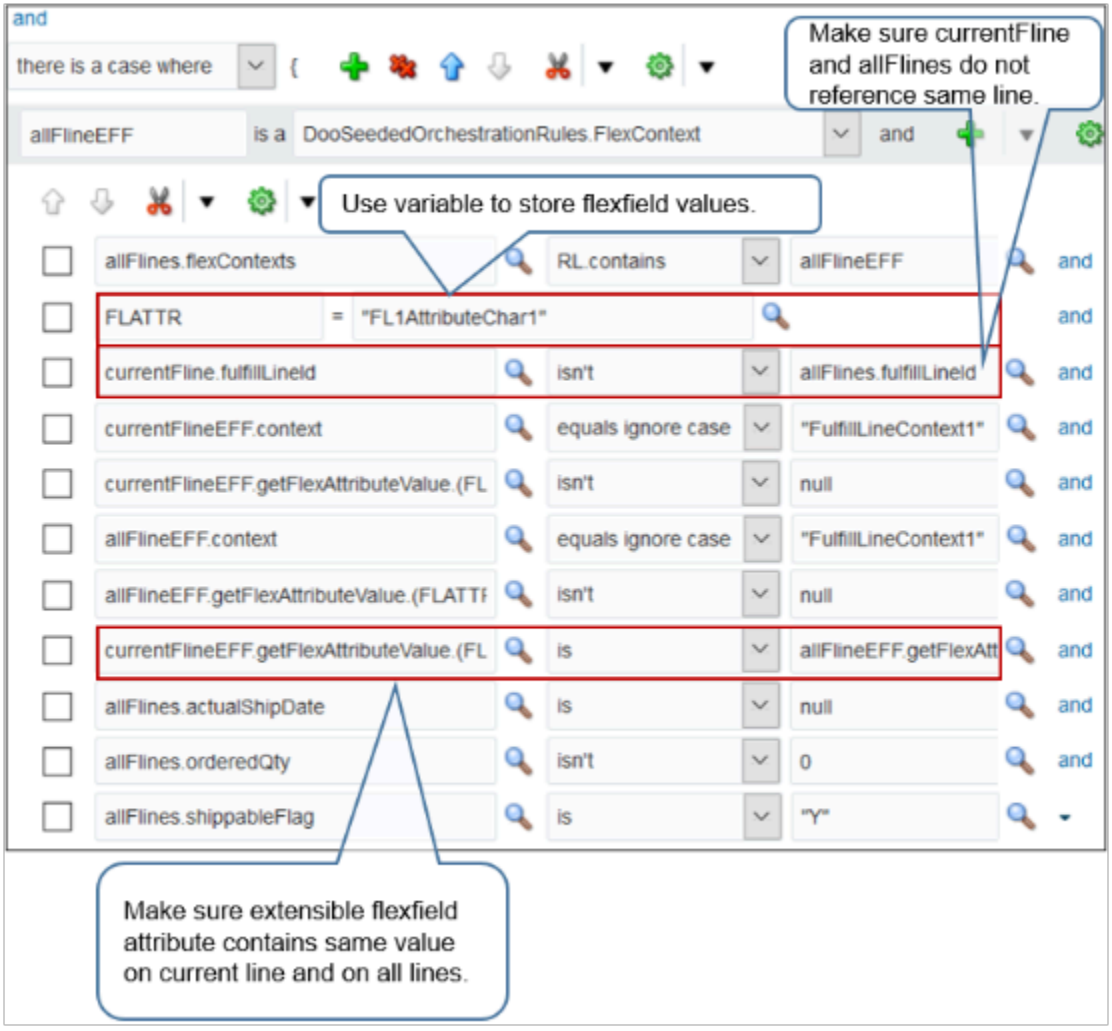


Note

Code	Description
`currentFline is a DOOSeededOrchestrationRules.DOOFI`	Declare the currentFline variable, then store attributes of the fulfillment line that the orchestration process is currently processing into this variable.

Code	Description
`header.childFLines RL.contains currentFline`	Declare currentFline into the rules language (RL) dictionary, then set the value of currentFline to the value that childFLines contains. This condition makes sure currentFline references a fulfillment line that the orchestration process is processing. It also makes sure you correctly declare the variable into the dictionary.
`allFlines is a DOOSeededOrchestrationRules.DOOFI`	Declare the allFlines variable, and set it to type DOOFLine.
`header.allFLinesInTheOrder RL.contains allFlines`	Declare the allFlines variable into the rules language dictionary, and set the value of allFlines to the value that allFLinesInTheOrder contains. This condition makes sure you correctly declare the variable into the dictionary.
`currentFlineEFF is a DOOSeededOrchestrationRules.FlexC`	Declare the currentFlineEFF variable, and set it to type FlexContext.
`currentFline.flexContexts RL.contains currentFlineEFF`	Declare currentFlineEFF into the rules language dictionary, and set the value of currentFlineEFF to the value that flexContexts contains. This condition makes sure you correctly declare the variable into the dictionary. flexContexts is a list of context rows for the extensible flexfield. FlexContext is a type of row or variable.

Here's the second part of the If statement.



Note that the FulfillLineContext1 extensible flexfield context and the FL1AttributeChar1 segment are one group.

Assume the sales order contains 10 fulfillment lines and you must group them into two sets of five lines each. You can use.

- FulfillLineContext1 and FL1AttributeChar1 to set up GROUP1 so it contains the first five lines
- FulfillLineContext2 and FL1AttributeChar2 to set up GROUP2 so it contains the last five lines

Here's the code from second part of the If statement.

Code	Description
`allFlineEFF is a DOOSeededOrchestrationRules.FlexC`	Declare the allFlineEFF variable, and store the value of object FlexContext into this variable. Method DooSeededOrchestrationRules contains FlexContext.
`FLATTR = "FL1AttributeChar1"`	Declare variable FLATTR (flexfield attribute), and set it to FL1AttributeChar1. You reference the FL1AttributeChar1 extensible flexfield attribute in more than one location in this rule. For efficiency, and to avoid problems because of typographical errors, you can declare FLATTR one time, set it to FL1AttributeChar1, and then reference FLATTR instead of declaring FL1AttributeChar1 more than one time.
`currentFline.fulfillLineId isn't allFlines.fulfillLineId`	Proceed to the next AND only if the fulfillLineId attribute in the currentFline variable and the fulfillLineId attribute in the allFlines variable doesn't reference the same fulfillment line.
`currentFlineEFF.context equals ignore case "FulfillLineContext1"`	Get the value of the flexfield context from currentFlineEFF. Proceed to the next AND statement only if the context contains the FulfillLineContext1 string. `equals ignore case` specifies to ignore the case sensitivity of the FulfillLineContext1 string. This condition avoids the situation where your deployment might use some other case for this attribute.
`currentFlineEFF.getFlexAttributeV (FLATTR) isn't null`	Use the getFlexAttributeValue method to get the value of the FL1AttributeChar1 attribute from the extensible flexfield on the current fulfillment line. Proceed to the next AND only if FL1AttributeChar1 contains a value. Recall that FLATTR contains FL1AttributeChar1.
`currentFlineEFF.getFlexAttributeV (FLATTR) isn't null`	Use the getFlexAttributeValue method to get the value of the FL1AttributeChar1 attribute from the extensible flexfield on the current fulfillment line. Proceed to the next AND only if FL1AttributeChar1 contains a value. Recall that FLATTR contains FL1AttributeChar1.
`allFlineEFF.context equals ignore case "FulfillLineContext1"`	Get the value of the flexfield context from allFlineEFF. Proceed to the next AND statement only if the context contains the FulfillLineContext1 string.
`allFlineEFF.getFlexAttributeValue (FLATTR) isn't null`	Get the value of the FL1AttributeChar1 attribute from allFlineEFF. Proceed to the next AND only if FL1AttributeChar1 contains a value.
`currentFlineEFF.getFlexAttributeV (FLATTR) is allFlineEFF.getFlexAttributeValue (FLATTR)`	Proceed to the next AND only if the extensible flexfield attribute contains the same value on the current line and on all lines. This condition makes sure these fulfillment lines are part of the same group.

Related Topics

- Use Case to Pause for an Event
- Pause Orchestration Processes Until Time Elapses
- Manage Order Management Parameters
- Manage Order Attributes That Identify Change
- Use Business Rules in Orchestration Processes

Use Case to Pause for an Event

Assume you work for a publisher who will release a new book at some point in the future, and must provide your Gold customers an opportunity to order the book before your company releases it to the public. You can set up an

You will create these rules:

- If the customer is Gold, then pause the orchestration process.
- If the customer isn't Gold, then skip the pause task.

Summary of the Set Up

1. Create the orchestration process.
2. Add orchestration process steps.
3. Create If statement for first rule.
4. Create Then statement for first rule.
5. Create second rule.
6. Deploy the orchestration process. For details, see *Deploy Orchestration Processes*.

This topic uses example values. You might need different values, depending on your business requirements.

For brevity, the screen prints in this topic don't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.

Get more details.

- Learn about the properties you set in this topic for the pause task. For details, see *Overview of Pausing Orchestration Processes*.
- Learn more about these business rules. For details, see *Overview of Using Business Rules With Order Management*.

Create the Orchestration Process

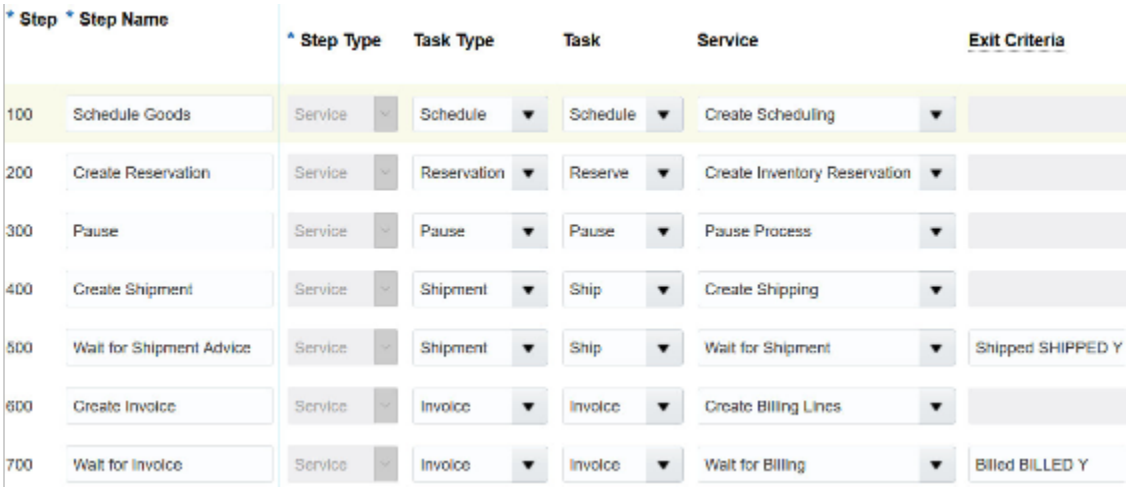
1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, click **Actions > Create**.
3. On the Create Orchestration Process Definition page, set the values, then click **Save**.

Attribute	Value
Process Name	Pause_for_Gold

Attribute	Value
Process Display Name	Pause_for_Gold
Process Class	Ship Order Class
Set	Common Set

Add Orchestration Process Steps

You will add these steps.



* Step	* Step Name	* Step Type	Task Type	Task	Service	Exit Criteria
100	Schedule Goods	Service	Schedule	Schedule	Create Scheduling	
200	Create Reservation	Service	Reservation	Reserve	Create Inventory Reservation	
300	Pause	Service	Pause	Pause	Pause Process	
400	Create Shipment	Service	Shipment	Ship	Create Shipping	
500	Wait for Shipment Advice	Service	Shipment	Ship	Wait for Shipment	Shipped SHIPPED Y
600	Create Invoice	Service	Invoice	Invoice	Create Billing Lines	
700	Wait for Invoice	Service	Invoice	Invoice	Wait for Billing	Billed BILLED Y

Try it.

1. On the Step Definition tab, click **Actions > Add Row**.
2. Set the values, then click **Save**.

Attribute	Value
Step Name	Schedule Goods
Step Type	Service
Task	Schedule
Service	Create Scheduling

3. Add more steps. Repeat steps 1 and 2. Use these values.
    - Create Reservation
    - Create Shipment
    - Wait for Shipment Advice
    - Create Invoice
    - Wait for Invoice

    Note
    - Set the Step Type attribute for all steps to Service.
    - Set other attributes so they reflect step behavior.

      For example, for the Create Reservation step, set these values.

Attribute	Value
Step Type	Service
Task	Reserve
Service	Create Inventory Reservation

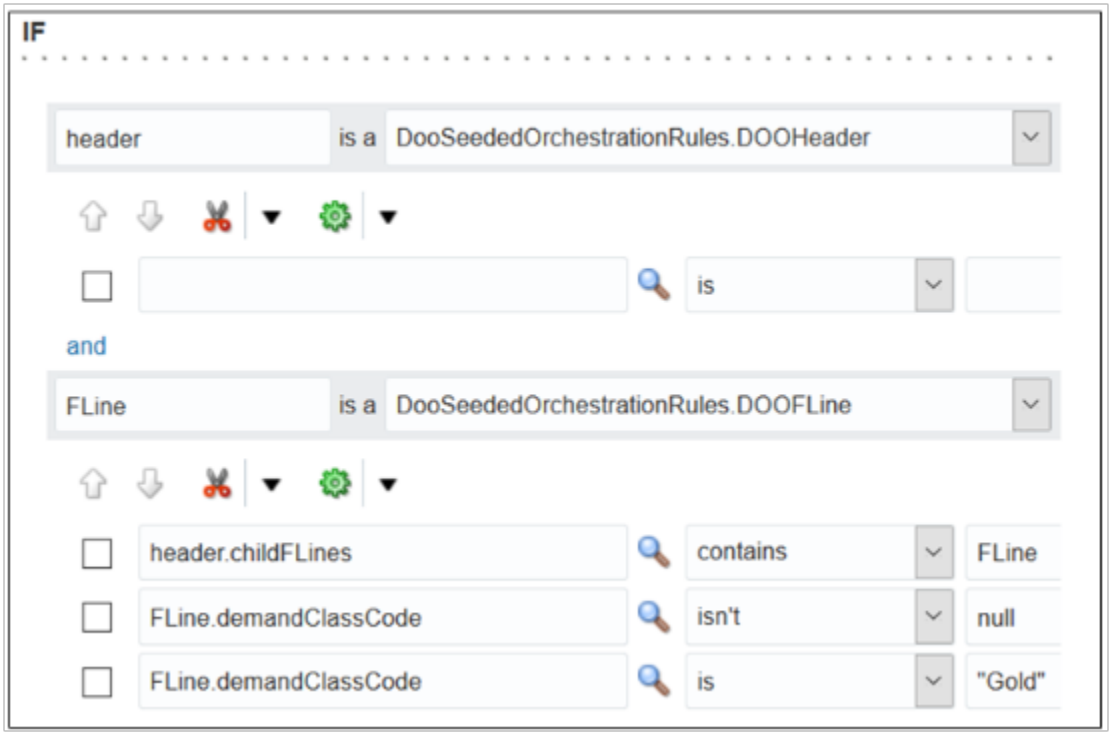
    - Make sure you set the Exit Criteria attribute for each wait step. For example, set Exit Criteria for the Wait for Shipment Advice step to Shipped.
4. Add the pause step. Click the Create Reservation step, click **Add Row**, set the values in the new row, then click **Save**.

Attribute	Value
Step Name	Pause
Step Type	Service
Task	Pause
Service	Pause Process

Create If Statement for First Rule

You will create the start-after condition. This condition determines whether the customer is Gold.

You will create this If statement.



where

- DOOFLine is a function in the DooSeededOrchestrationRules dictionary.
- DOO is an abbreviation for distributed order orchestration, which is an earlier name for order orchestration.
- FLine is an abbreviation for fulfillment line.
- demandClassCode is a fulfillment line attribute. This attribute stores the customer class, such as Gold, Silver, or Bronze.
- `isn't` determines whether demandClassCode contains Gold.
- For details, see *Guidelines for Pausing Orchestration Processes*.

Create the If statement for the first rule.

1. Create the rule.
    - On the Pause step, in the Pause Rule column, click **Click for Rule**.
    - In the Start After Condition dialog, click **Add Rule**, then replace `Rule 1` with `Pause for Gold Customers`.
    - Click **Properties**, then add check mark to Advanced Mode.
2. Specify the dictionary and facts to examine.
    - In the IF area, in the window to the left of Is A, enter `header`.
    - Set the window to the right of Is A to DooSeededOrchestrationRules.DOOHeader.
    - Click **Add Pattern**.
    - In the window to the left of Is A, enter `FLine`.
    - Set the window to the right of Is A to DooSeededOrchestrationRules.DOOFLine.

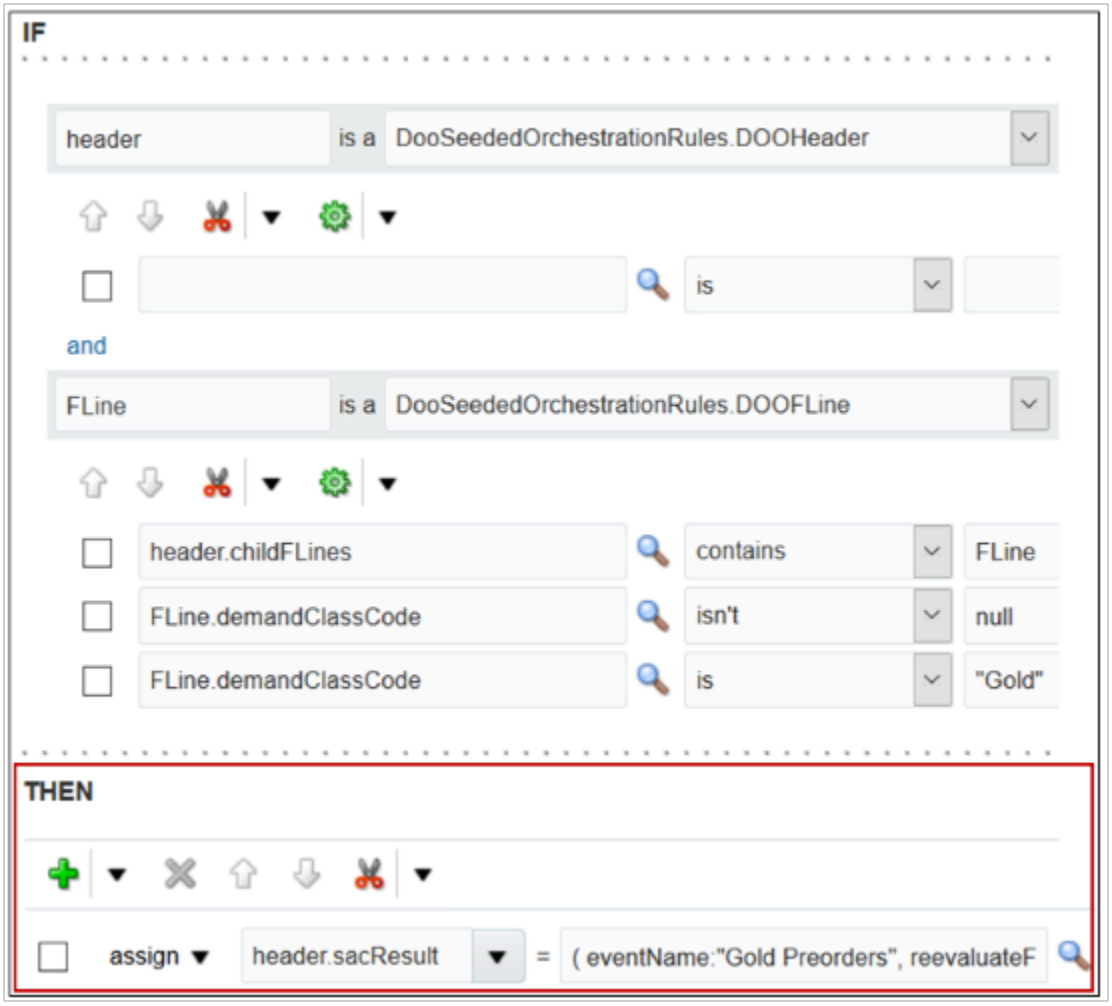
3. Make sure to evaluate only fulfillment lines.
    - Under FLine, click **Add Test**.
    - Click **Left Value**.
    - In the Condition Browser, expand **Header**, click **ChildFLines**, then click **OK**.
    - Change **Is** to **Contains**.
    - Click **Right Value**.
    - In the Condition Browser, click **FLine**, then click **OK**.
4. Make sure the attribute you're about to test isn't empty.
    - On the test you just added, to the far right, click the **down arrow**, then click **Simple Test**.
    - On the new line that displays, click **Left Value**.
    - In the Condition Browser, expand **FLine**.

      Notice that each value under FLine references a fulfillment line attribute.
    - Click **demandClassCode**, then click **OK**
    - Change **Is** to **Isn't**.
    - Click **Right Value**.
    - In the Condition Browser, click **null**, then click **OK**.
5. Determine whether the customer is Gold.
    - On the test you just added, to the far right, click the **down arrow**, then click **Simple Test**.
    - On the new line that displays, click **Left Value**.
    - In the Condition Browser, expand **FLine**.
    - Click **demandClassCode**, then click **OK**.
    - In the Right Value window, enter `"Gold"`.

      You must include the double quotation marks (" ").

Create Then Statement for First Rule

You will create this Then statement.



For brevity, this screen print doesn't include the entire value for the event.

Create the Then statement for the first rule.

1. In the Then area, click **Add Action > Assign**.
2. Click **Select a Target**, then click **header.SacResult**.
3. In the window to the right of the equal sign (=), enter this value.

```
( eventName:"Gold Preorders",
 reevaluateFlag:"N",sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_EVENT )
```

   where

   - eventName is a variable that you can use to help track the status of the pause task. The Order Management work area uses the text you enter wherever it displays status details, such as in the Gantt

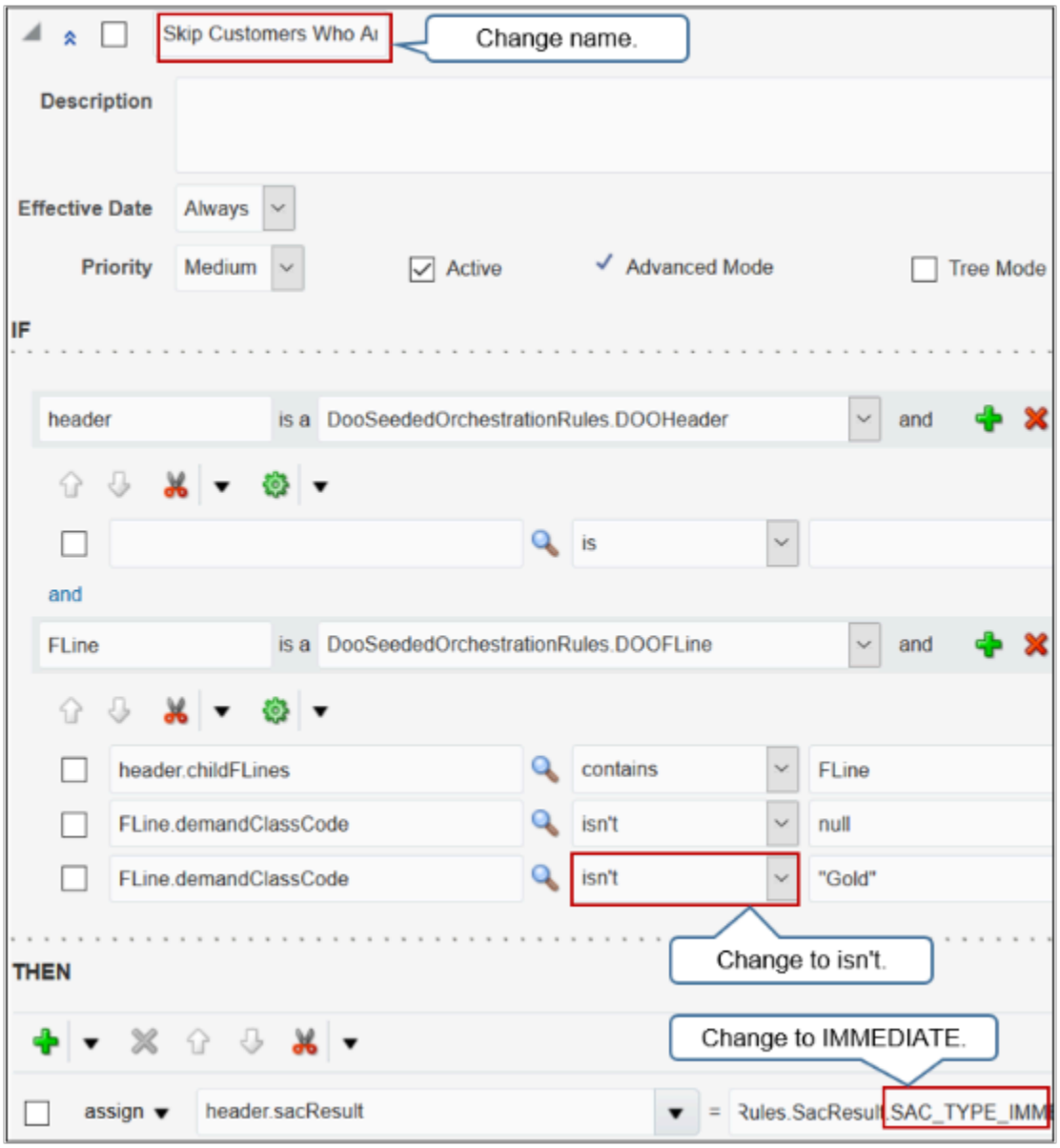
chart on tab Orchestration Plan of page Orchestration Process that the user can access from page Manage Orchestration Process. This example specifies to use event Gold Preorders.

- `reevaluateFlag:"N"` specifies not to evaluate this rule again.
- `sac` is an abbreviation for **start after condition**.
- `sacType` specifies the dictionary, rule set, and event to use for the start after condition.
- `DooSeededOrchestrationRules` is a dictionary that contains predefined rules you can use in your business rule.
- is a rule set in DooSeededOrchestrationRules. It contains events and variables you can use to specify how to handle the result of a start after condition.
- `EVENT` is an event in SacResult. It specifies to use a business event.
- A pause type of EVENT causes the orchestration process to pause until an event occurs.
- An attribute on an orchestration process references this event, but you can also use it with a web service. You can also use other parameters to release a pause task. This event isn't related to the events that the event framework in Oracle Middleware uses.
- You must include double quotation marks (" ").

4. Click **Save**.

Create the Second Rule

You will create this rule.



Try it.

1. On the Pause step, in the Pause Rule column, click **Click for Rule**.
2. In the Pause Rule Set dialog, in the check box to the left of Pause for Gold Customers, add a check mark.
3. Click **Cut > Paste**.
4. Scroll down to the bottom of the dialog and notice the rule that the paste added.
5. Make these changes to this new rule.

Replace This Value	With This Value
Pause for Gold Customers	Skip Customers Who Are Not Gold
FLine.demandClassCode is "Gold"	FLine.demandClassCode isn't "Gold"

Replace This Value	With This Value
header.sacResult = (eventName:"Gold Preorders", reevaluateFlag:"N", sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_EVENT)	header.sacResult = (eventName:"Gold Preorders", (reevaluateFlag:"N", sacType:DooSeededOrchestrationRules.SacResult.SAC_TYPE_IMMEDIATE) Start after condition IMMEDIATE ends the pause task and allows the orchestration process to continue to the next step.

6. Click **Save**.
7. On the Edit Orchestration Process Definition page, click **Save**.

Related Topics

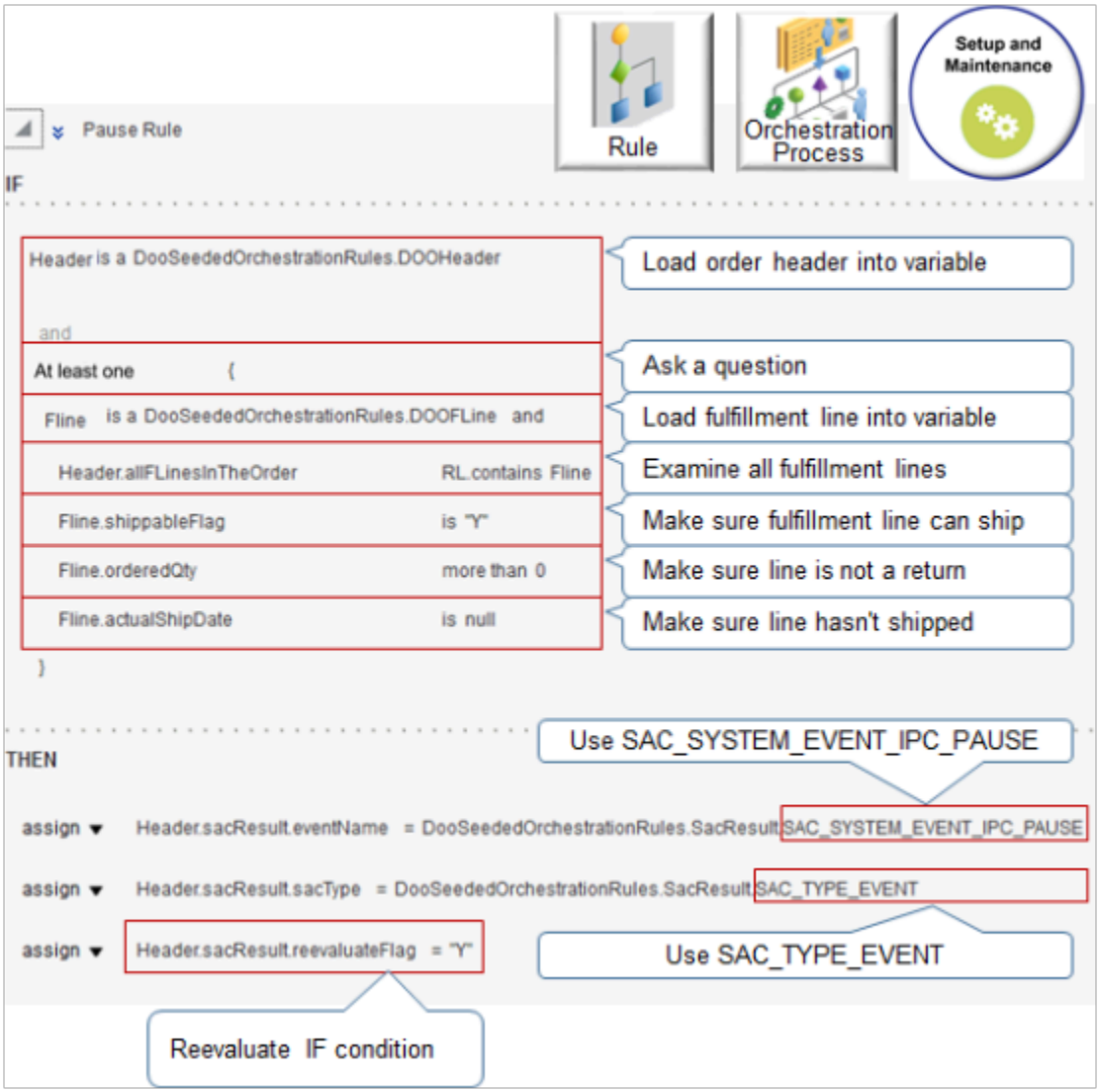
- Overview of Pausing Orchestration Processes
- Guidelines for Pausing Orchestration Processes

Pause for Dependencies

Pause Orchestration Processes Until Dependencies Resolve

Set up a pause task to temporarily stop an orchestration process from running until a dependency resolves.

The dependency in this example is to wait for all lines to ship before continuing to the invoicing step.



Note

- The Pause step references the pause task. The Pause step resides between the Shipping step and the Invoicing step in the orchestration process.
- If no fulfillment line meets any condition in the IF statement, then all fulfillment lines have shipped, the rule releases the pause, and the flow continues to the next orchestration process step.
- For brevity, this screen print doesn't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.

Here's your code.

Code	Description
`Header is a DooSeededOrchestrationRules.DOOHe`	Declare the Header variable, then load values into Header from object DOOHeader of method DooSeededOrchestrationRules. DOOHeader contains order header attributes and their values.

Code	Description
`At least one`	Ask a question. Does the sales order include at least one fulfillment line that order fulfillment hasn't shipped, and that order fulfillment can ship?
`Fline is a DooSeededOrchestrationRules.DOOFl and`	Declare variable Fline (fulfillment line), then load values into Fline from object DOOFline of method DooSeededOrchestrationRules. DOOFline (Distributed Order Orchestration Fulfillment Line) contains fulfillment line attributes and their values.
`Header.allFLinesinTheOrder`	Examine all fulfillment lines in the sales order, including lines that the orchestration process isn't currently processing. A dependency might exist between the fulfillment line that the orchestration process is currently processing, and other fulfillment lines in the sales order, so you must inspect all of them.
`Fline.shippableFlag is "Y"`	If the shippableFlag attribute on the fulfillment line is Y, then proceed. Make sure each fulfillment line is shippable. Some items can't ship, such as a subscription. If a fulfillment line contains a subscription, then shippableFlag is N for the line, the rule ignores the line, and moves to the next line.
`Fline.actualShipDate is null`	If the actualShipDate attribute on the fulfillment line doesn't contain a value, then it indicates the fulfillment line hasn't shipped, and the rule must pause the orchestration process.
`Fline.orderedQty more than 0`	If the orderedQty attribute on the fulfillment line is 0, then it indicates the fulfillment line is a canceled line, the rule ignores the line, and moves to the next line.
`SAC_SYSTEM_EVENT_IPC_PAUSE`	Use SAC_SYSTEM_EVENT_IPC_PAUSE, where IPC means Interprocess Communication. The rule evaluates this type of pause each time a task finishes for each fulfillment line.
`reevaluateFlag = Y`	Evaluate the condition each time an orchestration process step finishes. In this example, the Shipping step ships the fulfillment lines, so the rule must evaluate after the Shipping step finishes for each fulfillment line to determine whether order fulfillment shipped all fulfillment lines.

Related Topics

- Use Case to Pause for an Event
- Pause Orchestration Processes Until Time Elapses
- Manage Order Management Parameters
- Manage Order Attributes That Identify Change
- Use Business Rules in Orchestration Processes

Use Case to Pause for a Dependency

This use case coordinates orchestration processes to make sure Order Management invoices all the fulfillment lines that a sales order contains at the same time, even if the shipment dates vary.

The orchestration process uses the same sequence of steps to process each fulfillment line, but these lines might not be synchronized.

Assume a school district places a sales order for 600 new history books and requires that they receive the invoice for the sales order only after Order Management ships all books.

Assume that the orchestration process must get the books from different warehouses.

- Fulfillment line 1 gets 80 books from the Seattle warehouse
- Fulfillment line 2 gets 400 books from the Denver warehouse
- Fulfillment line 3 gets 120 books from the Chicago warehouse

You set up an orchestration process that processes the fulfillment lines, pauses until fulfillment shipped all lines, resumes processing, then sends the lines to billing.

Summary of the Steps

1. Set up the orchestration process.
2. Create the first rule.
3. Create the If statement for the second rule.
4. Create the Then statement for the second rule.

For brevity, the screen prints in this topic don't assign the result from the rules dictionary and to the header, but you must assign that result. For details, see the Assign the Result subtopic in *Guidelines for Pausing Orchestration Processes*.

This topic uses example values. You might need different values, depending on your business requirements.

For background details about:

- Properties that you set for the pause task, see *Overview of Pausing Orchestration Processes*.
- Business rules, see *Overview of Using Business Rules With Order Management*.

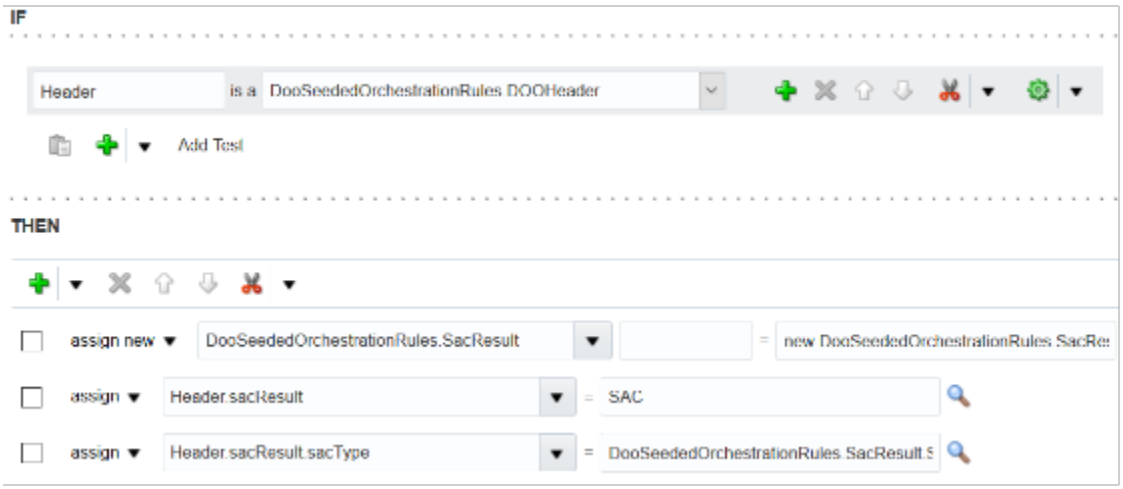
Set Up the Orchestration Process

View an example that describes how to do this set up. For details, see *Pause Orchestration Processes Until Events Happen*.

1. Create a new orchestration process and set up the header.
2. Add orchestration process steps.
    - Schedule
    - Reserve
    - Ship
    - Wait for Shipment
    - Pause
    - Invoice
    - Wait for Billing

Create the First Rule

You will create a business rule.



Note

Code	Description
`DooSeededOrchestrationRules`	A dictionary that contains predefined rules you can use in your business rule.
`DOOHeader`	An object in the DooSeededOrchestrationRules dictionary that stores details about the sales order header.
`Assign New`	Assign DOOHeader as a new object in the DooSeededOrchestrationRules dictionary. This assignment makes the object available throughout your rule.
`SacResult`	A rule set in the DooSeededOrchestrationRules dictionary. It contains events and variables you can use to specify how to handle the result of a start after condition.
`SAC`	A value you can set for SacResult. The term `sac` is an abbreviation for **start after condition**.
`Assign Header.sacResult = SAC`	Set the value of SacResult to SAC.
`sacType`	A property of SacResult. sacType stores the value that the rule uses to determine whether to pause the orchestration process or release it.
`SAC_TYPE_IMMEDIATE`	A value of sacType. If sacType contains IMMEDIATE, then the rule will immediately release the pause task.
`Header.SacResult.sacType = DooSeededOrchestrationRules.SacRe TYPE_IMMEDIATE`	Set sacType to SAC_TYPE_IMMEDIATE.

Create the first rule.

1. On the Pause step, in the Pause Rule column, click **Add Rule**.

   In the Start After Condition dialog, click **Add Rule > Properties**, then set the values.

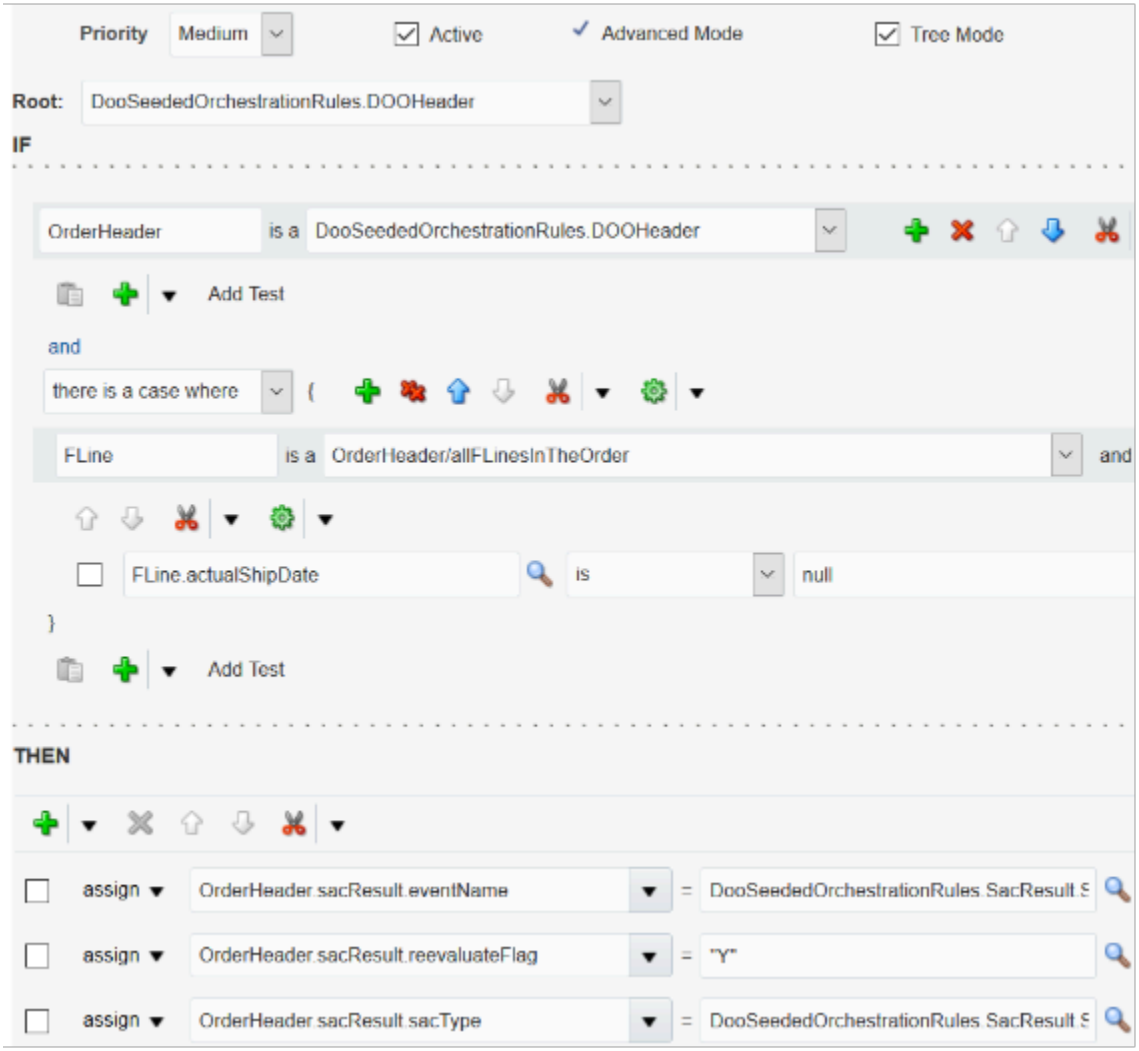
Attribute	Value
Name	Default Rule for Dependent Pause
Description	Default rule that pauses orchestration process until a dependency resolves.
Priority	Highest
Advanced Mode	Contains a check mark.

2. In the field below If, enter an alias, such as `Header`.
3. In the field to the right of Is A, select **DooSeededOrchestrationRules.DOOHeader**.
4. Click the **down arrow** next to Right Value, then click **Delete Test**.

   If you remove the test, then the rule always applies the actions that the Then area contains.
5. In the Then area, click **Add Action**, then click **Assign New**.
6. Click **Select a Target**, then click **DooSeededOrchestrationRules.SacResult**.
7. In the field to the left of the equal sign (=), enter `sac`.
8. In the field to the right of the equal sign (=), click **Expression Value**.
9. In the Condition Browser dialog, expand **DooSeededOrchestrationRules**, expand **SacResult**, click **New**, then click **OK**.
10. Click **Add Action > Assign**.
11. Click **Select a Target**, then click **Header.sacResult**.
12. In the field to the right of the equal sign (=), enter `sac`.
13. Click **Add Action > Assign**.
14. Click **Select a Target**, then click **Header.sacResult.sacType**.
15. In the field to the right of the equal sign (=), click **Expression Value**.
16. In the Condition Browser dialog, expand **DooSeededOrchestrationRules**, expand **SacResult**, click **SAC_TYPE_IMMEDIATE**, and then click **OK**.
17. Click **Validate**, then click **Save**.

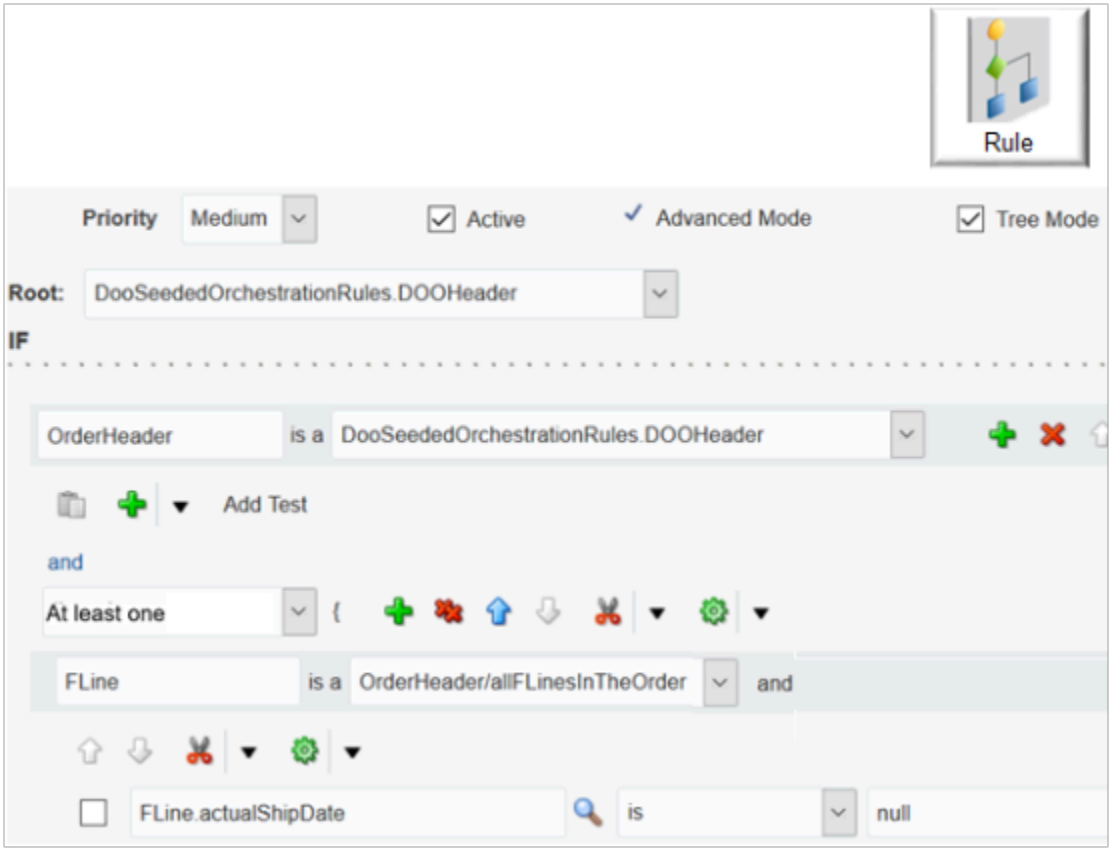
Create the If Statement for the Second Rule

You will create this rule.



It states that if the actual ship date isn't null on all fulfillment lines, then release the pause task.

First, you create the If statement.



where

Code	Description
`At least one`	Pause all shipped lines even if the fulfillment system shipped all lines except one line.
`FLine is a OrderHeader/ allFLinesIntheOrder`	Coordinate across more than one orchestration process when more than one orchestration process processes the fulfillment lines for the sales order.
`FLine.actualShipDate is null`	Determine whether any of the fulfillment lines haven't shipped. actualShipDate is null when orchestration starts. Orchestration sets actualShipDate to a value when it ships the item.

Create the If statement for the second rule.

1. On the Pause step, in the Pause Rule column, click **Add Rule**.

   In the Start After Condition dialog, collapse the rule you just created.

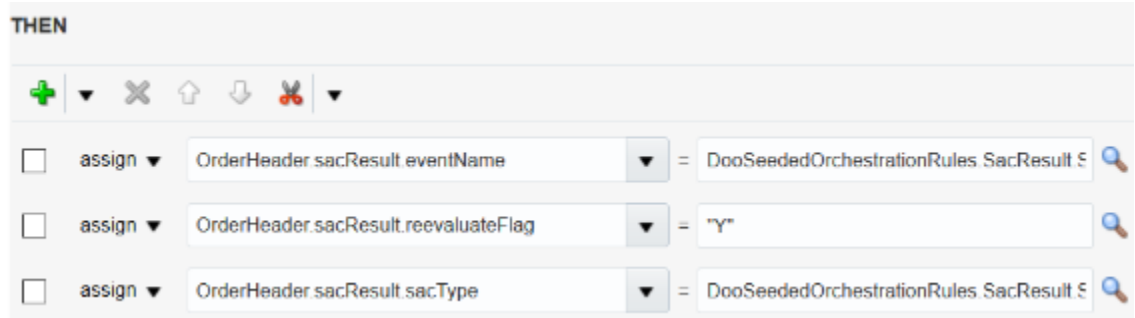
2. Click **Add Rule > Properties**, then set the values.

Attribute	Value
Name	Rule for Pause Dependency
Description	Rule that pauses orchestration process until a dependency resolves.
Priority	Medium
Advanced Mode	Contains a check mark.
Tree Mode	Contains a check mark.

3. Set Root to **DooSeededOrchestrationRules.DOOHeader**.
4. In the If area, in the field to the left of Is A, replace the default value with an alias, such as `orderHeader`.
5. Click the **down arrow** to the right of Right Value, and then click **Delete Test**.
6. Click **Add Pattern** to the right of the If statement.
7. To the right of the empty fields, click **Surround Pattern with Parenthesis**, then click **Surround**.
8. In the list of values under And, click `At least one`.
9. In the field on the left, under `At least one`, enter `FLine`.
10. In the field to the right of Is A, select **OrderHeader/allFLinesIntheOrder**.
11. Click the **down arrow** next to Add Test, then click **Simple Test**.
12. Click **Left Value**.
13. In the Condition Browser, expand **FLine**, click **actualShipDate**, then click **OK**.
14. Click **Right Value**.
15. In the Condition Browser, click **null**, then click **OK**.

Create the Then Statement for the Second Rule

You will create a Then statement.



For details about these properties, see *Overview of Pausing Orchestration Processes*.

Create the Then Statement for the second rule.

1. In the Then area, click the **down arrow** next to Add Action, then click **Assign**.
2. Click **Select a Target**, then click **OrderHeader.sacResult.eventName**.

3. Click **Expression Value**.
4. In the Condition Browser, expand **DooSeededOrchestrationRules**, expand **SacResult**, select **SAC_SYSTEM_EVENT_IPC_PAUSE**, then click **OK**.

   This step sets up the rule to evaluate fulfillment lines until the flow ships all of them, and then to release them. `Interprocess` communication (`IPC`) implements a pause according to an event.
5. Click the **down arrow** next to Add Action, then click **Assign**.
6. Click **Select a Target**, then click **OrderHeader.sacResult.reevaluateFlag**.
7. In the field to the right of the equal sign (=), enter `"Y"`.

   You must include the double quotation marks.
8. Click the **down arrow** next to Add Action, then click **Assign**.
9. Click **Select a Target**, then click **OrderHeader.sacResult.sacType**.
10. Click **Expression Value** on the same line.
11. In the Condition Browser, expand **DooSeededOrchestrationRules**, expand **SacResult**, click **SAC_TYPE_EVENT**, then click **OK**.
12. Click **Validate**, then click **Save**.

Related Topics

- Use Case to Pause for an Event
- Set Up Orchestration Processes
- Overview of Pausing Orchestration Processes

Hold

Guidelines for Setting Up Holds on Sales Orders

Use these guidelines to help set up a sales order hold.

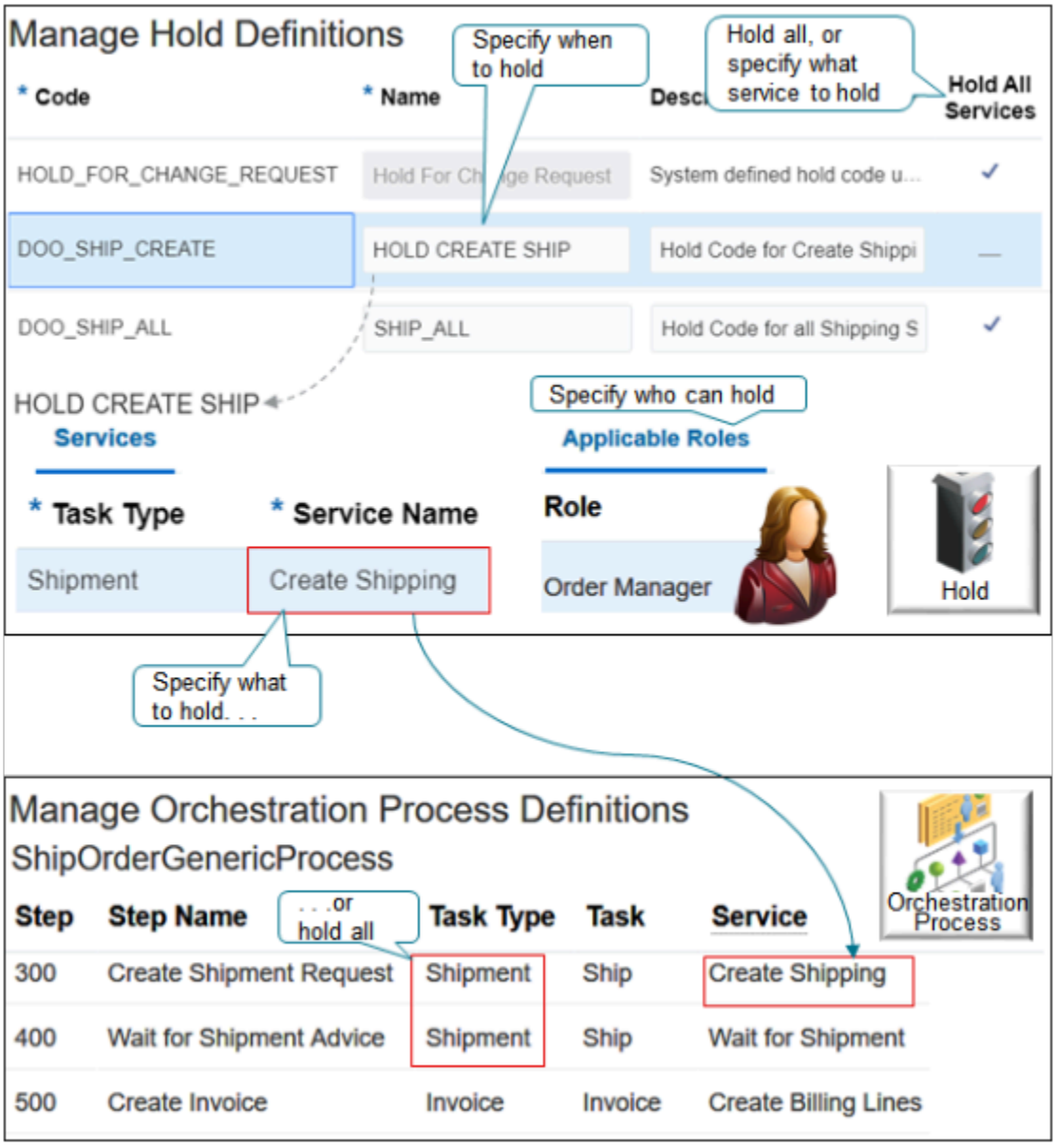
Determine the Type of Hold You Need

You can apply different kinds of holds.

Type	Description
Generic	Hold that you apply on any step of an orchestration process. It doesn't apply a hold for a specific task. You enable Hold All Services when you create the hold code.
Service	Hold that you apply on a specific task type and service, such as the Create Billing Lines service of the Invoice task type. You specify the service when you create the hold code.
System	Hold that Order Management automatically applies and releases. Here are some examples. • **Hold_for_Change_Request**. Order Management applies the Hold_for_Change_Request hold when the user revises a sales order so no other user or process can modify the sales order during the revision.

Type	Description
	• **DOO_CreditCheck**. Order Management applies this hold when the customer exceeds their credit limit. For example, if credit check fails because the customer exceeded their credit limit, then you don't want the orchestration process to proceed to the ship step and ship the item.

Create Your Hold Code



Note

- Use the Setup and Maintenance work area to manage hold codes, orchestration processes, and task types.
- Use the hold code to specify the task to hold, such as creating the invoice.
- Specify the service, or specify the task type.
- Set the attributes.

Attribute	Description
Applicable Roles	Use the Applicable Roles tab to specify who can apply and release the hold, such as the Order Manager. Note that you can't specify a role for applying a system hold, but you can specify a role to release a DOO_CreditCheck hold.
Set	As an option, use the Set attribute of the hold code to specify the business unit where Order Management applies the hold. You can also use Set to create a set of hold codes and apply them all to the same business unit.
Hold Tasks In Progress	Order Management automatically enables the Hold Tasks In Progress option when the user applies it on the Order page in the Order Management work area or when you import a hold. Disable this option only if you haven't set up your fulfillment system to correctly interpret it.
End Date	Don't set the End Date attribute for a predefined, system hold code, such as HOLD_FOR_CHANGE_REQUEST. Order Management uses these codes. If you set the end date on one, you might introduce errors in your implementation. Examine the description to determine whether a predefined hold code is a system hold code. The description usually identifies system holds. For example, here's the description for HOLD_FOR_CHANGE_REQUEST. `System defined hold code used in change management process` If you set the End Date for a hold code, then you can't apply the hold after the end date happens, but you can release the hold after the end date happens.

Hold All Services

If You Don't Enable the Hold All Services Option	If You Enable the Hold All Services Option
The **service** that you specify affects each orchestration process step that references the service. Assume you: • Don't add a check mark to Hold All Services for the HOLD CREATE SHIP hold. • On the services tab, you specify the Create Billing Lines service. Your hold affects only step 300 of orchestration process ShipOrderGenericProcess because step 300 is the only step that references the Create Shipping service.	The **task type** that you specify affects each orchestration process step that references the task type. Assume you: • Add a check mark to Hold All Services for the HOLD CREATE SHIP hold. • You leave the Services tab empty. Your hold affects step 300 and step 400 because these steps reference the Shipment task type. Use this option when you don't any part of the task to run. Let's say you're shipping a highly flammable, dangerous item. Before shipping, you place a hold to verify the carrier is licensed to handle hazardous material. Enable the Hold All Services option on the shipping task type to make sure no part of shipping runs.

If You Don't Enable the Hold All Services Option	If You Enable the Hold All Services Option
Use this option when you want some of the services of the task type to run but not others. For example, you want to put a hold on Create Shipping until you verify the orchestration process successfully created the shipment request, but you want the process to automatically do the Wait for Shipment task after you remove the hold for Create Shipping.	

See Where You Can Release Holds

If I Apply Hold Through	Can I Release it on Create Order or Revise Order Page in Order Management Work Area	Can I Release It in Fulfillment View in Order Management Work Area	Can I Release It Through Web Service	Can I Release It Through REST API
Create Order or Revise Order Page in Order Management Work Area	Yes	Yes	Yes, but only if you applied the hold on the order header or order line. You can't use the web service to release a hold that you apply on a fulfillment line on the Revise Order page.	Yes, but only if you applied the hold on the order header or fulfillment line.
Fulfillment View	Yes, but only if you applied the hold on the order header or fulfillment line. You can't use the Revise Page to release a hold that you applied on an order line because you already submitted the order, and Order Management transformed the order line to a fulfillment line.	Yes	Yes, but only if you applied the hold on the order header. You can't use the web service to release a hold that you apply on a fulfillment line in a fulfillment view.	Yes, but only if you applied the hold on the order header or fulfillment line. You can't use REST API to release a hold that you apply on a order line in a fulfillment view.
Web Service	Yes, but only if you applied the hold on the order header. You can't use the Create page or Revise page to release a hold that you apply on an order line or fulfillment line through the web service.	Yes	Yes	Yes, but only if you applied the hold on the order header.
REST API	Yes, but only if you applied the hold on the order header or fulfillment line.	Yes, but only if you applied the hold on the fulfillment line.	Yes, but only if you applied the hold on the order header.	Yes

If I Apply Hold Through	Can I Release it on Create Order or Revise Order Page in Order Management Work Area	Can I Release It in Fulfillment View in Order Management Work Area	Can I Release It Through Web Service	Can I Release It Through REST API
			You can't use the web service to release a hold that you applied on a fulfillment line through REST API.	

Use a Wait Step

- If you enable the FOM_SKIP_HOLDS_WAIT_VALIDATION profile option, then Order Management will process a request to apply or release a hold on the fulfillment line only if the line is on a wait step. For example, when the line is in the Awaiting Shipping status. If the line isn't on a wait step, then Order Management will return an error. To enable the profile option, go to the Setup and Maintenance work area, click **Tasks > Search**, search for and open the Manage Profile Options page, search the Profile Option Code attribute for FOM_SKIP_HOLDS_WAIT_VALIDATION, then make sure the Enabled option contains a check mark at the Site level.

  An orchestration process that references a service that does a Wait operation is a wait step. To determine which services do a wait operation, go to the Manage Task Types page in the Setup and Maintenance work area, click the row that includes your task type of interest, such as Shipment, then, in the Details area, examine the services. Look for services that do a Wait operation.
- If you create a hold code for a long-running task, such as a shipping or invoicing task, then set it up so Order Management applies it on a wait step.

  Using this approach will help to avoid a concurrency problem that happens when the user revises a sales order while a long-running task is running. Placing the hold on a wait step helps to make sure any actions that were in process finish before the user can revise the order.
- Apply the hold on the Wait for Shipment service for a shipment task, and the Wait for Billing service for an invoice task. Each of these services do a wait operation.

Apply Holds on Shipment Sets

If you apply or release a hold on any source line that's part of a shipment set, then Order Management applies or releases the hold on all lines that are part of the set.

If you apply a hold on a line that's part of a shipment set, then Order Management queries the database to determine whether other lines in the same shipment set exist on the sales order. If it doesn't find any other lines in the set, then it applies the hold only on the line that you specify in the payload. The same behavior applies when you release a hold.

Assume shipment set n includes lines x, y, and z. If your import applies a hold on line x, then Order Management applies the same hold on lines y and z. Order Management applies the hold on lines y and z even if you don't include them in your payloads.

Make Sure You Have the Privileges

A job role has privileges. Examine the job role and make sure you have its privileges.

What I Need to Do	Job Role	Formal Name of Job Role
Set up a hold code.	Order Administrator	ORA_DOO_ORDER_ADMINISTRATOR_JOB
Apply, release, or view a hold on a sales order, order line, or fulfillment line on the Create Order page, Revise Order page, or in a fulfillment view.	Order Entry Specialist or Order Manager	ORA_DOO_ORDER_ENTRY_SPECIALIST_JOB ORA_DOO_ORDER_MANAGER_JOB
View a hold on an orchestration process in a fulfillment view.	Order Manager	ORA_DOO_ORDER_MANAGER_JOB
Apply or release a hold through a web service.	Create your own role.	Add privileges to your role. • DOO_MANAGE_ORDER_ORCHESTRATION_DECOMPOSITION_WEB_SERVICE_PRIV • DOO_MANAGE_ORDER_ORCHESTRATION_ORDER_MODIFICATION_PRIV
Apply a hold through REST API.	Create your own role.	Add privileges to your role. • FOM_APPLY_UPDATE_HOLDS_PRIV • DOO_MANAGE_ORCHESTRATION_ORDER_MODIFICATION_PRIV • FOM_SALES_ORDER_REST_POST_PRIV
Release a hold through REST API.	Create your own role.	Add privileges to your role. • FOM_RELEASE_HOLDS_PRIV • DOO_MANAGE_ORCHESTRATION_ORDER_MODIFICATION_PRIV • FOM_SALES_ORDER_REST_POST_PRIV

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Import Holds Through Web Service

Use a web service to import holds.

Here's an example payload that applies the DOO_SHIP_ALL hold on order line 101 for source order 20156 that you import from source system LEG.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope>
 <soapenv:Header/>
 <soapenv:Body>
 <dood:RequestHoldProcessRequest>
 <!-1 or more repetitions:-->
 <dood:ApplyHoldRequestParams>
 <dood:SourceOrderSystem>LEG</dood:SourceOrderSystem>
 <!--Optional:-->
 <dood:SourceOrderId>20156</dood:SourceOrderId>
 <!--Optional:-->
```

```
 <dood:SourceLineId>101</dood:SourceLineId>
 <!--Optional:-->
 <dood:SourceHoldCode>DOO_SHIP_ALL</dood:SourceHoldCode>
 <!--Optional:-->
 <dood:HoldComments>Hold Shipping</dood:HoldComments>
 <!--Optional:-->
 <dood:SourceOrderNumber></dood:SourceOrderNumber>
 <!--Optional:-->
 <dood:SourceOrderLineNumber></dood:SourceOrderLineNumber>
 <!--Optional:-->
 <dood:HoldName></dood:HoldName>
 <!--Optional:-->
 <dood:AppliedBy>?</dood:AppliedBy>
 </dood:ApplyHoldRequestParams>
 </dood:RequestHoldProcessRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Here's an example payload that releases the hold.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope>
 <soapenv:Header/>
 <soapenv:Body>
 <dood:ReleaseHoldProcessRequest>
 <!-1 or more repetitions:-->
 <dood:ReleaseHoldRequestParams>
 <dood:SourceOrderSystem>LEG</dood:SourceOrderSystem>
 <!--Optional:-->
 <dood:SourceOrderId>20156</dood:SourceOrderId>
 <!--Optional:-->
 <dood:SourceLineId>101</dood:SourceLineId>
 <!--Optional:-->
 <dood:SourceHoldCode>DOO_SHIP_ALL</dood:SourceHoldCode>
 <!--Optional:-->
 <dood:HoldReleaseReasonCode>DOO_RELEASE_SHIP</dood:HoldReleaseReasonCode>
 <!--Optional:-->
 <dood:HoldComments>Remove Hold</dood:HoldComments>
 <!--Optional:-->
 <dood:SourceOrderNumber>?</dood:SourceOrderNumber>
 <!--Optional:-->
 <dood:SourceOrderLineNumber>?</dood:SourceOrderLineNumber>
 <!--Optional:-->
 <dood:HoldName>?</dood:HoldName>
 <!--Optional:-->
 <dood:HoldReleaseReasonName>?</dood:HoldReleaseReasonName>
 <!--Optional:-->
 <dood:ReleasedBy>?</dood:ReleasedBy>
 </dood:ReleaseHoldRequestParams>
 </dood:ReleaseHoldProcessRequest>
 </soapenv:Body>
</soapenv:Envelope>
```

Note

- Use the ReceiveOrderRequestService web service. For details, see *Web Services That You Can Use to Integrate Order Management*.
- To apply a hold on the order header, include a value for SourceOrderId but not for SourceLineId. The hold will apply to the header and all lines in the order.
- To apply or release a hold on the order line, you must include a value for SourceOrderId and for SourceLineId.

- Apply or release a hold on more than one sales order. Use the same `soapenv:Header` and `soapenv:Body`, and include a separate RequestHoldProcessRequest section or ReleaseHoldProcessRequest section for each sales order. For example:

```
<soapenv:header> ..... </soapenv:header>
<soapenv:body>
<RequestHoldProcessRequest>
 <SalesOrderNumber>1234</SalesOrderNumber>
 <SalesOrderSystem>GPR</SalesOrderSystem>
</RequestHoldProcessRequest>
<RequestHoldProcessRequest>
 <SalesOrderNumber>2345</SalesOrderNumber>
 <SalesOrderSystem>GPR</SalesOrderSystem>
</RequestHoldProcessRequest>
</soapenv:body>
```

- The web service imports the order with the Hold Tasks In Progress option as enabled when applying or releasing a hold.
- You can use the web service to apply or release a hold on a sales order that's in Draft status.
- You can use the web service to apply or release a hold on the order header or order line, but you can't use it to apply or release a hold on a fulfillment line. To apply or release a hold on a fulfillment line, go to a fulfillment view in the Order Management work area, then use the Apply Hold action or the Release Hold action. For details, see *Use Holds to Temporarily Stop Processing*.
- You can't include RequestHoldProcessRequest and ReleaseHoldProcessRequest in the same `soapenv:Body`.
- You can't enable a hold on a service that uses a Wait, Apply Hold, Release Hold, or Inbound operation. For example, the Wait for Shipment task places the orchestration process in a wait state that's effectively the same as a hold, so its not necessary to place a hold when the process is already waiting. If you must hold the orchestration process, then add the hold to the step immediately before the Apply Hold, Release Hold, or Inbound operation step.

Import Holds Through REST API

For details, see *Use REST API to Apply and Release Holds*.

Example Hold Codes

Here are some examples you might find useful.

Code	Name	Task Type	Service to Hold
DOO_ACTIVTY	HOLD CREATE ACTIVITY	Activity	Create Activity.
DOO_SCHDL	SCHLD	Schedule	Create Scheduling and Cancel Scheduling.
DOO_SCHLD_ALL	SCHLD_ALL	-	All scheduling services.
DOO_RSRV	RESRV	Reservation	Create Reservation and Cancel Reservation.
DOO_RSRV_ALL	RESRV_ALL	-	All reservation services.

Code	Name	Task Type	Service to Hold
DOO_SHIP_CREATE	HOLD CREATE SHIP	Shipment	Create Shipping.
DOO_SHIP_ALL	SHIP_ALL	-	All shipping services.
DOO_RCV_CREATE	HOLD CREATE RECEIVE	Return	Create Expected Receipt Advice.
DOO_INVOICE_CREATE	HOLD CREATE INVOICE	Invoice	Create Billing Lines.

Drop Shipments and Purchase Orders

If Order Management applies a hold on a fulfillment line, and if the line involves a drop shipment and a purchase order, then the Order Management work area displays a hold icon on that line. Order Management displays the icon even if it applied the hold on only on one line in the purchase order.

Assume your sales order has these fulfillment lines:

- Line 1, purchase order 1
- Line 2, purchase order 2
- Line 3, purchase order 1

If you use the Order Management work area to apply a hold on line 1, then the work area will display a hold icon on lines 1 and 3 because the entire purchase order is now on hold in Oracle Procurement. If you then release the hold on line 1 or line 3, then the work area will remove the icon from lines 1 and 3 because the entire purchase order is no longer on hold.

Other Points to Consider

- Order Management comes with a number of predefined hold codes that hold a variety of task types and services. Add your own hold code only if the predefined ones don't meet your needs.
- If you don't see the task type and the service that you need on the Services tab, then use the Manage Task Types page to specify the tasks and services that you can select for the hold. You must set the Hold Enabled option to make the service display on the Services tab. You can't enable some services for hold, such as Billing Response or Wait for Billing. For details, see *Set Up Task Types for Holds*.
- If you apply a hold on the order header, then Order Management will also apply the hold on all order lines that are currently open in that sales order, but it won't apply a hold on lines that are closed or cancelled.
- Order Management ignores a hold that you apply on a pause task. The pause task will resume according to its own conditions.

  For a configured item, make sure you apply the hold on the configured item. Don't apply a hold on a configure option. For details, see *Sales Order Hold*.
- Your downstream fulfillment systems can't finish the task until you release the hold.
- If the total shipped quantity in Order Management doesn't equal the shipped quantity in Shipping, then Order Management rejects the hold request and displays an error.
- Save time. You can migrate your hold setups. For details, see *Copy Setups Between Instances of Order Management*.
- Order Management won't process a hold request or release request for a line that's in error recovery.

Related Topics

- Set Up Task Types for Holds
- Web Services That You Can Use to Integrate Order Management
- Copy Setups Between Instances of Order Management
- Sales Order Hold
- Use Holds to Temporarily Stop Processing

Use Holds to Stop Orchestration Processes

Set up a hold code that allows your users to place a hold on a sales order.

The hold stops the service that the orchestration process references.

You can set up a hold code only for a hold that starts in Order Management. You can't use it for a hold that starts in some other application.

Summary of the Set Up

1. Create a hold code.
2. Specify job roles that manage holds.
3. Create a release reason.
4. Test your set up.

Assume you're an order administrator for Vision Corporation, and your sales engineers have informed you that a problem might exist with an item. You must create a hold code that your users can use to temporarily hold all sales orders that reference the item while your sales engineers investigate the problem.

This topic uses example values. You might need different values, depending on your business requirements.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Create a Hold Code

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Hold Codes
2. On the Manage Hold Definitions page, click **Actions > Add Row**, then set the values.

Attribute	Value
Code	hold_item_issue
Name	Hold to Investigate Item

Attribute	Value
Description	This hold allows us to investigate a problem that occurred with the item.
Hold All Services	Contains a check mark. This value specifies to apply a hold on all services. Fulfillment tasks determine whether Order Management attaches a hold code to the fulfillment line, or to the sales order for one or more fulfillment tasks that the orchestration process references.
Set	COMMON You can use the Set value to specify the business unit set that Order Management uses with a hold code. You can assign a hold code to a single business unit, or you can assign more than one hold code to sets of business units. These sets can share hold codes.

3. Optional. Specify each service where Order Management must apply the hold.

   For example, do these steps to apply the hold only for the Shipment service.

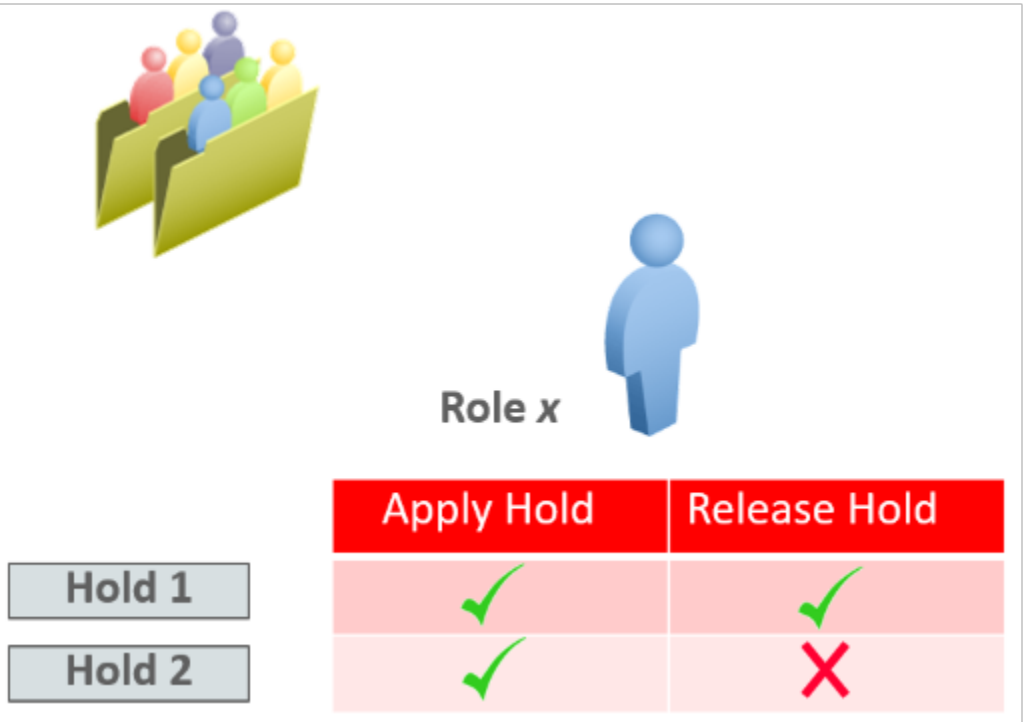
   - Make sure the Hold All Services option doesn't contain a check mark.
   - In the Hold to Investigate Item: Services area, click **Actions > Select and Add**.
   - In the Select and Add: Services dialog, leave all attributes empty, then click **Search**.
   - Click a **row** that displays Shipment in the Task Type column.
   - For this example, you must apply the hold to all services, so click **Cancel**, then add a check mark to option Hold All Services.
4. Click **Save**.

Specify Job Roles That Manage Holds

As an option, you can specify who can apply or release a hold according to the job role that the user uses when signing into Order Management.

Order Management might place a sales order on hold for a variety of reasons. Your business requirements might demand that a person who possesses the correct knowledge or authority to release a hold. Controlling holds according to job role helps you improve security by limiting functionality according to role. It also helps you decrease order processing time and reduce downstream problems that might be related to a hold that nobody releases.

For example, allow role x to apply and release hold 1, and to apply hold 2 but not release hold 2.



Assume you must allow the Order Manager job role to apply and release a hold, and allow the Order Entry Specialist job role to apply a hold but not release it.

Try it.

1. In the Details area, click **Applicable Roles**, then, in the Apply Hold area and in the Release Hold area, make sure the Selected Roles option is black.
2. Click **Actions > Select and Add**.
3. In the Search dialog, search for Order Manager, then click **Apply > OK**.
4. Click **Actions > Select and Add**.
5. In the Search dialog, search for Order Entry Specialist, then click **Apply > OK**.
6. In the Details area, set options for the roles.

Role	Apply Hold	Release Hold
Order Manager	Contains a check mark.	Contains a check mark.
Order Entry Specialist	Contains a check mark.	Doesn't contain a check mark.

7. Click **Save and Close**.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements.

For details about how to set up privileges, see *Security Reference for Order Management*.

Create Your Own Job Role

Assume users in the Vision Services department of Vision Corporation receive service calls from customers who are inquiring about the status of a sales order, such as what item the customer ordered, and expected

delivery date. The user must view the sales order, but not edit it. Assume you already created a user named order_entry_specialist_vision_services to meet this requirement.

For details, see *Privileges That You Need to Implement Order Management*.

Create your own job role.

1. Examine privileges in the predefined job roles.
    - Scan *Security Reference for Order Management*.
    - For this example, examine the Privileges section in the Job Role: Order Entry Specialist chapter, then notice the View Orders privilege.
2. Make sure you have the privileges that you need to administer IT security and users.
3. Create the job role.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Order Management
        - Functional Area: Users and Security
        - Task: Manage Job Roles
    - On the Roles page, click **Create Role**, set the values, then click **Next**.

Attribute	Value
Role Name	View Orders
Role Code	view_orders
Role Category	SCM Job Roles

    - Click **Add Function Security Policy**.
    - In the Add Function Security Policy dialog, search for View Orders, click **Add Privilege to Role**, confirm the addition, then click **Cancel**.
    - On the Create Role View Orders: Function Security Policies page, click **Next** several times until you reach the Users page.
    - On the Create Role View Orders: Users page, search for, then add the order_entry_specialist_vision_services user.

      The values you can search for depend on the users you created in your environment.
    - Click **Next > Save and Close**.
4. Get LDAP (Lightweight Directory Access Protocol) changes.
    - Go to the Scheduled Processes work area.
    - Run the *Retrieve Latest LDAP Changes* scheduled process.
    - In the Confirmation dialog, note the process number, such as 75603.
    - On the Scheduled Processes page, click **Actions > Refresh**. Continue to refresh until the status of your scheduled process is Succeeded.

5. Synchronize users and roles.
    - Go to the Setup and Maintenance work area.
    - Click **Tasks > Search**, search for, then open Run User and Roles Synchronization.

      For details about this task, see *Implementing Common Features for Oracle SCM*.
    - On the page that displays, click **Submit**.
6. Use the Manage Hold Definitions page to add your new job role to the hold code.

Create a Release Reason

Create a reason that your user can choose to indicate why they released the hold.

Try it.

1. In the Setup and Maintenance work area, go to the task.

- Offering: Order Management
- Functional Area: Orders
- Task: Manage Order Lookups

2. On the Manage Order Lookups page, enter the value, then click **Search**.

Attribute	Value
Lookup Type	DOO_HLD_RELEASE_REASON

3. In the DOO_HLD_RELEASE_REASON: Lookup Codes area, click **Actions > New** then set the values.

Attribute	Value
Lookup Code	doo_hldrel_item_ok
Display Sequence	1
Start Date	The current date.
End Date	One week from today.
Meaning	Fixed Problem with Item
Description	Fixed the problem with the item.

4. Click **Save and Close**.

Test Your Set Up

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area.
   On the Overview page, click **Actions > Manage Orders**.
3. On the Manage Orders page, enter the value, then click **Search**.

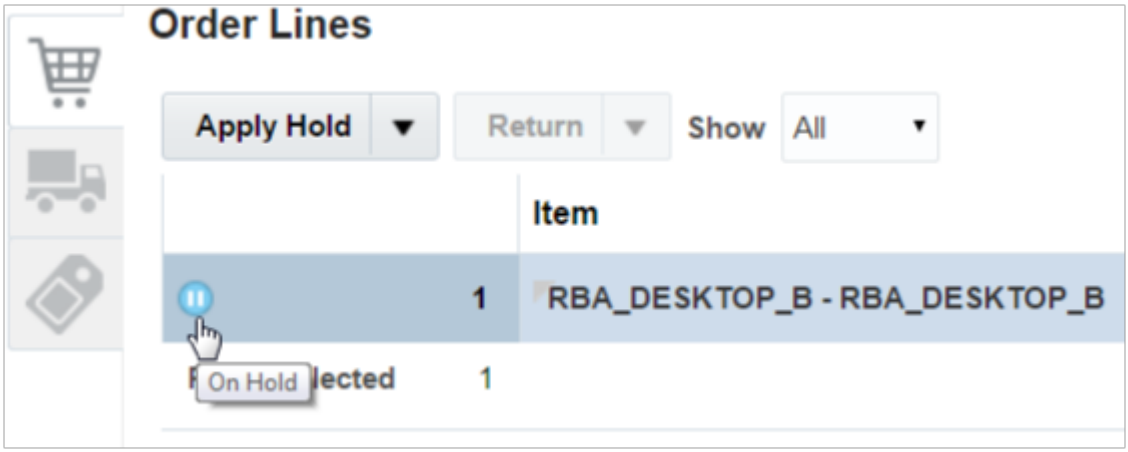
Attribute	Value
Status	Equals Processing

4. In the search results, in the Order column, click a **link**.
5. On the Order page, in the Order Lines area, click **Apply Hold**.

6. In the Apply Hold dialog, set the value, then click **Save and Close**.

Attribute	Value
Hold Name	Hold to Investigate Item This is the name of the hold that you created earlier in this topic.

7. In the Order Lines area, to verify that Order Management placed a hold on the order line, make sure it displays the On Hold icon.



8. Click the **arrow** next to Apply Hold, then click **Release Hold**.
9. In the Release Hold dialog, set values, then click **Save and Close**. You defined these values earlier in this topic.

Attribute	Value
Hold Name	Hold to Investigate Item
Release Reason	Fixed the Item Problem

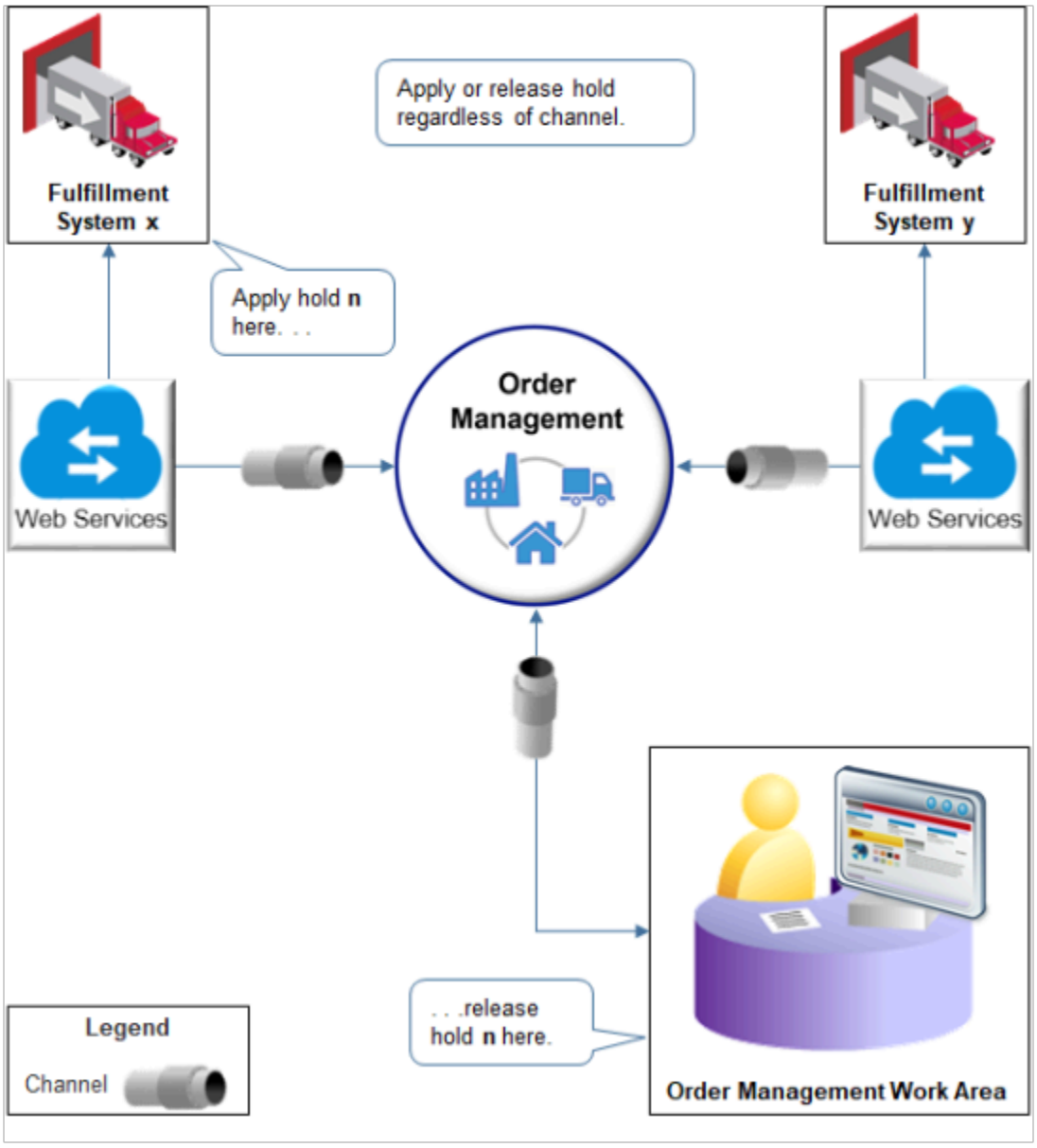
10. Sign out of Order Management, then sign back in with the Order Entry Specialist role.
11. Repeat the steps described above, but verify that Order Management doesn't display Release Hold when you click the **arrow** next to Apply Hold.

Related Topics

- Set Up User Roles and Privileges in Order Management
- Privileges That You Need to Implement Order Management
- Sales Order Hold

Manage Sales Order Holds Across Systems

Use a web service to manage sales order holds across systems and channels, including your source systems and fulfillment systems.



Note

- Apply or release a hold regardless of the channel where you applied it.
- Apply hold n in fulfillment system x, then release it in fulfillment system y.
- Use web services to communicate changes through Order Management to your fulfillment systems.
- Release holds in the Order Management work area.

Here's an example web service payload that uses the SourceHoldCode attribute to apply and release a hold.

Note

- The top line in the image is truncated. Here's the full line.

  ```
  <soapenv:Evelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:dood=href
  ```

- Release a hold regardless of the channel that applied the hold. For example, assume fulfillment system x is a channel, fulfillment system y is a channel, and the Order Management work area is a channel. Assume fulfillment system x uses a web service to apply hold on sales order 759674. You can use the Order Management work area to release the hold on sales order 759674.
- Use a web service to release a hold on a sales order header that the Order Management work area applied.
- Use a web service to apply a hold on a fulfillment line.
- Use the Order Management work area to manually release a hold that a web service automatically placed.
- Use the Order Management work area to manually apply a hold on a Draft sales order.
- Use the Order Lines tab in a fulfillment view in the Order Management work area to apply a hold on a fulfillment line. The Order page that you access from the Manage Orders page is an example of a fulfillment view.

Note the limitations.

- Release a hold on a fulfillment line only from the Order Lines tab in a fulfillment view in the Order Management work area.
- You can't use a web service to release a hold on a fulfillment line.
- You can't use a web service to apply a hold on a fulfillment line when the web service integrates with an order capture system. Most order capture systems focus on inputs to sales order, such as customer and item. They don't integrate directly with a fulfillment system, so they don't provide or manage the details that a fulfillment line hold requires.

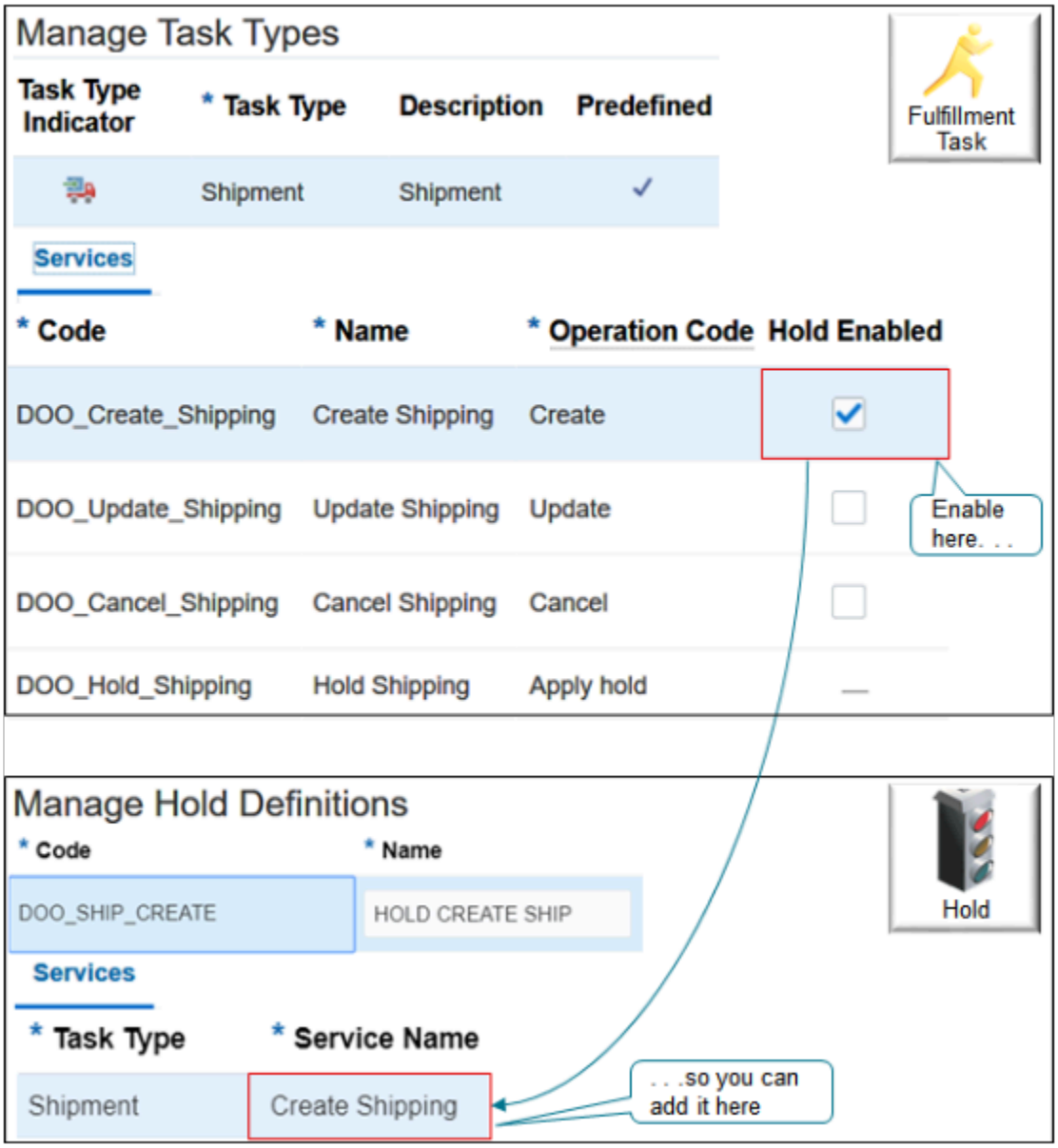
Related Topics

- Guidelines for Using Web Services to Integrate Order Management
- Sales Order Hold

Set Up Task Types for Holds

The Order Entry Specialist can select the service when placing a hold on a sales order or order line, such as the shipping service.

If you create your own task type that references the service, then you must set up the task type so it supports the hold.



Assume you create your own task type named My Shipment. You want to allow the Order Entry Specialist to place the sales order on hold when the order reaches the orchestration process step that references the My Shipment task type.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Task Types
2. On the Manage task types page, click **Actions > Add Custom**, set values, then click **Save**.

Attribute	Value
Task Type	My Shipment

Attribute	Value
Description	My Shipment

3. Click the **row** that contains My Shipment in the Task Type column.
4. Click **Services**.

   Notice that the Manage Task Types page automatically adds a create service and enables it for holds.

Code	Name	Operation Code	Hold Enabled
My Shipment Create	My Shipment Create	Create	Contains a check mark.

   Don't modify this service.
5. Add a service that applies the hold and another one that releases the hold.

   For example:

Code	Name	Operation Code	Hold Enabled
Hold My Shipping	Hold My Shipping	Apply hold	Doesn't contain a check mark.
Release Hold on My Shipping	Release Hold on My Shipping	Release hold	Doesn't contain a check mark.

6. Click **Save and Close**.

Related Topics

- How Data Flows Through Order Management
- Fulfillment Tasks
- Create Your Own Task Type
- Actions That You Can Set When Routing Requests to Fulfillment Systems
- Use Holds to Stop Orchestration Processes

Use Order Management Extensions to Apply Holds

Use an order management extension to apply a hold on the order header or on the order line.

- Use an order management extension when you have a business requirement that you can't meet through Oracle Order Management's predefined hold behavior.
- Use the On Start of Submission Request or the On Save extension point to apply your hold.

- Make sure you have the Manage Order Management Extensions (FOM_MANAGE_ORDER_MANAGEMENT_EXTENSIONS_PRIV) privilege.

You use the applyHold method to apply a hold. Your extension must check for these conditions when you use applyHold:

Apply a Hold Only	Method That You Can Use to Meet Condition
On a new sales order or a sales order that's in Draft status. You can't use applyHold with a change order or a sales order that you already submitted to fulfillment.	isFirstDraftOrder
When the order line isn't a child line of a model or kit, and when you haven't already applied a hold on the line.	canApplyHold

You might also find these methods useful.

- holdExists
- isChildLine
- Debug

For details, see *Methods That You Can Use with Order Management Extensions*.

We use the CustomerPONumber attribute as a test to make sure the extension runs only on sales orders that are on hold. In your test environment, you add the text ApplyHold to an attribute. We use the customer's purchase order number because it is a text attribute that's always available on the order header. You can use a different attribute to meet your needs.

Make sure you remove this test before you publish the extension to your production environment:

```
def poNumber = header.getAttribute("CustomerPONumber")

if (poNumber != "ApplyHold") return;
```

For more:

- *Guidelines for Setting Up Holds on Sales Orders*
- *Overview of Creating Order Management Extensions*
- Click *here* to see a nifty demonstration.

Set Up Profile

If you want to apply a hold on the order header, then you must enable a profile option. It's optional if you want to apply a hold on the order line. If you enable the profile, then Order Management uses a faster and more efficient way to process your holds. We recommend that you enable it even if you apply a hold only on the order line.

1. Create a profile option.
    - Go to the Setup and Maintenance work area, click **Tasks > Search**, then search for Manage Profile Options.
    - On the Manage Profile Options page, in the search results, click **Actions > New**.
    - On the Create Profile Option page, set the values.

Attribute	Value
Profile Option Code	FOM_NEW_HOLDS_PROCESSING
Profile Display Name	FOM New Hold Processing
Application	Order Management
Module	Manage Orders
SQL Validation	select MEANING, LOOKUP_CODE from FND_LOOKUPS where LOOKUP_TYPE='YES_NO'
Start Date	Today or a date in the future.

- In the Profile Option Levels area, set the values.

Attribute	Value
Level	Site
Enabled	Contains a check mark.
Updatable	Contains a check mark.

- Click **Save and Close**.

2. Administer the profile value.
   - On the Overview page, search for and open the Manage Administrator Profile Values task.
   - On the Manage Administrator Profile Values page, search for the value.

Attribute	Value
Profile Option Code	FOM_NEW_HOLDS_PROCESSING

   - In the Profile Values area, set the values, then click **Save and Close**.

Attribute	Value
Profile Level	Site
Profile Value	Yes

Apply Hold on Order Header

Increase your efficiency. Apply a single hold on the order header when your sales order has more than one order line instead of applying a separate hold on each line. This is particularly helpful when your order has hundreds or even thousands of lines. You can apply a single hold on the order header instead of applying a separate hold on each line.

This example applies a hold on an order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.HoldResult;

// Run the extension only on sales orders that are on hold.
def poNumber = header.getAttribute("CustomerPONumber")
if (poNumber != "ApplyHeaderHold") return;

if (!isFirstDraftOrder()) {
 // Make sure the order is in draft status, and not already submitted to fulfillment.
 debug("order is not in draft status");
 return;
}

// Make sure the current user has the privilege they need to apply hold. You can remove this code to meet
 your needs.
if (!context.isUserInRole("ORA_DOO_ORDER_MANAGER_JOB")) {
 debug("user does not have privilege");
 // Display a warning message that the user does doesn't have the privilege to apply a hold. As an option,
 you can display an error message or
 // silently return from the extension without raising any error or warning.
 throw new ValidationException(new Message(Message.MessageType.WARNING, "User " + context.getUserName() + "
 does not have privilege to apply hold"));
}

if (canApplyHold(header, "DOO_SHIP_ALL")) {
 debug("Applying hold");
 def holdResult = header.applyHold("DOO_SHIP_ALL", "Apply Header Hold through Extension"); // Create a hold
 and use the DOO_SHIP_ALL hold code.
 def status = holdResult.getStatus();
 def message = holdResult.getMessage();
 if ("FAILURE".equals(status)) {
 throw new ValidationException(message);
 }
}
```

Apply Hold on Order Line

If you don't enable the FOM_NEW_HOLDS_PROCESSING profile, then you must explicitly set the value for the HoldComments attribute:

```
def hold = line.applyHold("DOO_RSRV");
 hold.setAttribute("HoldComments", "comment"
```

This example applies a hold on an order line where the line's quantity is greater than 10.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

// Run the extension only on sales orders that are on hold.
def poNumber = header.getAttribute("CustomerPONumber")

if (poNumber != "ApplyHold") return;

if (!isFirstDraftOrder()) {
 // Make sure the order is in draft status, and not already submitted to fulfillment.
 debug("order is not in draft status");
 return;
}

// Make sure the current user has the privilege they need to apply hold. You can remove this code to meet
 your needs.
if (!context.isUserInRole("ORA_DOO_ORDER_MANAGER_JOB")) {
 debug("user does not have privilege");
```

```
 // Display a warning message to explain that the user doesn't have the privilege they need to apply a hold.
 You can remove this code to meet your needs.
 throw new ValidationException(new Message(Message.MessageType.WARNING, "The " + context.getUserName() + "
 user doesn't have the privilege they need to apply a hold."));
}

def lines = header.getAttribute("Lines");

while (lines.hasNext()) {
 def line = lines.next();

 // Make sure the ordered quantity is greater than 10.
 BigDecimal qty = line.getAttribute("OrderedQuantity");
 if (qty.compareTo(new BigDecimal(10)) > 0) {
 if (canApplyHold(line, "DOO_RSRV")) {
 debug("Applying hold");
 def hold = line.applyHold("DOO_RSRV"); // Create a hold and use the DOO_RSRV hold code.
 // Display text that explains what we're doing.
 hold.setAttribute("HoldComments", "We're holding this line so we can manually review and approve it.");
 }
 }
}
```

Apply Hold on Order Line when You Enable the Profile

If you enable the FOM_NEW_HOLDS_PROCESSING profile, then you can implicitly set the value for the HoldComments attribute: `line.applyHold("DOO_RSRV", "comment");`.

For example:

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.HoldResult;
def poNumber = header.getAttribute("CustomerPONumber")
if(poNumber != "ApplyHold") return;
if( !isFirstDraftOrder() ) {
 debug("order is not in draft status");
 return;
}
if( !context.isUserInRole("ORA_DOO_ORDER_MANAGER_JOB") ) {
 debug("user does not have privilege");
 throw new ValidationException( new Message(Message.MessageType.WARNING, "User " + context.getUserName() + "
 does not have privilege to apply hold") );
}

def lines = header.getAttribute("Lines");

while(lines.hasNext()) {
 def line = lines.next();
 BigDecimal qty = line.getAttribute("OrderedQuantity");
 if(qty.compareTo(new BigDecimal(10)) > 0) {
 if(canApplyHold(line, "DOO_RSRV")) {
 debug("Applying hold");
 def holdResult = line.applyHold("DOO_RSRV", "We're holding this line so we can manually review and approve
 it."); // Create a hold and use the DOO_RSRV hold code
 def status = holdResult.getStatus();
 def message = holdResult.getMessage();
 if("FAILURE".equals(status)) {
 throw new ValidationException(message);
 }
 }
 }
}
```

Apply Hold on Shipment Set

This example applies a hold on all lines in a shipment set where the ordered quantity is greater than 10.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

// These two lines are only for testing. They make sure the extension applies only when customer purchase
 order number is ApplyHold
def poNumber = header.getAttribute("CustomerPONumber")
if(poNumber != "ApplyHold") return;

if( !isFirstDraftOrder() ) {
 // Make sure the sales order is in draft status.
 debug("order is not in draft status");
 return;
}

// Make sure the current user has the privilege they need to apply a hold. This is optional.
if( !context.isUserInRole("ORA_DOO_ORDER_MANAGER_JOB") ) {
 debug("user does not have privilege");
 // Display a warning message here to explain that the user doesn't have the privilege to apply a hold. This
 is optional.
 throw new ValidationException( new Message(Message.MessageType.WARNING, "User " + context.getUserName() + "
 does not have privilege to apply hold") );
}

// Map the lines that are on hold. We will populate this map while we put lines on hold.
Set linesOnHold = new HashSet<Long>();
Set shipsetsToHold = new HashSet<String>();

def lines = header.getAttribute("Lines");

while(lines.hasNext()) {
 def line = lines.next();

 // Make sure the line is eligible to apply a hold. Make sure the line has an ordered quantity that's
 greater than 10.
 BigDecimal qty = line.getAttribute("OrderedQuantity");
 if(qty.compareTo(new BigDecimal(10)) > 0) {
 if(canApplyHold(line, "DOO_RSRV")) {
 debug("Applying hold");
 def hold = line.applyHold("DOO_RSRV"); // Create a hold and use the DOO_RSRV hold code.
 hold.setAttribute("HoldComments", "We're holding this line so we can manually review and approve it.");
 Long fulfillLineId = line.getAttribute("FulfillmentLineIdentifier");
 linesOnHold.add(fulfillLineId);

 String shipsetName = line.getAttribute("ShipSetName");
 if( shipsetName != null ) {
 // The line that we're hold is part of a shipment set. Let's remember the shipset name in a set. Later we
 will make another pass over the lines to find other
 // lines in the same shipment set and hold them too.
 shipsetsToHold.add(shipsetName);
 }
 }
 }
}

// We are done making one pass over the lines. Some of the lines that we held might be part of a shipment
 set. Let's make another pass over all lines and hold the other lines
// from the shipment sets.

lines = header.getAttribute("Lines");

while(lines.hasNext()) {
```

```
def line = lines.next();
Long fulfillLineId = line.getAttribute("FulfillmentLineIdentifier");
if( linesOnHold.contains(fulfillLineId) ) {
// We already held this line during the first pass. Let's continue to the next line in the loop.
continue;
}

String shipsetName = line.getAttribute("ShipSetName");
if( shipsetName != null ) {
// The line is in a shipment set. Let's see whether we should hold this shipment set. If yes, then we will
hold the line.
if( shipsetsToHold.contains(shipsetName) ) {
if(canApplyHold(line, "DOO_RSRV")) {
debug("Applying hold");
def hold = line.applyHold("DOO_RSRV"); // Create a hold and use the DOO_RSRV hold code
hold.setAttribute("HoldComments", "We're holding this line so we can manually review and approve it.");
}
}
}
}
```

Use REST API to Apply and Release Holds

Use the applyHold operation and the releaseHold operation on the Sales Orders for Order Hub REST API to apply and release holds on sales orders and fulfillment lines.

Apply and release a hold on:

- A draft or submitted sales order
- One or more sales orders or fulfillment lines in a single request
- The sales order entity
- The fulfilment line entity

Release a hold that you applied on the order header or fulfilment line through some other channel, such as through the View Order page or a fulfilment view in the Order Management work area, or a header hold that you applied through a SOAP service.

Note

- You can use REST API to apply or release a hold on one or more sales orders or on one or more fulfillment lines.
- You can't use REST API to apply a hold when you create a sales order. The order must already be in draft or submitted status.
- If you apply a hold on an order line through a SOAP service, then you can't use REST API to release it.

For details, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Sales Orders for Order Hub > Holds**.

For more, see *Guidelines for Setting Up Holds on Sales Orders*.

Try It

1. Go to the Setup and Maintenance work area, then click **Tasks > Search**.
2. Search for, then open the Manage Profile Options task.

3. On the Manage Profile Options page, click **Actions > New**.
4. On the Create Profile Option page, set the values, then click **Save and Close**.

Attribute	Value
Profile Option Code	FOM_NEW_HOLDS_PROCESSING
Profile Display Name	FOM_NEW_HOLDS_PROCESSING
Application	Order Management
Module	Order Management
SQL Validation	`select MEANING , LOOKUP_CODE from FND_LOOKUPS where LOOKUP_TYPE='YES_NO'` Make sure you set LOOKUP_TYPE to `'YES_NO'`. Don't set it to any other value.

5. In the Profile Option Levels area, make sure the Site level is enabled.
6. Go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Administrator Profile Values
7. Search for the value.

Attribute	Value
Profile Option Code	FOM_NEW_HOLDS_PROCESSING

8. In the search results, set the value.

Profile Level	Value
Site	Yes

9. Add the hold request to your REST API payload.

Examples

Here's an example request payload that applies the DOO_SHIP_ALL hold on the order header for source order AASHIP190404.

```
{
 "processRequestOfflineAfter":"240",
 "holdRequests": [
 {
 "HoldCode":"DOO_SHIP_ALL",
 "HoldComments":"Order On Hold",
 "SourceTransactionId":"AASHIP190404",
 "SourceTransactionSystem":"GPR"
 }
 ]
```

```
}
```

Here's the REST API response from that hold request.

```
"result"
 {
 "RequestStatus":"SUCCESS",
 "SourceTransactionSystem":"GPR"
 "SourceTransactionId":"AASHIP190404",
 "OrderNumber":"531258",
 "SourceTransactionLineId": null,
 "FulfillLineId": null,
 "HoldCode":"DOO_SHIP_ALL"
 }
]
```

Here's an example request payload that releases the DOO_SHIP_ALL hold on the order header for source order AASHIP190404.

```
{
 "processRequestOfflineAfter": 240,
 "holdRequests": [
 {
 "SourceTransactionSystem" : "GPR",
 "SourceTransactionId": "AASHIP190404",
 "HoldCode": "DOO_SHIP_ALL",
 "HoldReleaseComments": "Order released from hold",
 "HoldReleaseReasonCode": "QAREL"
 }
 ]
}
```

Here's the response.

```
{
 "result": [
 {
 "RequestStatus": "SUCCESS",
 "SourceTransactionSystem": "GPR",
 "SourceTransactionId": "AASHIP190404",
 "OrderNumber": "531257",
 "SourceTransactionLineId": null,
 "FulfillLineId": null,
 "HoldCode": "DOO_SHIP_ALL"
 }
 ]
}
```

SourceTransactionLineId Attribute

Use the SourceTransactionLineId attribute to identify the line.

- To apply a hold on the order header, include a value for SourceTransactionId but not for SourceTransactionLineId and FulfillLineId. Order Management will apply the hold on the header and all order lines in the sales order.
- To apply or release a hold on the fulfillment line, you must include a value for FulfillLineId in the order line entity. You can also use SourceTransactionLineId in the order line entity to apply or release a hold on all fulfillment lines for the order line.
- REST API will apply the hold only on the fulfilment lines that reference SourceTransactionLineId.
- If you use SourceTransactionLineId, then REST API will apply the hold only on fulfillment lines that are part of the order line entity, and won't apply a hold on the order line entity.

Process Your Requests in the Background

You can include the ProcessRequestOfflineAfter attribute in your REST API request to specify the number of seconds to wait before sending control back to your calling application, and then run the request in the background.

- REST API will process the request immediately regardless of how you set ProcessRequestOfflineAfter.
- You will need to query the sales order to monitor the request's progress.

Here are the values that you can use.

ProcessRequestOfflineAfter Value	What REST API Does
0 (zero), or any value that's less than -1 (negative 1).	Process the request immediately in the background.
-1 (negative 1).	Time out after 5 minutes, then continue to process the request in the background.
Any value that's 1 to 240.	Time out after the number of seconds that you specify, then continue to process the request in the background.
Any value that's more than 240.	Time out after 240 seconds, then continue to process the request in the background.

Timeout

If your REST API request to apply or release a hold times out for some reason after 240 seconds, then you might encounter a message that's similar to:

```
Order management is processing your request to apply or release a hold on the sales order. Query for order {ORDER_NO} to monitor your request.
```

if the request times out, then Order Management will continue to process it in the background and will send this message in the REST response. If you encounter this message, then use REST API to query the sales order again to determine whether the request was successful. If it isn't successful, wait a few minutes, then try again.

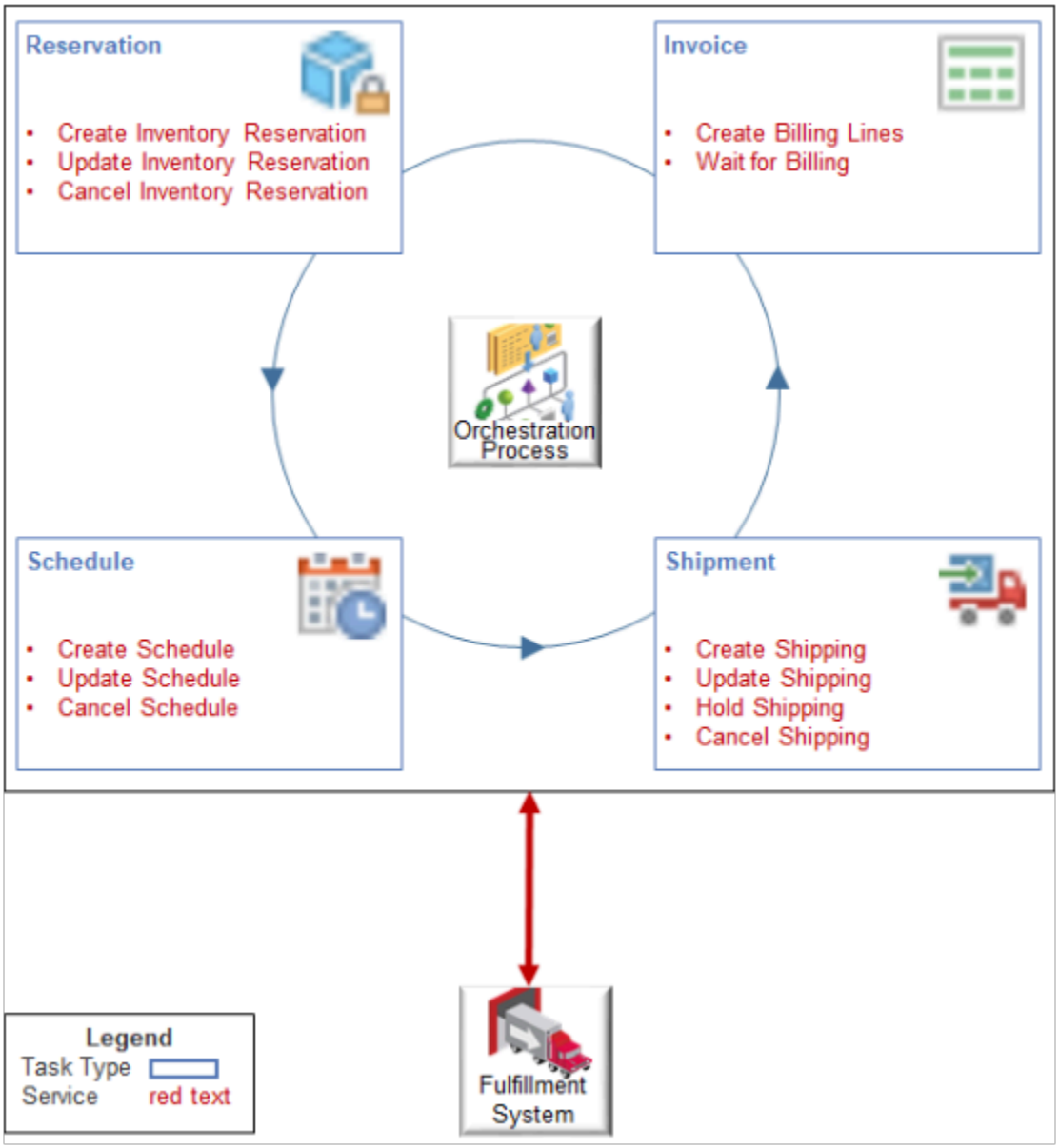
Fulfillment Task

Fulfillment Tasks

Use a task type to specify the type of fulfillment task that the orchestration process step does, such as schedule a fulfillment line for shipment, ship it, or confirm delivery.

A task type is a group of services that Order Management uses to do a fulfillment task. It represents the work that Order Management does to process a sales order from the time your user creates the sales order to the time that Order Management sends the order to your order fulfillment system.

Here are some typical task types, the services they call, and the sequence that Order Management uses to run them. This example starts with a reservation.



Note

- Each predefined task type comes with a set of services. For details, see *Task Services*.
- Reservation is an example of a task type. It contains a set of services that communicate with your fulfillment system to reserve inventory according to the Item attribute and Quantity attribute of your fulfillment line. For example, it uses the Create Inventory Reservation service to reserve supply in inventory.
- A typical sequence is to use services to reserve inventory for your item, schedule it for shipping, ship it, then invoice it.
    - Create Inventory Reservation
    - Create Schedule
    - Create Shipping
    - Create Billing Lines

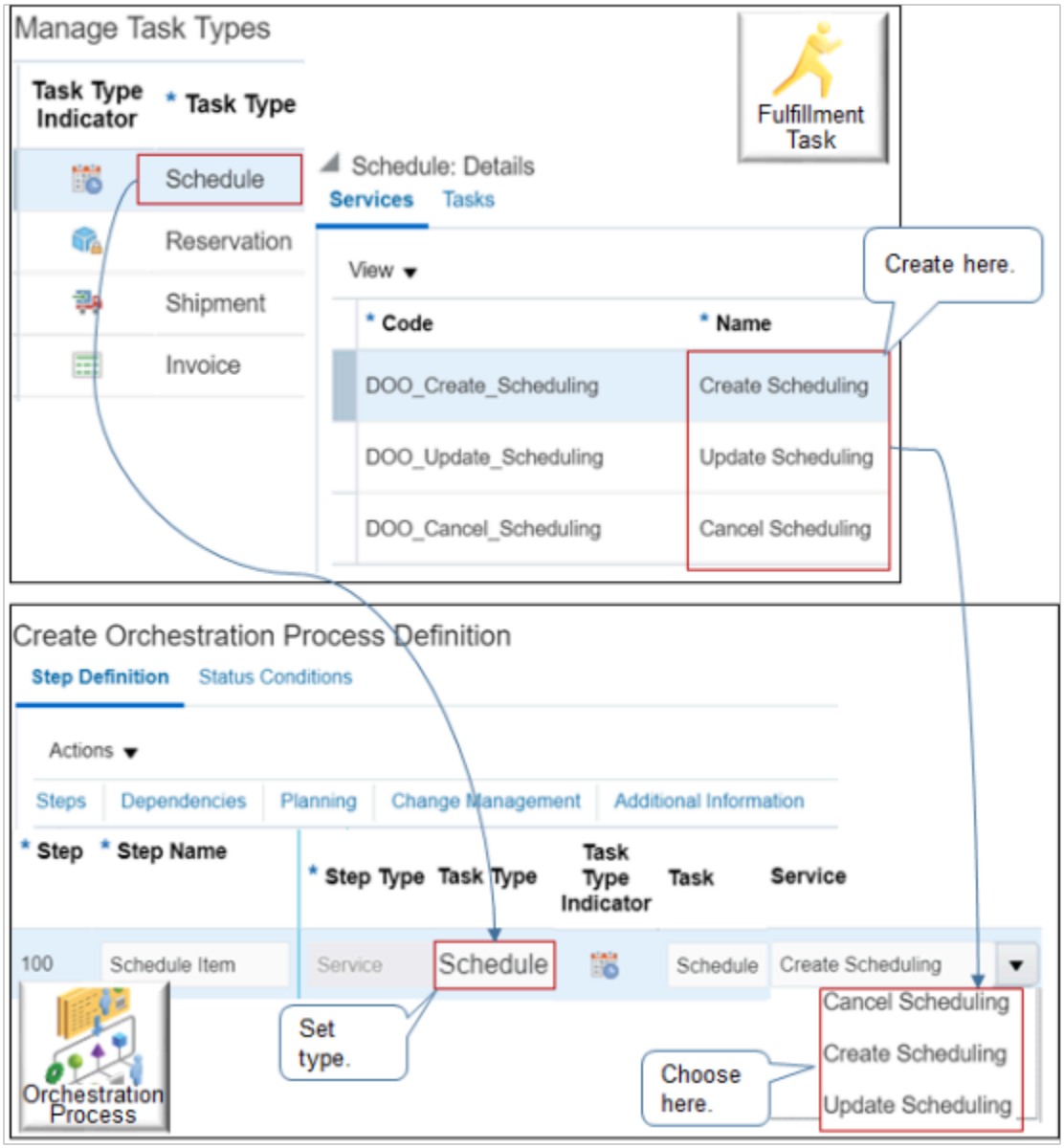
You can:

- Use a task to represent the services that the task type references. For example, if you create a Ship Goods task that references services from the Shipment task type, then the Order Management work area displays Ship Goods when a Shipment service runs, regardless of whether it calls the Create Shipment service or the Update Shipment service. The Order Management work area doesn't display the service. It displays only the task.
- Create a new task type.
- Add more than one task to your new task type, such as ShipGoods or ShipWidgets.
- Add a predefined service to your new task type.
- Edit the service names of an activity or task type that you create.

You can't:

- Modify the name of a task type after you create it because other objects in your setup might reference the name.
- Modify or delete a predefined task type.
- Edit or delete a predefined service.

The Create Orchestration Process Definition page references the Manage Task Types page to get the values that it displays for the task, type task, and service that you can choose on each step.



In this example, the orchestration process references the Schedule task type, and it also references the Schedule task type's services.

Predefined Task Types

Here are some of the predefined task types that Order Management typically uses.

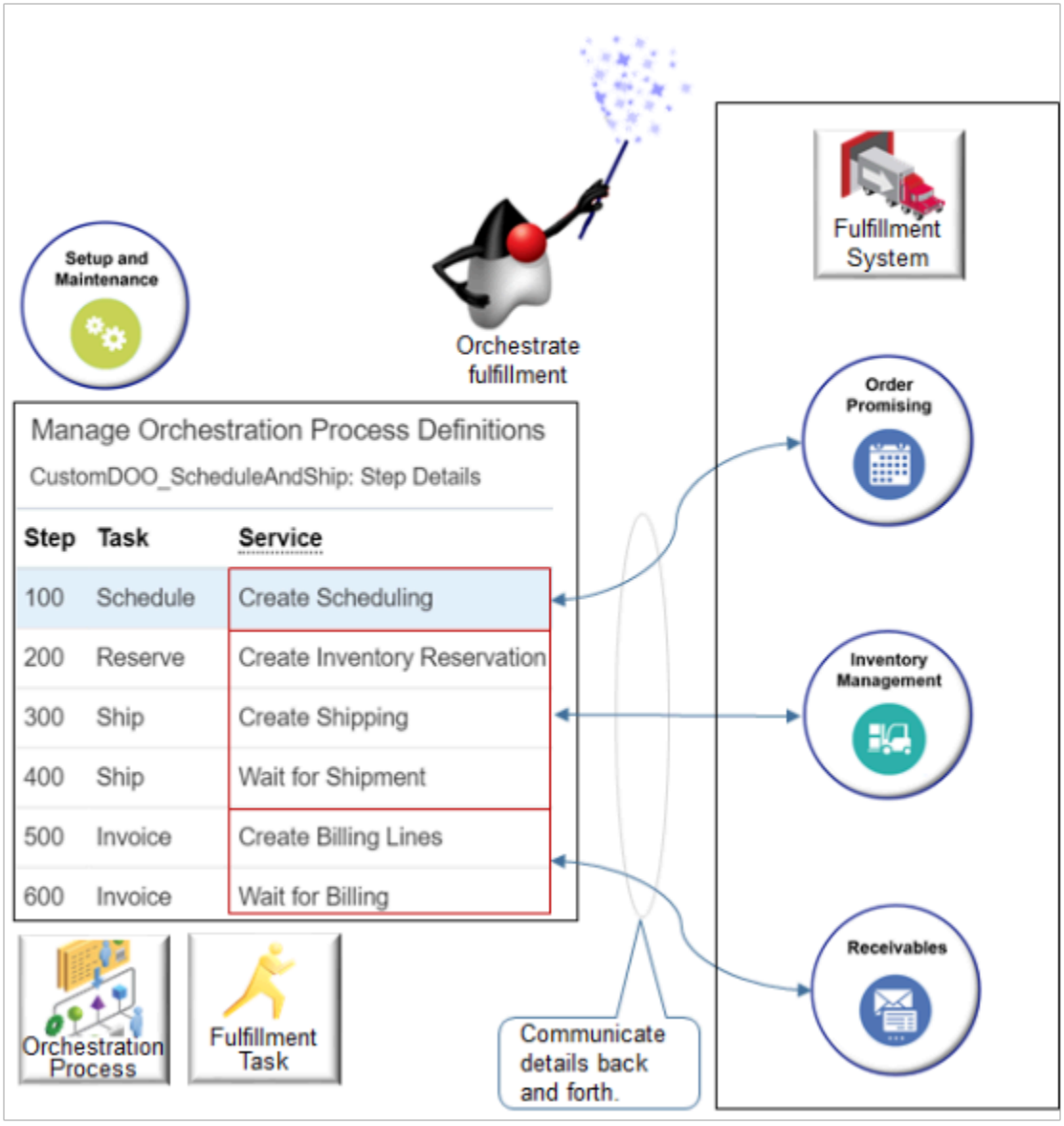
Task Type	Description
Schedule	Schedule the fulfillment line.
Reservation	Reserve inventory for the item.

Task Type	Description
Shipment	Communicate with your fulfillment system to ship the item.
Invoice	Communicate with your billing system to create an invoice for the fulfillment line.
Return	Communicate with your fulfillment system to return the item.
FulfillOrder	Integrate between Order Management and an Enterprise Resource Planning (ERP) system. Note that fulfillment tasks can run more than one fulfillment action through a single request, such as shipment and invoice.
Pause	Temporarily pause processing to wait until a date to occur, an event to occur, or a dependency to resolve before proceeding to the next orchestration process step. Use the pause task to coordinate processing across more than one fulfillment line in one sales order.
Activity	Communicate with your fulfillment system to manage a human activity, such as manually install the AS54888 Desktop Computer.
DOO_Procurement	Source and ship the item when you procure the item from an organization that resides outside of your typical supply chain.
DOO_Supply	Communicate with Oracle Supply Chain Orchestration so you can use more complex logic when you source the item.
DOO_Subscription	See *Set Up Orchestration Processes for Coverage Items*.

To reduce set up time and maintenance, we recommend that you use predefined task types. Create a new task type only if the predefined ones don't meet your needs. For details, see *Create Your Own Task Type*.

Task Services

Use different task services to fulfill different types of fulfillment requests.



Note

- You set the task service on the orchestration process step.
- Each service does a specific fulfillment task, such as Create Scheduling.
- The service communicates a fulfillment request to an application in your fulfillment system. For example:
  - The Create Scheduling service communicates with Order Promising.
  - The Create Inventory Reservation, Create Shipping, and Wait for Shipment services communicate with Inventory Management.
  - The Create Billing Lines and Wait for Billing services communicate with Receivables.
- Fulfillment systems communicate updates back to the services, such as status.
- Communication occurs through the interface.
- You can set up other fulfillment systems. For details, see *Overview of Integrating Order Management*.

Activity Services

Activity services send an activity request to your fulfillment system. The fulfillment system creates and fulfills the activity, then sends replies and updates to the activity service. The activity service interprets these replies and updates.

An activity is an event that occurs outside of Oracle Order Management. For example, an orchestration process might include an activity task type to set up a network router.

- An activity contains the details needed to finish the task.
- Your users can do an activity as part of finishing order fulfillment. Order Management can assign an activity to one of your users.
- Each activity includes attributes, such as subject, activity type, earliest start date, due date, scheduled duration, actual duration, percent complete, and assignee.
- Order Management can associate an activity with one or more fulfillment lines.
- The activity service doesn't typically immediately fulfill an activity task, so a wait service allows the orchestration process to wait for the activity to finish.
- Order Management doesn't support partial fulfillment. An activity step must finish before the orchestration process can proceed to the next orchestration process step.

Note

Service Feature	Description
Send request to fulfillment system.	Send a Create Activity request to the fulfillment system that creates the activity. If the service receives a change order, then it changes or cancels the activity, as necessary. If Order Management applies a hold on a sales order, then the activity service sends a request to the fulfillment system to hold the activity that's currently in progress.
Receive activity status update.	You can schedule an orchestration process that periodically gets the updated activity status.
Modify activity.	• An activity is a predefined task type. You can also create a new activity task type, and use an extensible flexfield to modify an activity. • You can enable an activity default in a task type or orchestration process step. Default the activity type so you can categorize activities, then your fulfillment system can run business logic and do validation according to the activity type. • If you don't set the default value for the task type or the orchestration process step, then Order Management sets the subject of the activity to the name of the step. • You can use some fulfillment systems to create an activity template that supports a human task. To use the template, you can specify the activity template for a task type or an orchestration process step that creates the activity according to the predefined template when you define the task type and the orchestration process.

Reserve, Schedule, and Hold Services

Use these services to orchestrate the item in inventory.

Service	Description
Reserve services	Send a reservation request to the part of your fulfillment system that manages inventory. A reservation reserves the supply for a sales order so no other sales order or inventory system can use the supply.
Schedule services	Send a scheduling request to order promising. For example, to schedule a sales order, remove the schedule from a sales order, or determine whether the item is available. • Scheduling applies to fulfillment lines that are waiting for manual scheduling and fulfillment lines that fail scheduling in the automated or manual process. • Scheduling works only for fulfillment lines that aren't scheduled. Order Management doesn't allow automatic rescheduling from the Order Management work area.
Hold services	Send a hold request to the fulfillment system. For example, the Hold Shipping service can send a hold request from Order Management to the part of your fulfillment system that does shipping.

Shipment Services

Shipment services send a shipment request to the part of your fulfillment system that does shipping.

Service Feature	Description
Send shipment request to fulfillment system.	If Order Management receives a change order, then the shipment service changes or cancels the shipment request, if necessary. If someone applies a shipping hold, and if the Hold on Running Task option is enabled, and if the task that's running is a Shipping task, then the shipment service sends a request to the shipping system to hold the shipment request that its currently processing.
Consolidate fulfillment lines.	Consolidate the fulfillment lines of a shipment set or a configured item, then send all the lines of the shipment set or the configured item as a group to the shipping system.
Receive fulfillment line details and status updates from the shipping system and update business objects in Order Management.	The shipping system might send fulfillment line details to Order Management when a status update occurs, including before it confirms the shipment. These details might include freight cost, tracking number, way bill number, and so on. Order Management interprets the update it gets from the shipping system, then uses one of these predefined values to update status. • Picked • Packed • Shipped • Backordered Note • The shipment service continues to interpret the updates it receives from the shipping system even after the shipping system ships fulfillment lines. It sends details about these updates to Order Management. • If the shipping system uses more than one currency to represent cost, then the shipment service converts them before it sends the update to Order Management. • If the shipping system uses a unit of measure to represent shipping that's different from the unit of measure that Order Management uses in the sales order, then the shipment service converts

Service Feature	Description
	the unit of measure back to the unit of measure that the sales order uses, then communicates the shipped quantity to Order Management.
Split a fulfillment line, shipment set, or configured item when only part of a shipment ships.	If only part of a fulfillment line ships, then the shipment service splits the line into. • One fulfillment line that includes the quantity that shipped • Another fulfillment line that includes the quantity that didn't ship If only some fulfillment lines ship for a. • **Shipment set**. The shipment service removes the lines that didn't ship from the shipment set. • **Configured item**. The shipment service splits the configured item into a shipped item and an item that hasn't shipped.

Invoice Services

Invoice services send a request to the part of your fulfillment system that does billing and interprets the replies it receives from this system.

- The billing system creates the invoice and the credit transactions.
- Order Management doesn't allow you to modify a fulfillment line after the invoice service sends the request.
- If you use a source system that resides outside of Order Management, then the source system must provide most of the data that the billing system requires to finish billing. Order Management stores these details, then routes the billing request to the billing system.

Service Feature	Description
Send fulfillment line details from the fulfillment system.	The Create Billing Lines service sends billing details from the sales order or the return order each time the fulfillment line is eligible for billing. For example, it sends discounts, charges, tax attributes, sales credits, and fulfillment details. • An invoice service sends charges that occur in the order header with the first fulfillment line that Order Management fulfills for the sales order. • If the fulfillment line doesn't include payment details or sales credits, then the invoice service sends these details from the order header. • The invoice service sends prepayment details from the order header for all fulfillment lines. • Order Management doesn't support discounts that occur in the order header.
Return lines	The invoice service sends the reference to the original sales order line, the return reason, received quantity, and delivered quantity.
Shipment set or configured item	The invoice service sends the fulfillment lines that the shipment set or the configured item contains together. If the fulfillment system fulfills only some lines in the shipment set or configured item, then the invoice service sends only the fulfilled lines.

Note: A change order from Order Management can't update an invoice. A change to an invoice is typically a credit from a return order or prepayment. Its not typically a cancel.

The billing system processes the data that it receives, then sends details to the invoice service.

- Invoice details
- Credit memo details
- Billing amount
- Billing date
- Invoice date or credit memo date
- Number
- Status
- Legal entity details

It sends one of these statuses.

- Await Billing
- Billed

Return Services

Return services send a request to the part of your fulfillment system that does receiving and interprets the reply and update that it receives from the receiving system.

- The return service creates a change receipt advice or a cancel receipt advice when Order Management receives a return request.
- The request might include one or more attribute updates, such as to increase the receipt quantity.
- If Order Management receives a request to change the original copy of the sales order that the customer returned, then the return service sends a request to the fulfillment system that creates the receipt advice.
- If Order Management receives a request to cancel the original copy of the sales order line that the customer returned, then the return service cancels the receipt advice. Order Management typically allows cancel until the fulfillment system receives the returned items.
- If the ordered quantity is greater than the delivered quantity on the receipt advice, and if the customer doesn't require the ordered quantity, then the return service can request to cancel the remaining quantity.

How Return Services Handle Partial Returns

Return services can process a partial receipt, such as the return of only some items of a configured item.

If the customer returns only part of the return, then the return service splits the fulfillment line into two lines.

- One line includes a status of Delivered for the items the customer returned
- One line includes the items the customer didn't return

If the customer returns only part of the original order, and if the return includes a configured item or kit, then the return service splits the fulfillment line into two orchestration groups.

- One group includes the fulfillment lines that the customer returned
- One group includes the fulfillment lines that the customer didn't return or that aren't returnable

How Return Services Handle Events

The receiving system that starts the event might send a status update for the return. For example, if the receiving dock receives the item, then the receiving system might send a status update that starts an event that creates the receipt advice. Here are the events in the receiving system that might start a status update in Order Management.

- Receive the item on the receiving dock when the receipt is created.
- Deliver the item into inventory.
- Return the item to a customer.
- Correct the sales order after a receipt transaction occurs. For example, a customer can't return a deliver transaction.

Fulfill Order Services

Fulfill order services send a request to and receives a status update from your fulfillment system. They can also send a request to and receive a status update from a system that manages enterprise resource planning (ERP).

Fulfill order services can send a request that modifies a sales order that resides in Order Management, and that the fulfillment system uses.

- Create.
- Update.
- Place hold.
- Release hold.
- Update status.
- Cancel.

Fulfill order services also.

- Send an update to the fulfillment system each time Order Management accepts a change order that affects fulfillment.
- Receive interim and final status updates from the fulfillment system. Fulfill order services don't immediately send a reply. They send the reply when the fulfillment activity runs.

For details, see *Actions That You Can Set When Routing Requests to Fulfillment Systems*.

Related Topics

- How Data Flows Through Order Management
- Fulfillment Tasks
- Create Your Own Task Type
- Actions That You Can Set When Routing Requests to Fulfillment Systems

Create Your Own Task Type

Create your task type to specify the services that you use to finish a fulfillment task.

Assume you need to create a new task type named Get_Customer_Acceptance.

1. Go to the Setup and Maintenance work area, then go to the task.

- Offering: Order Management
- Functional Area: Orders
- Task: Manage Task Types

2. Add the task type.
   - On the Manage Task Types page, click **Actions > Add Custom**.
   - The page adds a new row. Enter a value.

Attribute	Value
Task Type	Get_Customer_Acceptance

   - Step out of the Task Type attribute.

Notice that the page adds services in the Services list. The page copies the value that you enter in the Task Type attribute, appends it with the type of service, such as Create, then inserts the value in the Code column of the service. For example:

Code	Name
Get_Customer_Acceptance Create	Get_Customer_Acceptance Create
Get_Customer_Acceptance Inbound	Get_Customer_Acceptance Inbound

Note

- One service references the outbound Create operation code.
- Another service references the Inbound operation code.
- You must create at least one task for each new task type.
- You can specify a name for each service, and you can add a service that references some other operation code, such as Change, Get Status, Apply Hold, Release Hold, or Cancel.
- If you step out of the Task Type attribute, and then come back to the Task Type attribute and change the value, then the page doesn't update values in the Code column, and values between the task type and the services it references will be different. If you then click **Save**, the Code attribute in the Services list becomes read-only and you can't change it.

  The task type and its services will work, but having code names that are different from the task type might be confusing in other parts of your set up. So, we recommend that if you change the Task Type attribute, that you also change the Code in the Services list before you click Save.

  For example, assume you create a new task type, set the Task Type attribute to `Get_Customer_Acceptance`, tab out of the attribute, the page adds the `Get_Customer_Acceptance Create` service, you change the Task Type attribute to `Get_Customer_Acceptance_During_Drop_Ship`, but the Code attribute in the service is still

`Get_Customer_Acceptance Create`. We recommend that you change the Code column in the services lists to `Get_Customer_Acceptance_During_Drop_Ship Create` before you click **Save**.

3. Assign a status code to the task type.

   Order Management sets a default value for some system status codes, such as Pending, Change Pending, Cancel, or Canceled. The status code that each task type references also controls the values for the exit criteria on a wait step that uses the task type, and the value of the task status in the next orchestration process step. You can create a new status code, or you can assign a status code that already exists. For details, see *Manage Status Values*.
4. Manage the task status condition. For details, see *Manage Task Status Conditions*.
5. Click **Save**.
6. Connect Order Management to the fulfillment system that will do the tasks and services that your new task type references. For details, see *Overview of Connecting Order Management to Your Fulfillment System*.
7. Reference your new task type when you create the orchestration process step.

   Reference it in the same way that you reference a predefined task type.

For details about exit criteria, branches, wait steps, and using a task type in an orchestration process, see*Overview of Orchestration Processes*.

Send a Response to Your Fulfillment System

You must use a web service to send a response to your fulfillment system.

- The payload that you use is different depending on the service that you use.
- These services can provide an immediate response or a delayed one.
- You use one WSDL for an immediate response and a different WSDL for a delayed response.

For details, see *Connect Order Management to Your Fulfillment System*.

Fulfill Order Response Service

Here's an example.

```
<ns1:FulfillmentRequest
xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/taskLayer/fulfillOrder/
DooTaskFulfillOrderResponseInterfaceComposite">
 <ns1:FLine
xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/common/process/model/">
<ns2:SourceOrderSystem>LEG</ns2:SourceOrderSystem>
 <ns2:FulfillLineId>300100135839049</ns2:FulfillLineId>
 <ns2:Status>COMPLETED</ns2:Status>
 <ns2:TaskType>my_task_type</ns2:TaskType>
 </ns1:FLine>
</ns1:FulfillmentRequest>
```

where

- LEG is an example name of a source system. The term LEG usually means *legacy*. Replace it with the name that identifies your source system.
- 300100135839049 is an example value that identifies the fulfillment line. Replace it with the value that identifies your line.
- COMPLETED in the Status attribute is the status of the fulfillment line. Replace it with the status that your line is in.
- my_task_type is the name of a custom task type. Replace it with the name of the task type that you created.

You must provide a value for each of these attributes.

Order Fulfillment Response Service

Here's an example.

```
 <typ:processFulfillmentResponse>
 <typ:responseMessageHeader>

<com:IntegrationContextCode>ExportCompliance</com:IntegrationContextCode>
 <com:FulfillmentSystem>LEG</com:FulfillmentSystem>
 </typ:responseMessageHeader>
 <!--Zero or more repetitions:-->
 <typ:fulfillLineList>

<com:FulfillLineIdentifier>300100564138254</com:FulfillLineIdentifier>

<com:TaskInstanceStatusCode>COMPLETED_VAR</com:TaskInstanceStatusCode>
 </typ:fulfillLineList>
</typ:processFulfillmentResponse>
```

where

- LEG is an example name of a fulfillment system. Replace it with the name that identifies your fulfillment system.
- 300100564138254 is an example value that identifies the fulfillment line. Replace it with the value that identifies your line.
- COMPLETED_VAR is the status of a custom task type. Replace it with the status that your task type is in.

You must provide a value for each of these attributes.

Maintain Data Integrity

Order Management automatically maintains data integrity for your new task type. It:

- Makes sure the service data object includes data for each required attribute.
- Determines the transaction data to update as a result of the service call to your fulfillment system. This data resides in Order Management's transaction tables.

Maintaining data integrity makes sure the task type that you create displays correctly throughout the Order Management work area. Order Management also makes sure functionality works correctly with your new task type for:

- Status update
- Wait step
- Forward planning
- Jeopardy
- Hold processing
- Split processing
- Change management
- Error recovery

Include Your Charges

You must include values for the SourceChargeComponentId attribute and the HeaderCurrencyUnitPrice attribute in the Charge Components entity in your fulfillment response.

We recommend that you thoroughly test your entire fulfillment flow when you use a custom task to update an existing or add a new charge, including the fulfilment response. You must test these use cases:

- Freeze your pricing and freight charges, and then copy a sales order.
- Don't freeze your pricing and freight charges, and then copy a sales order.
- Revise a sales order.
- Create a return order that has the charge from the original order.

Do Other Optional Setups for Your New Task Type

Type of Setup	Description
Preprocessing	Add preprocessing logic. For example: • Set default values for data onto the outbound request. • Validate data on the outbound request.
Postprocessing	Add postprocessing logic. For example: • Set default values for data on the inbound request. • Validate data on the inbound request. • Interpret attributes or messages that your fulfillment system returns that might require Order Management to split processing across more than one orchestration process.
Change management	Use change management on an orchestration process step that references your new task type. Specify the attributes for the task type on the Manage Order Attributes That Identify Change page. Make sure your task type references the Update service and the Cancel service, and the connectors that these services require.
Hold code	To apply a hold to your new service, create hold codes for it. Hold All applies to your new service and to services that already exist.
Jeopardy threshold	To include a jeopardy score for your new task, set up a jeopardy threshold for it.
Processing constraint	Create a processing constraint that controls when to use your new task. For example, use a processing constraint that specifies the attributes that are required in the outbound request or on the inbound reply.
Data set used as part of outbound request	Consider these requests. • GetValidFLineData • The preprocessing service • Routing rules for the interface If you use one of them, then the template task uses a complete data set to communicate Order Management attributes.

Type of Setup	Description
	You can reduce the data set to make processing more efficient.
Register error messages	If your fulfillment system sends error messages to Order Management and you prefer to process and display them in Order Management, then you must register them.

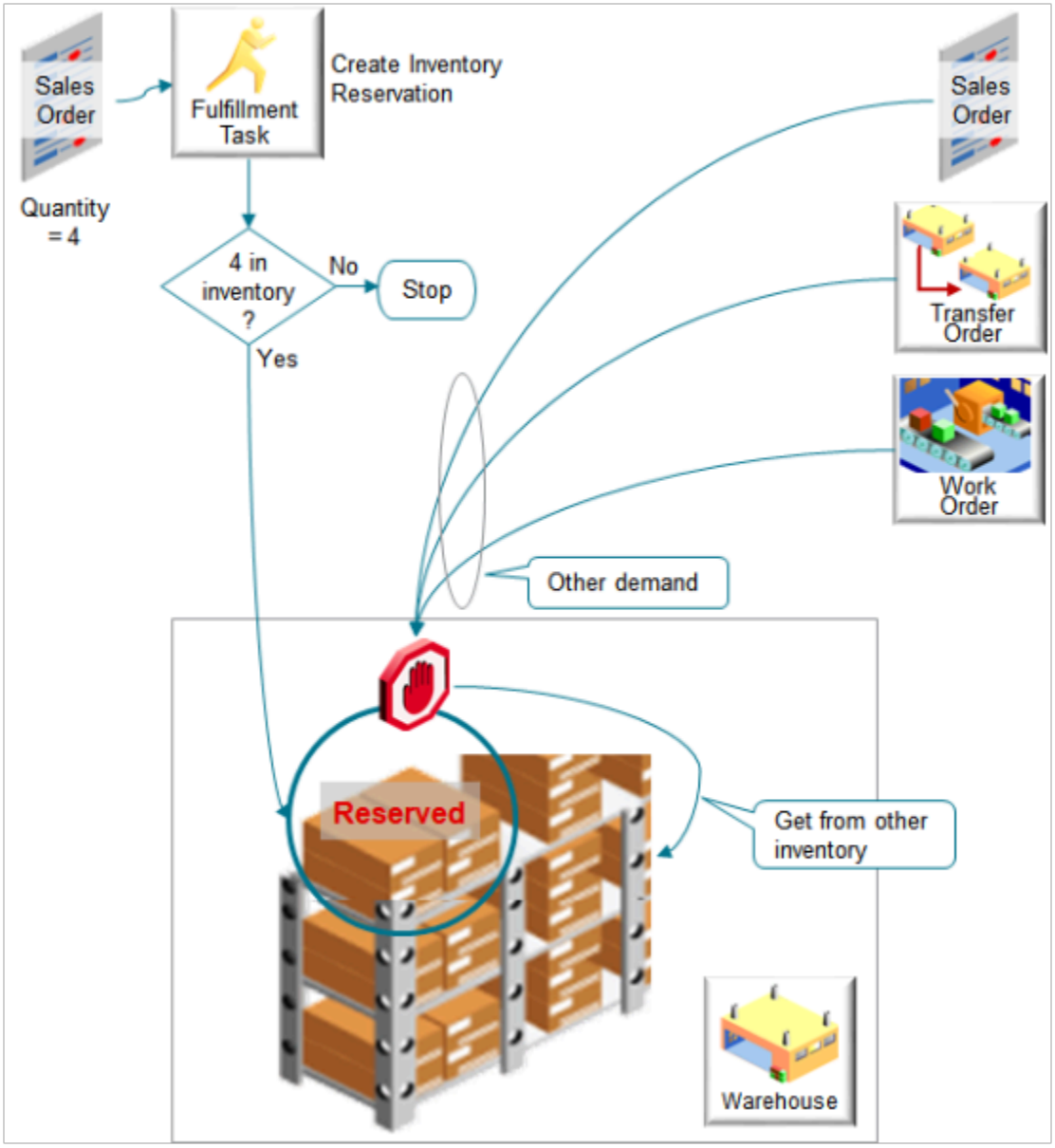
Related Topics

- Fulfillment Tasks
- Manage Connector Details Between Order Management and Your Fulfillment System
- Overview of Orchestration Processes

Guidelines for Reserving Inventory

Reserve inventory to make sure its available for your sales order when its time to pick your item in the warehouse.

Reserve supply for a sales order so no other demand can use the supply. Demand includes other fulfillment lines in your sales order, fulfillment lines in other sales orders, transfer orders, work orders, and so on.



You use the Create Inventory Reservations fulfillment task to reserve supply. Here's what it does.

- Uses your inventory system to reserve physical supply that currently exists in the warehouse.
- Doesn't reserve future supply, such as supply that's scheduled to be built.
- Reserves supply for each fulfillment line.
- Reserves the entire quantity on the fulfillment line.
    - You can't reserve only part of the quantity.
    - If the entire quantity isn't available in inventory, then the reservation task doesn't reserve any quantity.

      For example, if the quantity on fulfillment line x for item AS54888 is 4, and if there's only a quantity of 3 in inventory, then the reservation task doesn't reserve any quantity.

- Reserves only from inventory that's currently available. It doesn't reserve according to date. It reserves from current inventory even if your sales order includes a scheduled ship date that happens in the future.
- Automatically reserves each item where you set the Reservable attribute and the Back-to-Back Enabled attribute to Yes for the item in Product Information Management.
- Reserves items that aren't configured and items that are configured.
- Reserves all items in each assemble-to-order model, pick-to-order model, kit, or ship set together. It doesn't reserve individual items or only some items in each model, kit, or ship set.

  Assume you sell a desktop computer as a model, and the model includes the CPU, memory, hard drive, monitor, keyboard, and mouse. If inventory has everything in stock except the memory, then the task doesn't reserve anything.

You can't reserve.

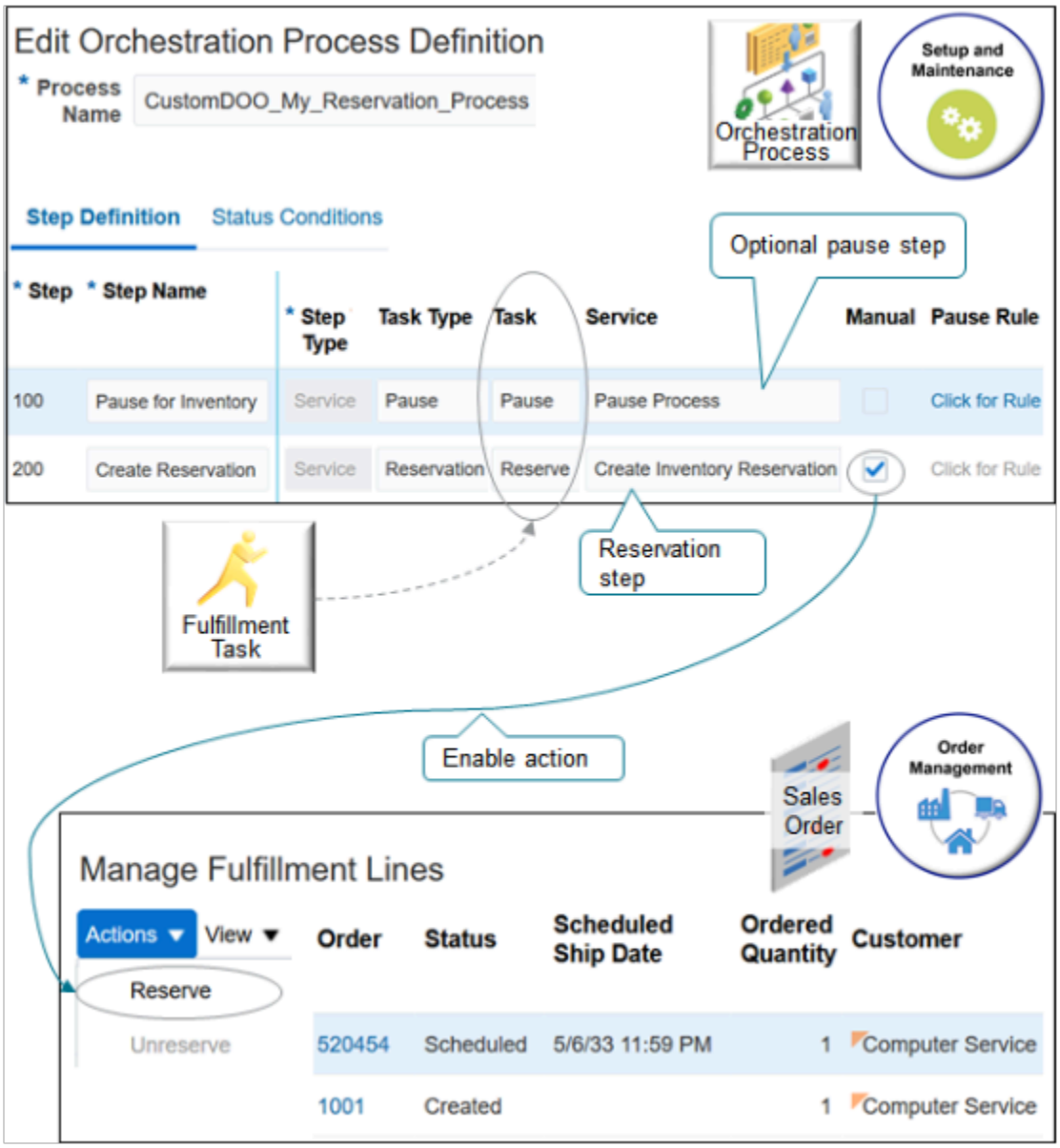
- Coverage, such as a warranty for a desktop computer
- Subscription, such as a magazine subscription
- Item that a lot or serial controls from a specific lot or a specific set of serial numbers
- From a specific subinventory.
- Through a web service.

Set Up Your Orchestration Process

The predefined DOO_OrderFulfillmentGenericProcess orchestration process comes already set up with a step that references the create reservation task. Use it to automatically reserve inventory.

- Reserve only according to quantity.
- Your flow must provide the item, quantity, unit of measure, and warehouse.
- Reserve the total requested quantity or it fails. You can't use it to reserve only part of the quantity.
- It doesn't consider the requested date.

Use DOO_OrderFulfillmentGenericProcess or create a copy of it, then add an optional pause step.



Note

- Use the pause step to pause the orchestration process until the shipment is almost ready to ship.
- The reservation step will provide a more accurate picture of supply that's available because its closer to the ship date.
- Waiting to reserve supply can also reduce the cost of holding inventory until its time to ship.

If you.

- **Enable the Manual Attribute on any step in the orchestration process**. Order Management enables the Reserve action on the Management Fulfillment Lines page and waits for the user to manually reserve the fulfillment line.
- **Don't enable it on any step.**. Order Management disables the Reserve action and the process doesn't wait for the user to manually reserve the fulfillment line.

The step details in the graphic might be a little hard to read. Here are the same details.

Step	Step Name	Step Type	Task Type	Task	Service	Manual
100	Pause for Inventory	Service	Pause	Pause	Pause Process	Doesn't contain a check mark
200	Create Reservation	Service	Reservation	Reserve	Create Inventory Reservation	Contains a check mark

Reserve Supply Automatically for Your Item

Try it.

1. Set up your item in Product Information Management.
    - Go to the Product Information Management work area.
    - On the Product Information Management page, search for your item, then open it for editing.
    - On the Edit Item page, click **Specifications**.
    - Click **Inventory**, then set the attribute.

Attribute	Value
Reservable	Yes

    - Click **Sales and Order Management**, then set the attribute.

Attribute	Value
Back-to-Back Enabled	Yes

2. Add a reserve step to your orchestration process.

Reserve Supply Manually for Your Item

Try it.

1. In the Order Management work area, click **Tasks > Manage Fulfillment Lines**, then search for your fulfillment line.
2. In the search results, in the General Tab, click the **link** next to Orchestration Process Number, such as 300100181483263.
3. On the Orchestration Process page, click **Fulfillment Lines**.
4. Click **Actions > Reserve**.

If you encounter an error, it could be that there's a hold on the line and you need to release it. For details, see *Schedule Fulfillment Lines Manually*.

Use Global Order Promising

Use Global Order Promising to reserve supply.

- Global Order Promising is an application that collects supply data from a planning system and promises to reserve the supply it collects.
- Supply isn't physical inventory. Its planned supply that the factory hasn't built yet.
- You can set up Global Order Promising to account for expected future supply and to split a fulfillment line.
- The reservation service in Order Management doesn't consider future supply. It only reserves from on-hand inventory.
- If you use Global Order Promising to reserve future supply, then don't use the reservation service in Order Management. Use one or the other, but not both at the same time.

There's a trade off between using Global Order Promising or the reservation service in Order Management. If there isn't enough supply to fulfill the line by the request date, and if you set the Allow Partial Shipments of Lines attribute on the line to Yes, then.

- The scheduling service in Global Order Promising will split the line into two lines. The second line will contain a future date.
- The reservation service in Order Management never splits the line, so it fails.

If you use Global Order Promising to schedule your fulfillment lines, then consider removing the reserve step from your orchestration process. If you must keep the reserve step, or if you don't use Global Order Promising, then.

- Enable the Manual property on the step so the user must click a button when close to the request date.
- Only send lines to the reservation step that you can fulfill with the inventory that you have on hand. Or use a pause step to pause lines that you can't fulfill from on-hand inventory. The pause delays the reservation until nearer the request date.

Another option is to remove the reserve step and let shipping handle reserving inventory.

Related Topics

- Schedule Fulfillment Lines Manually

Transfer Inventory Between Business Units to Fulfill Sales Orders

Transfer inventory between business units when there isn't enough inventory to fulfill a sales order for one of them.

Assume your company has two business units. You create sales order 75864 in the Los Angeles Operations business unit, then add an order line for the AS54888 Desktop Computer item with a Quantity of 100.

There isn't enough quantity in inventory to fulfill the item in Los Angeles, but the Denver Manufacturing business unit does have enough quantity. You can set up some rules that allow you to use Denver's inventory to fulfill Los Angeles' sales order.

Try it.

1. Use the Purchasing work area to create and approve a purchase request that Los Angeles can use to fulfill the quantity of 100.

2. Use the Manage Supply Execution Document Creation Rules task to set up a rule that uses the purchase order to create supply for Los Angeles. For details, see .*Set Up Rules That Create Supply*
3. Use the Manage Supply Order Enrichment Rules task to transform the transfer order into a sales order. For details, see *Guidelines for Setting Up Rules That Create Supply Orders*.

At run time, Denver reviews and approves the purchase order, Order Management uses the purchase order as input when it automatically creates a sales order for Denver, Denver ships the item, Order Management closes the transfer order, and Los Angeles receives the purchase order.

Related Topics

- Set Up Rules That Create Supply
- Guidelines for Setting Up Rules That Create Supply Orders

Don't Use Order Promising to Schedule Fulfillment

Your implementation might not need to use Order Promising to schedule fulfillment. You can set up Order Management so it doesn't.

Here are some examples.

- You import source orders that already have values for attributes that involve scheduling, such as ScheduleShipDate or Warehouse, or you have a transformation rule or order management extension that sets them, so you don't need Order Promising to calculate these attributes.
- You manually set the Warehouse attribute each time you create a sales order.
- You have a drop shipment where your supplier schedules fulfillment, or where you manually set the Supplier attribute and Supplier Site attribute for each sales order, so you don't need Order Promising to set them.
- You have your own downstream system that schedules, reserves, ships, and invoices your sales order.

Summary of the Setup

1. Verify the setup for the item.
2. Modify the orchestration process.
3. Set the default value for the scheduled ship date.
4. Test your set up.

Verify the Setup for the Item

1. Go to the Product Information Management work area.
2. On the Product Information Management page, search for your item, then open it for editing.
3. On the Edit Item page, click **Specifications**.
4. Click **Sales and Order Management**, then set the attribute.

Attribute	Value
Back-to-Back Enabled	Yes

Modify the Orchestration Process

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, search for the value.

Attribute	Value
Name	OrderFulfillmentGenericProcess

3. In the search results, click **Actions > Duplicate**.
4. On the Orchestration Process page, set the values, then click **Save**.

Attribute	Value
Process Name	CustomDOO_OrderFulfillmentGenericProcess
Process Display Name	CustomDOO_OrderFulfillmentGenericProcess

5. Leave the other values in the header area at their current values.
6. In the Process Details area, click **Status Conditions**.
7. In the Edit Status Rule Set column, click the **Edit Status Rule Set** button.
8. On the Edit Status Rule Set page, delete the rows, then click **Save and Close**.

Sequence	Status Value
100	Scheduled
2000	Manual Scheduling Required
2020	Unscheduled

9. On the Edit Orchestration Process Definition page, click **Orchestration Process Status Values**, then delete the rows.

Sequence	Status Value
100	Scheduled

Sequence	Status Value
1400	Manual Scheduling Required

10. Click **Step Definition**, delete the row, then click **Save**.

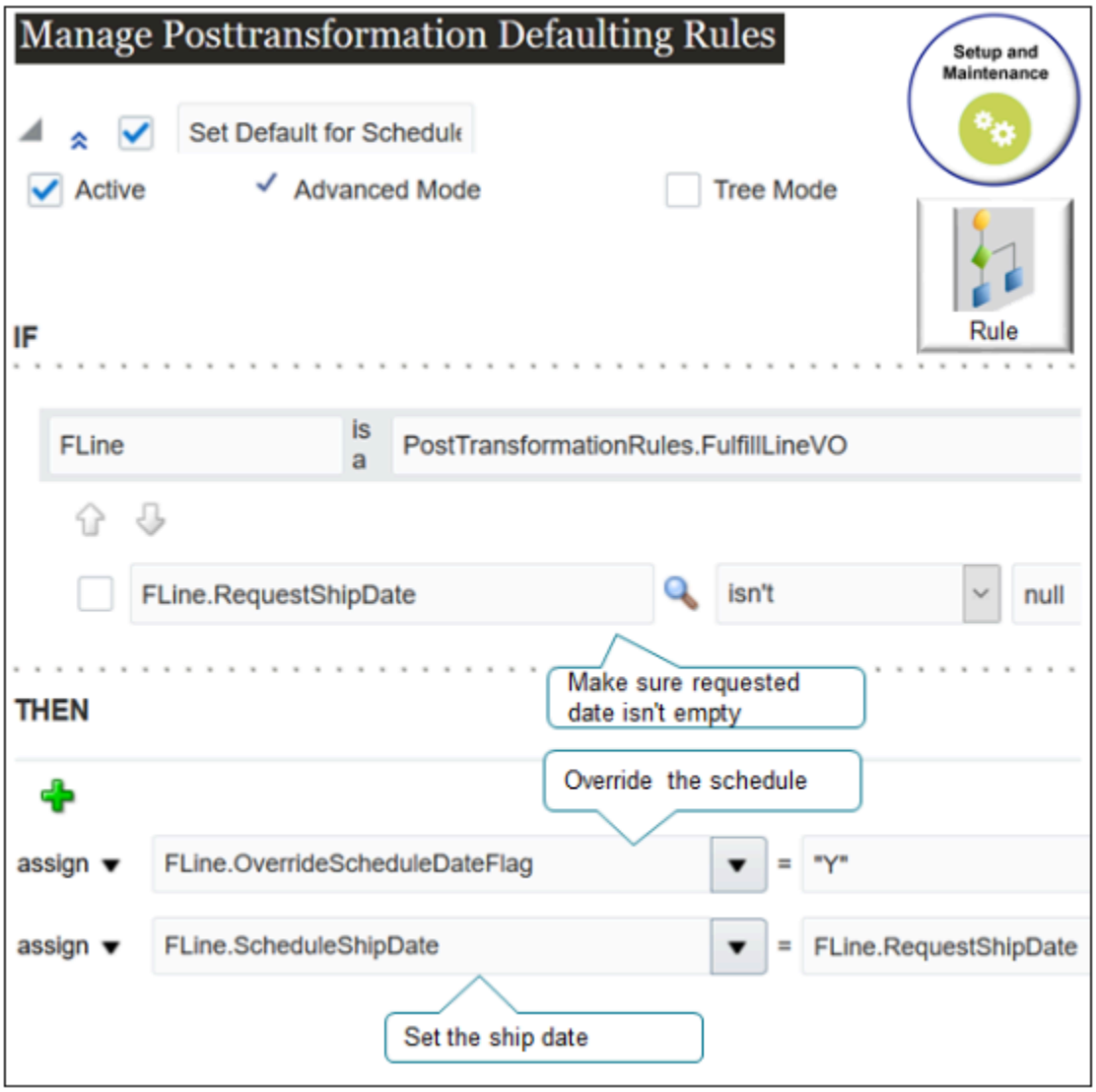
Step	Step Name
100	Schedule

11. At the top of the page, click **Actions > Validate**.
12. Click **Actions > Release**.
13. Click **Actions > Deploy**.

    For details, see *Deploy Orchestration Processes*.

Set the Default Value for the Scheduled Ship Date

You don't want Order Promising to set the Scheduled Ship Date, but you still need to set Scheduled Ship Date to a value. You create a rule that uses the Requested Ship Date to set the default value for Scheduled Ship Date.



Here are details about the If statement.

Code	Description
FLine	A variable that you declare into the PostTransformationRules dictionary.
PostTransformationRules	A dictionary that you use to access and store data for your posttransformation rule.
FulfillLineVO	A view object that contains attributes. VO is an abbreviation for view object.
If FLine is a PostTransformationRules. FulfillLineVO	Assign the name Fline to the object.
FLine.RequestShipDate	Store the value of the RequestShipDate attribute in the FLine variable.

Code	Description
FLine.RequestShipDate isn't null	Proceed to the Then statement only if RequestShipDate contains a value. If RequestShipDate doesn't contain a value, then you can't use it to set the value for ScheduleShipDate.

Here are details about the Then statement.

Code	Description
FLine.OverrideScheduleDateFlag = "Y"	Override the schedule that Global Order Promising calculates.
FLine.ScheduleShipDate = FLine.RequestShipDate	Set the value of the ScheduleShipDate attribute in the FLine variable to the value that the RequestShipDate attribute contains in the FLine variable.

Try it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional area: Orders
    - Task: Manage Posttransformation Defaulting Rules
2. On the Manage Posttransformation Defaulting Rules page, click **Add**, then set the value.

Attribute	Value
Name	Set Default for Scheduled Ship Date

3. Click **Expand**, click **Properties**, set the values, then click **Save**.

Attribute	Value
Effective Date	Always
Advanced Mode	Contains a check mark

4. In the If area, create an IF statement.

    ```
    If FLine is a PostTransformationRules.FulfillLineVO
    ```

5. In the area below the statement you just created, click **Left Value**.
6. In the Condition Browser dialog, expand **FLine**, click **RequestShipDate**, then click **OK**.
7. Set Is, to Isn't, then enter `null` after Isn't.

    Your condition should like `FLine.RequestShipDate isn't null`.

Create a Then Statement

Create a Then Statement that sets the value of the overrideScheduleDateFlag attribute to Y.

1. In the Then area, click **Add Action > Assign**.
2. Click **Select a Target**, then click **FLine.OverrideScheduleDateFlag**.
3. In the window to the right of the equal sign, enter `"Y"`.

   Make sure you enclose the Y with double quotation marks (" ").

Create Another Then Statement

Create a Then Statement that sets the value of the ScheduleShipDate attribute to the value of the RequestShipDate on the fulfillment line.

1. In the Then area, click **Add Action > Assign**.
2. Click **Select a Target**, then click **FLine.ScheduleShipDate**.
3. In the window to the right of the equal sign, enter `FLine.RequestShipDate`.

For details, see *Overview of Using Business Rules With Order Management*.

Test Your Set Up

1. Go to the Order Management work area and create a sales order.

   Make sure you set these attributes on the order line.

   - Warehouse
   - Shipping Method

   If you're importing, make sure these attributes contain a value in your import payload.
2. Click **Submit**.

Drop Ship

If you want to skip Global Order Promising in a drop shipment, then set the value of the ship date or arrival date, depending on the value of the Buyer Managed Transportation option on the purchase order.

Buyer Managed Transportation	Date You Must Set
Enabled on the purchase order	Requested Ship Date on the fulfillment line
Not enabled on the purchase order	Requested Arrival Date on the fulfillment line

To determine whether Buyer Managed Transportation is enabled.

1. Make sure you have the privileges that you need to manage purchase orders.
2. Go to the Purchase Orders work area.
3. Search for your purchase order, then open it.
4. On the Terms tab, examine the Buyer Managed Transportation attribute.

Related Topics

- Deploy Orchestration Processes
- Overview of Using Business Rules With Order Management

7 Use Business Rules

Overview

Overview of Using Business Rules With Order Management

Set up a business rule in Oracle Order Management to implement a dynamic decision at run time that automates a company policy, does a calculation, or does some processing.

A business rule is a statement that describes how to implement a business policy or make a business decision. It can implement logic.

- Enforce a spending policy.
- Constrain a process so it meets a regulatory requirement.
- Compute a discount or premium.
- Provide an offer according to customer value.

Here are some business requirements you can meet with a business rule.

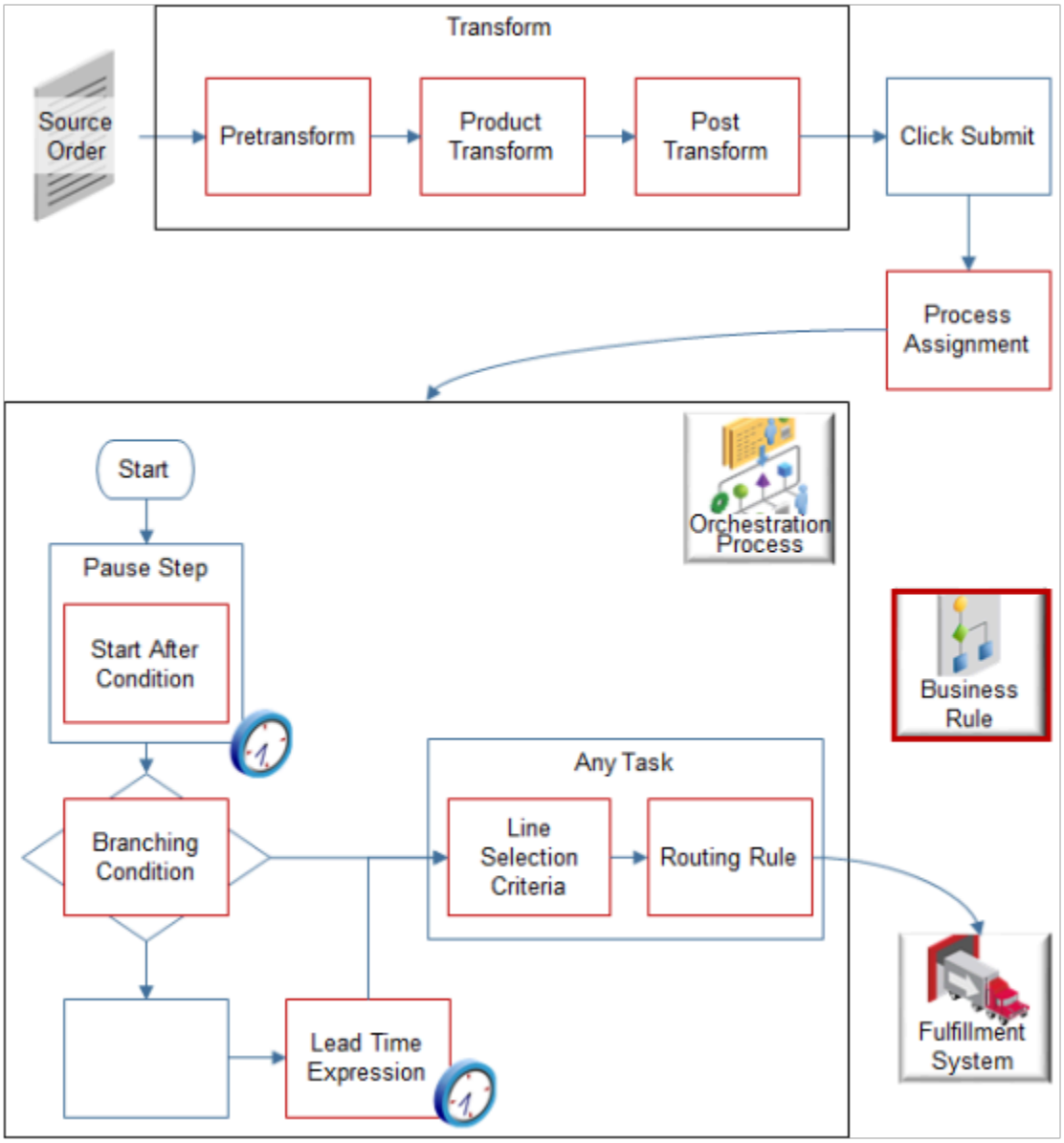
- Set a default value for shipment priority
- If quantity is more than 10, then add free items to the sales order.
- If customer is Computer Service and Rentals, then use orchestration process y to fulfill the sales order.
- If destination is Japan, then route shipment through Pacific Northwest Warehouse.

A business rule keeps rule logic separate from the underlying application code, which allows a business analyst to modify rule logic without using programming code and without interrupting your business process.

Here's an example of a business rule.

- If the sales order is valued at $50,000 or more, then make sure a representative calls the customer before sending an invoice.

Here are the types of business rules you can use and where you can use them.



Note

Type of Business Rule	Description
Transform a sales order. • Pretransformation • Product Transformation • Posttransformation	Order Management transforms each source order that you create in Order Management or that it receives from a source system so it can optimize order fulfillment. You can write a rule to: • Populate order attributes before transformation. • Transform a source order. • Populate order attributes after transformation. For example: • Populate an attribute on a fulfillment line. If the item is a widget, then populate the Request Date attribute on the fulfillment line.

Type of Business Rule	Description
	• Convert a measurement. If the item is a widget, then convert the value in the Size attribute from centimeters to inches in the fulfillment line. • Create fulfillment lines from one item. For example, if the item is a laptop that includes a docking station, then transform the item into one fulfillment line for the laptop and another fulfillment line for the docking station. For details, see *Transformation Rules*.
Process Assignment	Assign the orchestration process that Order Management runs to process fulfillment lines. For example: • If the ordered quantity is large, then assign the sales order to an orchestration process that optimizes scheduling and delivery for large orders. • If the customer is Important, then assign the sales order to an orchestration process that expedites delivery. • If the ship-to address in the order header resides outside of your country, then assign the sales order to an orchestration process that handles international fulfillment, such as checking for trade compliance. For details, see *Assign Orchestration Processes*.
Process a sales order. • Start After Condition • Branching Condition • Lead Time Expression • Line Selection Criteria Process a change order. • Compensation Pattern • Cost of Change	Set up a rule that affects processing, such as branch in an orchestration process, do a complex calculation that determines planning lead time, or manage a change that happens to the sales order. For example: • If an invoice exceeds $100,000, then make sure a representative phones the customer. • If the status of the Create Shipment orchestration process step is Shipped, then send a notification to your customer that the sales order is on its way. • If the sales order includes a shippable item, such as a laptop, and an item that isn't shippable, such as a warranty for the laptop, then make sure the orchestration process doesn't attempt to ship the warranty.
Routing Rule	Set up a rule that routes a fulfillment request to a fulfillment system according to an attribute on the sales order, fulfillment line, or orchestration process. For example: • If product type is Goods, and if task type is Invoice, then route the request to fulfillment system ABCInvoicingSystem. • If item is 2TX Server, then route the service request to fulfillment system Big Server. For details, see *Overview of Connecting Order Management to Your Fulfillment System*.

Get background details about how to create an Oracle Business Rule. For details, see *Designing Business Rules with Oracle Business Process Management*.

Use Visual Information Builder to Create Rules

Prior to Update 12, you use Oracle Business Rules to create rules in Order Management. Starting with Update 12, you can use Visual Information Builder to create some types of rules, which is a rule editor that supports a simplified drag-

and-drop interface. It helps you visualize data, visualize your business processes, implement your business logic, and implement your business rule sets.

Starting with Update 13B, we strongly recommend that you use only Visual Information Builder for routing, pretransformation, and assignment rules.

Here are pages you use to access the editors.

Editor for Oracle Business Rules	Editor for Visual Information Builder
Manage External Interface Routing Rules	Manage External Integration Routing Rules for Sales Orders
Manage Pretransformation Defaulting Rules	Manage Pretransformation Rules for Sales Orders
Manage Orchestration Process Assignment Rules	Manage Process Assignment Rules for Sales Orders

Examples of Creating Business Rules

Visual Information Builder

See:

- *Manage Routing Rules*
- *Manage Pretransformation Rules*
- *Route Requests from Order Management to Fulfillment Systems*
- *Route Requests from Order Management to Fulfillment Systems Without Cross-References*
- *Create Cross-References in Order Management*
- *Integrate Order Management Without Cross-Referencing Customer Attributes*

Oracle Business Rules

See:

Page	Details
Manage Orchestration Process Definitions	Get details about the rules that you set up on an orchestration process. • *Add Lead-Times for Orchestration Process Steps* • *Select Fulfillment Lines for Orchestration Process Steps* • *Measure the Cost of Change* • *Compensate Sales Orders That Change* • *Add Branches to Orchestration Processes* Get details about the rules that pause an orchestration process. • *Pause Orchestration Processes Until Events Happen* • *Pause Orchestration Processes Until Time Elapses*

Page	Details
	• *Pause Orchestration Processes Until Dependencies Resolve* Get details about the rules that control status. • *Add Status Conditions to Orchestration Processes* • *Add Status Conditions to Fulfillment Lines*
Transformation Rules	Get details about transformation rules. • *Create Transformation Rules* • *Create Advanced Transformation Rules*

Learn how to use business rules with extensible flexfields. For details, see *Overview of Setting Up Extensible Flexfields in Order Management*.

Examples That Include Orchestration Process Attributes

Orchestration Process Attribute	Details
Cost of Change	*Measure the Cost of Change*
Compensation Pattern	*Manage Order Attributes That Identify Change* *Compensate Sales Orders That Change*
Lead-Time Expression	*Add Lead-Times for Orchestration Process Steps*
Line Selection Criteria	*Select Fulfillment Lines for Orchestration Process Steps*
Branching Condition	*Add Branches to Orchestration Processes*
Start-After Condition	*Pause Orchestration Processes Until Events Happen* *Pause Orchestration Processes Until Time Elapses* *Pause Orchestration Processes Until Dependencies Resolve*

You use the Manage Orchestration Process Definitions page to set these attributes. For details, see *Guidelines for Setting Up Orchestration Process Steps*.

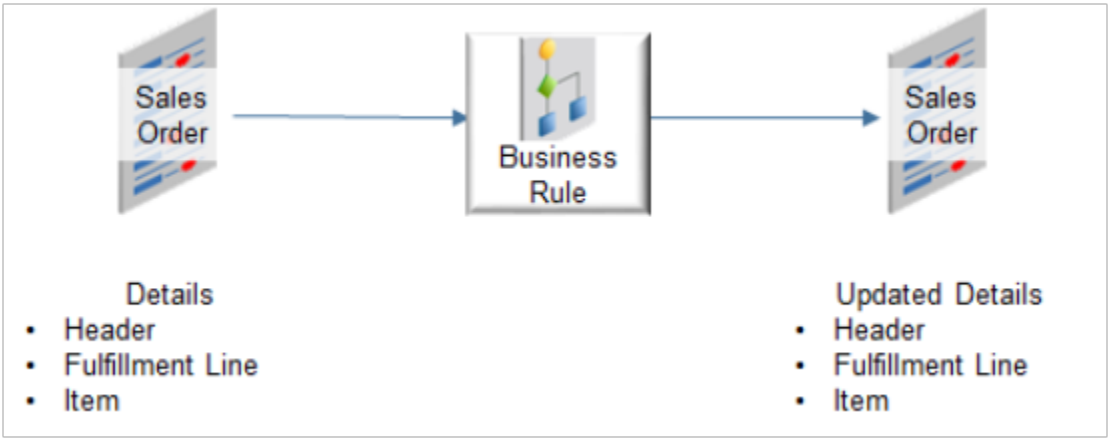
Related Topics

- How Business Rules Work
- Manage Errors and Conflicts in Business Rules
- Functions You Can Use in Business Rules
- Use the Business Rules Editor
- Use Business Rules in Orchestration Processes

How Business Rules Work

Write a business rule that uses conditional logic to process a business event.

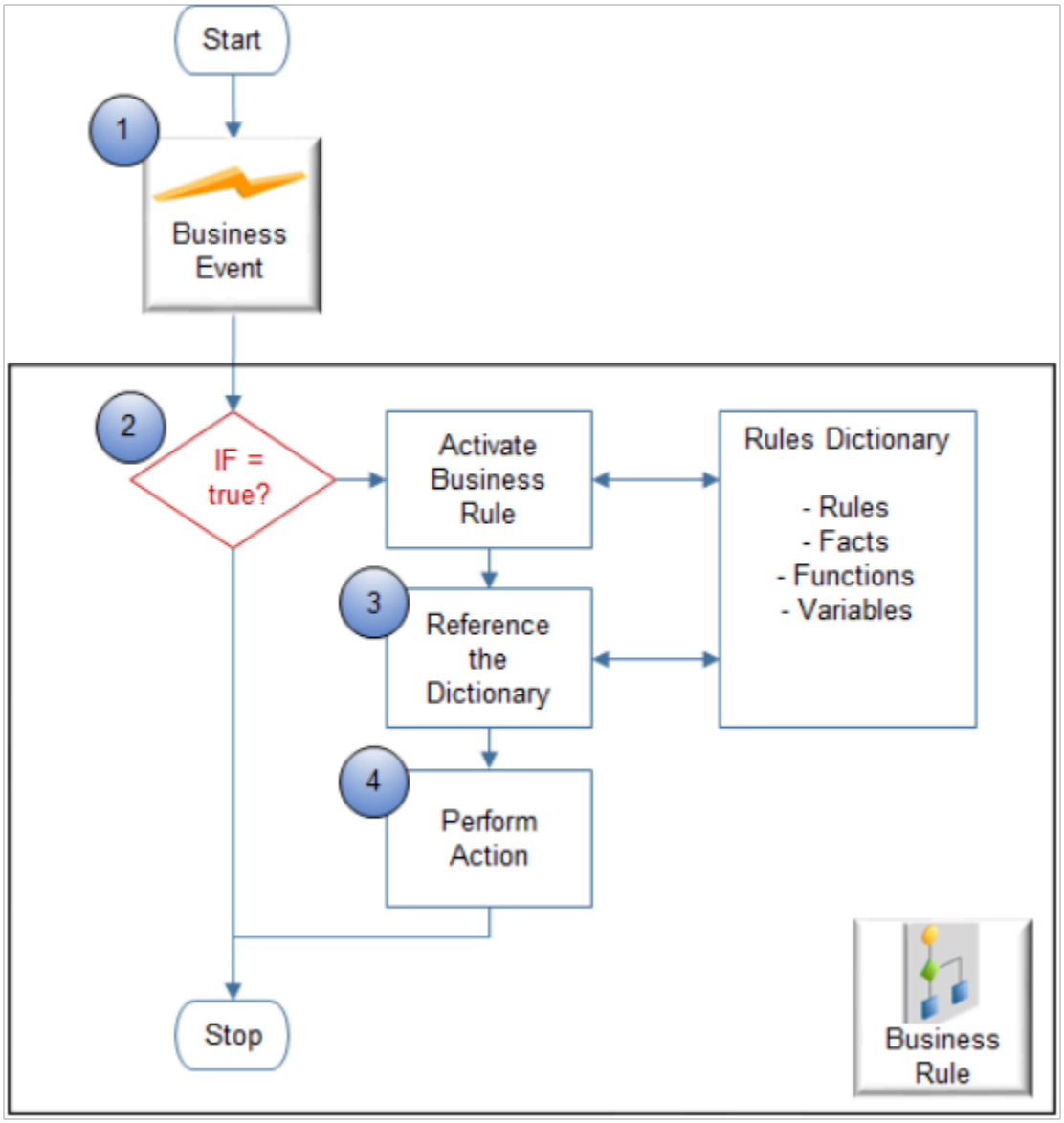
A business rule uses the sales order as input, uses functions to get details or to do an action on a fact, then provides a result that Order Management uses to update sales order details.



Assume you must implement a rule.

- If the item is a Green Server, then use priority shipping.

Here's the flow.



Note

1. An event happens that causes Order Management to run the rule. The event depends on the context where you write the rule. For example, if you set up a rule.
    - **On a step in an orchestration process**. The rule runs when the process reaches the step.
    - **On a routing rule**. The rule runs when a task references the rule.

2. If the If statement is true according to the facts, then Order Management activates the rule.

   For example:

   - If the item is a Green Server

     where

     - Item is a fact.
     - Green Server is a fact.

3. The rule references the dictionary to get the objects it needs at run time.

   For example, you create variables and declare objects when you set up the rule. You store these objects in the dictionary.

4. The rule does the action.

   The action is a Then statement. For example:

   - Use priority shipping.

     where

     - **priority shipping** is a fact

For background details about business rules, see *User's Guide for Oracle Business Rules*.

Parts of a Business Rule

A business rule contains an If statement and a Then statement.

For example:

- If the sales order is valued at $50,000.00 or more, then make sure a representative calls the customer before sending an invoice.

Here's a rule in Oracle Business Rules editor.

Note

1. **Left Value of the If statement**. Specifies the object the rule compares in the If statement.
   In this example, Left Value contains a string.
   `DooSeededOrchestrationRules.DOOFLine`
   where
   - DooSeededOrchestrationRules is the name of a dictionary. A **dictionary** is an XML file that contains rules organized in rule sets, facts, functions, variables, bucket sets, links, the data model, and so on.
   - DOOFLine is an object in the dictionary.
     - DOO is an abbreviation for distributed order orchestration, which is a name Order Management used in earlier releases. It means the same as order orchestration.
     - extendedAmount is a fulfillment line attribute. It contains the total value of the sales order.
     - FLine is an alias for fulfillment line.
     - DOOFLine contains the data model for the fulfillment line in order orchestration. For example, DOOFLine contains the fulfillment line attributes, such as orderedQty, customerPONumber, creationDate, and so on. You can reference these attributes and use their values in your rule.
2. **Right Value of the If statement**.
   In this example, Right Value contains:
   `line.extendedAmount more than 50000`
   It specifies the value the rule uses to determine whether the If statement is true. The example contains a literal value of 50000. You can also reference an object from the dictionary. If you reference an object, then the rule uses the value that the object contains at run time to make the comparison.
3. **Action**. Specifies how to add your target to the dictionary. You can specify a variety of actions. Assign means to add assign a fact to the dictionary.
   A **fact** is transactional data that the rule uses, such as the items on a sales order.
   - An object stores this data. You reference the object when you create a rule.
   - Order Management provides a hierarchy of facts according to the transactional data for the sales order in each dictionary.
   - Each object instance corresponds to a single fact.
   - You must assert each fact before you can use it in a rule. Assert makes the fact available so you can reference it elsewhere in the rule, such as in the properties of the Target.
4. **Target**. Specifies the object that the rule modifies when the If statement evaluates to true.
   `DooSeededOrchestrationRules.Result.Result`
   In this example, if extendedAmount is more than 50000, then the rule will assert the Result object as a fact into the DooSeededOrchestrationRules dictionary.
5. **Edit Properties**. Specifies the properties of the target. If the rule evaluates the If statement to true, then the rule modifies the values of the properties of the target according to how set up these values. Here's the code that this example uses for the properties.
   `DooSeededOrchestrationRules.Boolean.TRUE`
   where

- resultObj is a property of the Result fact that the target specifies.
- Boolean is a variable in the DooSeededOrchestrationRules dictionary.
- TRUE specifies to set the Boolean variable to TRUE.

The Then statement sets the resultObj property of the Result fact to TRUE. This example is a branching condition, so the orchestration process examines the value of resultObj at run time. If it contains TRUE, then the process runs the branch that makes sure a representative phones the customer.

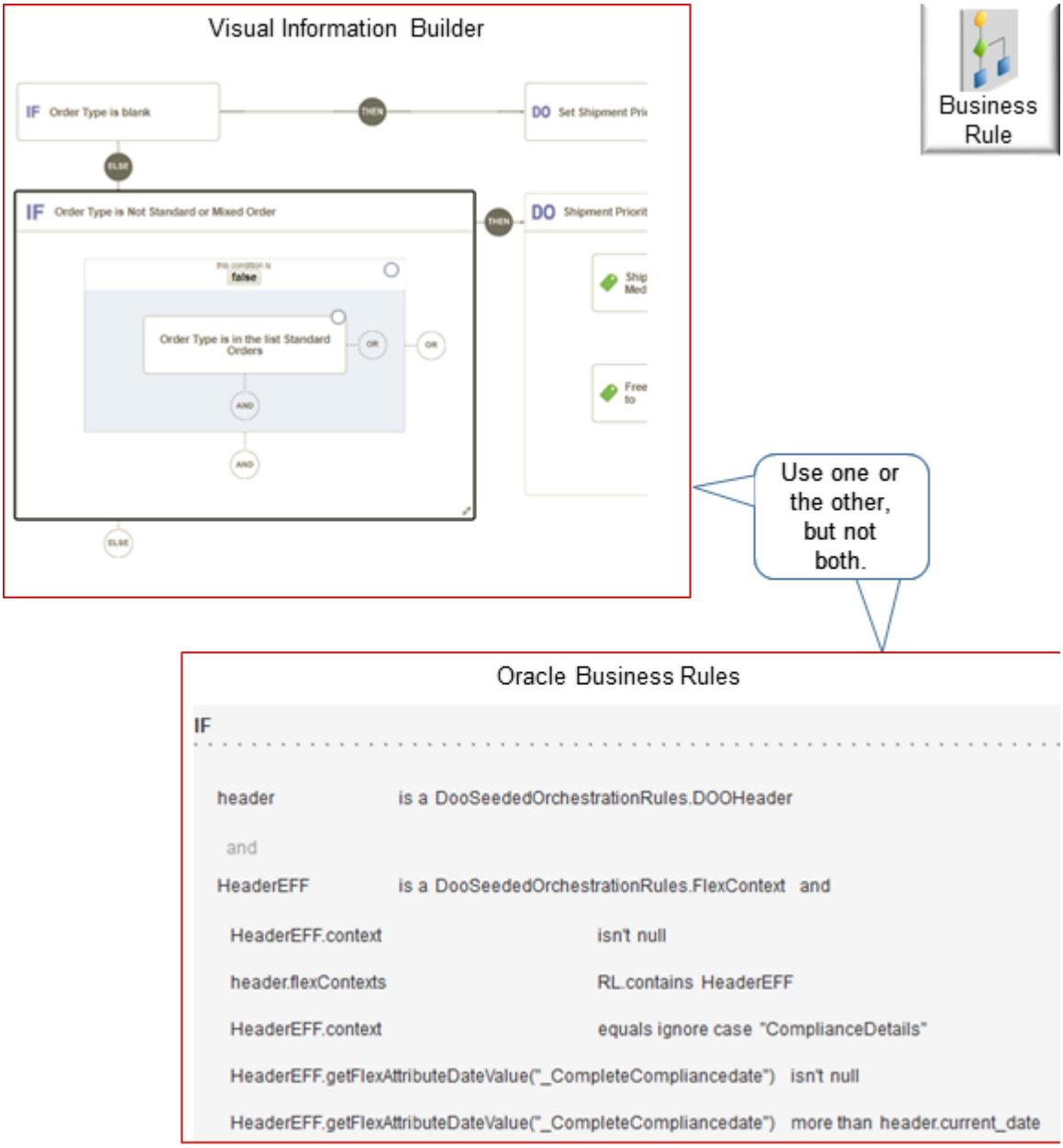
Guidelines

Use Tools and Environments to Create Business Rules

Use these guidelines to help you create a business rule.

Use Visual Information Builder or Oracle Business Rules

Use Visual Information Builder to create all your rules, or use Oracle Business Rules to create all your rules. Don't use Visual Information Builder to create some rules, and then use Oracle Business Rules to create other rules.



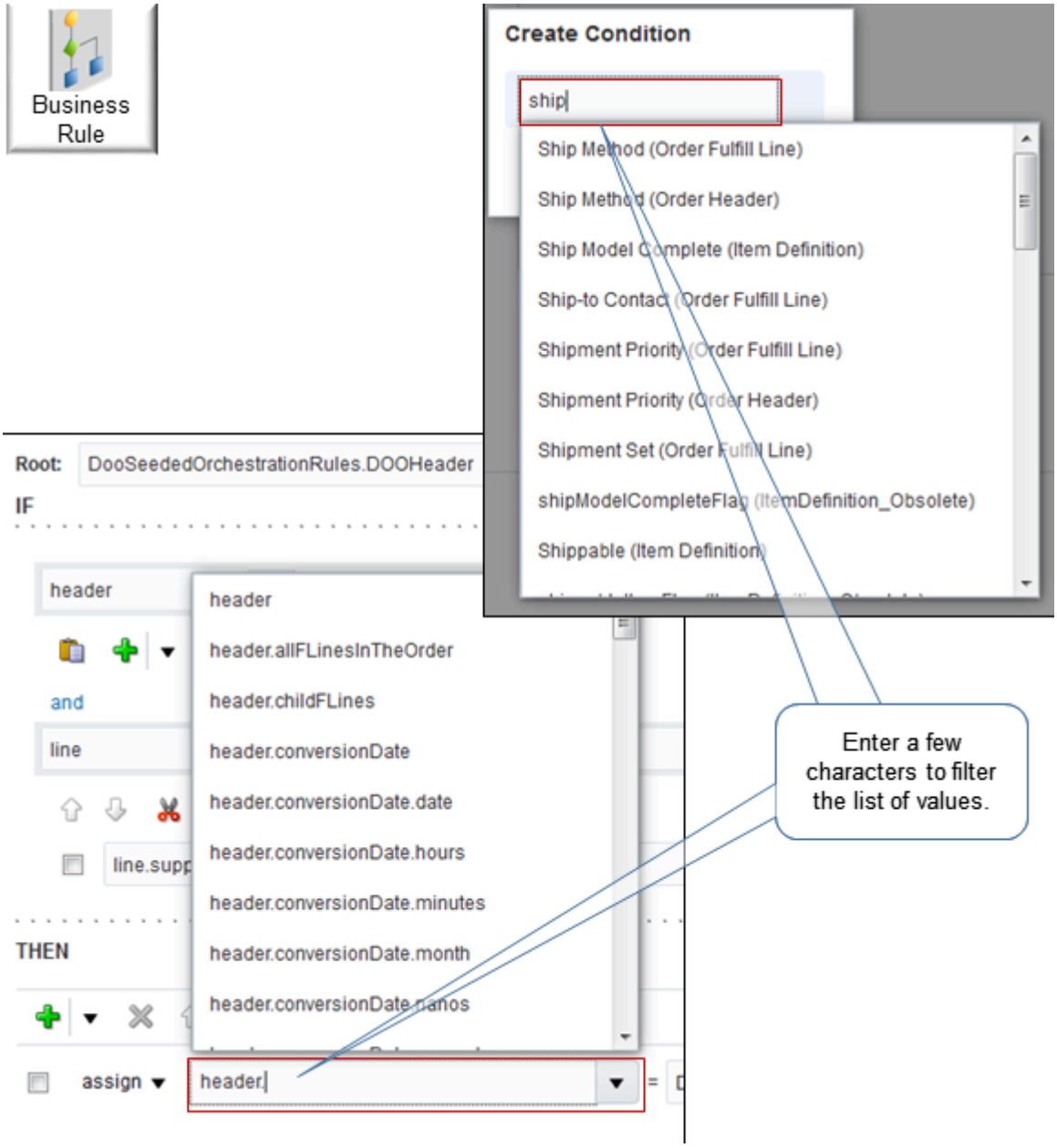
Note

- Create all your rules in the same instance of Order Management. Don't create some rules in one instance, and then some other rules in another instance or environment.
- Some rule dictionaries are available in jDeveloper. However, don't use jDeveloper because problems might happen during patching, instantiating objects, and migrating. Instead, use the Setup and Maintenance work area to set up your rules.

  Use the work area to migrate setup data from one environment to another environment. You can create rules for orchestration processes only through the Manage Orchestration Process Definitions page in the Setup and Maintenance work area. Migration migrates rules that you create when you set up an orchestration process.

Use Auto Fill

Use automatic fill to display the list of values that are available according to the rule structure that you set up. Automatic fill helps you to simplify creating the rule because you don't need to research what attributes and facts are available, and you don't need to research whether you can use an attribute or fact in the logic that you set up. Automatic fill refines the search result each time you enter another character.



Use Automatic Fill in Oracle Business Rules

We strongly recommend that you use automatic fill or the drop down list and dialog in Oracle Business Rules when you set the values. Don't enter a value directly in a field unless you're entering a literal string. Oracle Business Rules dynamically modifies the values that are available in fields depending on the values that you set while you're creating

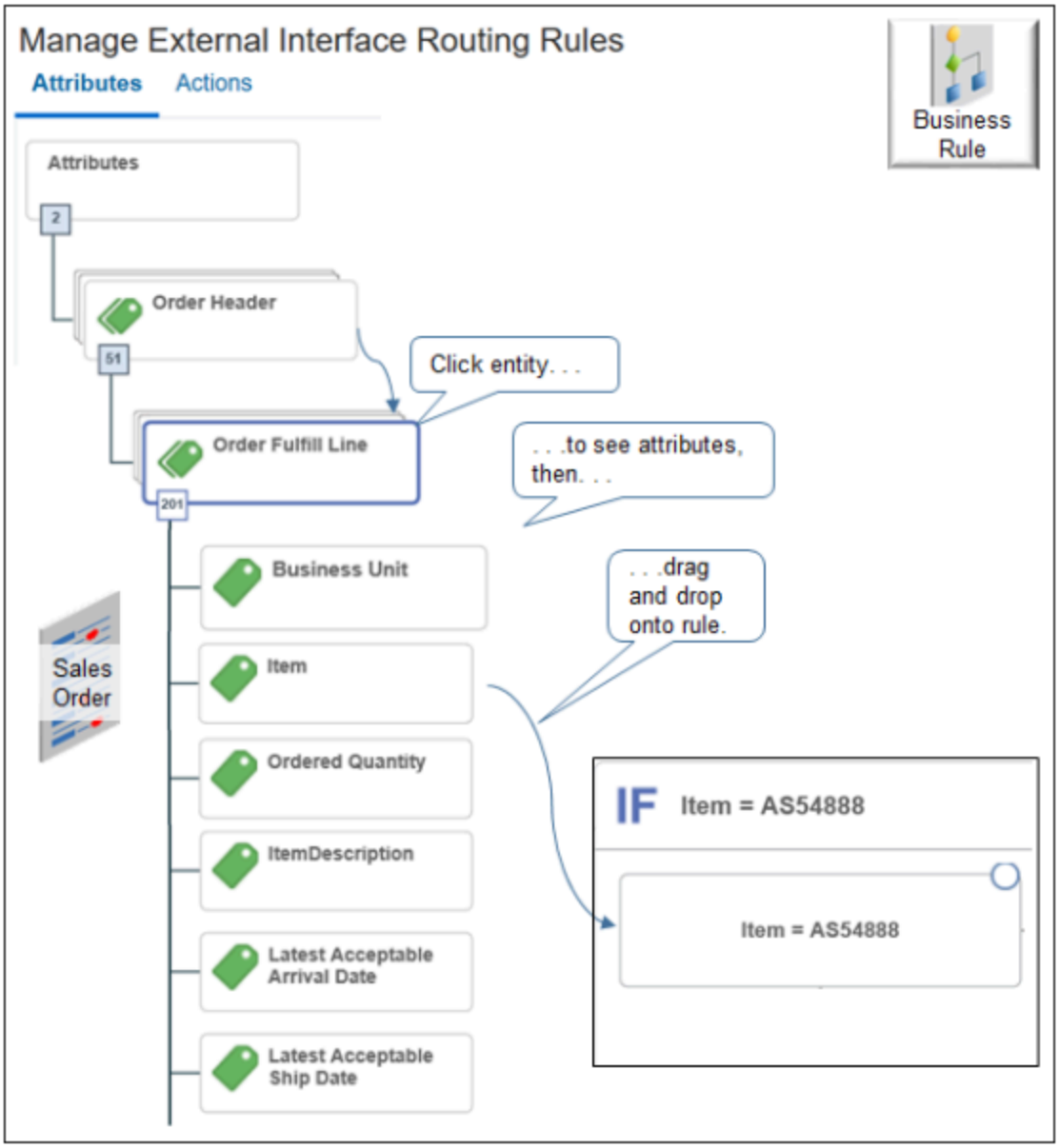
the rule. If you enter a value directly in an field, its possible that the rule will fail validation even if the value that you enter is exactly the same value that the rules editor populates when you use automatic fill or the dialog.

If you declare a variable in one field, but don't see it available in another field, then click Save in the rule builder dialog.

If you enter a literal string in a field and get a validation error when you click Validate, then remove all content from the field, click Expression Value, enter you value in the Condition Browser, then click OK.

Use the Tree in Visual Information Builder

Use the tree in Visual Information Builder to find the attributes that you can use in your rule, then add them.



The tree displays the hierarchy of the objects that you can add to the rule. For example, the Manage External Interface Routing Rules page displays the hierarchy of attributes that you can add.

- A sales order is an object in Order Management.

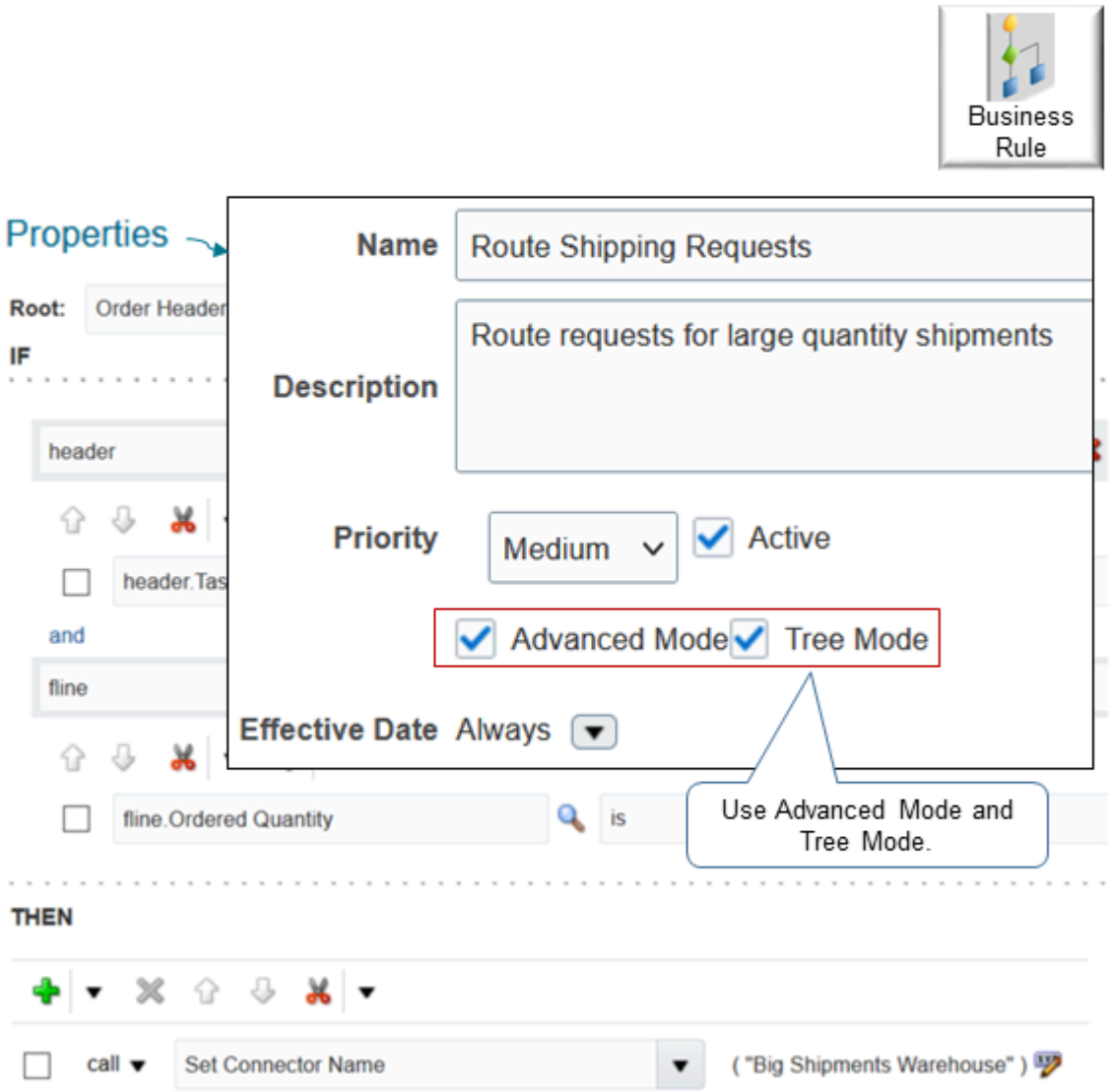
- A sales order includes entities, such as.
    - Order Header
    - Order Fulfill Line
- Each entity includes attributes. For example, here are some attributes in the Order Fulfill Line entity.
    - Business Unit
    - Item
    - Ordered Quantity
    - And many others
- To add an attribute to your rule, click the attribute, drag, then drop it onto the rule.

  For example, drag the Item attribute onto the If statement of your rule.

You can use the tree to set up a wide range of conditional logic that meets your business requirements. If you prefer not to use the tree, you can search directly in the If statement or Then statement for the entity, attribute, values, and so on.

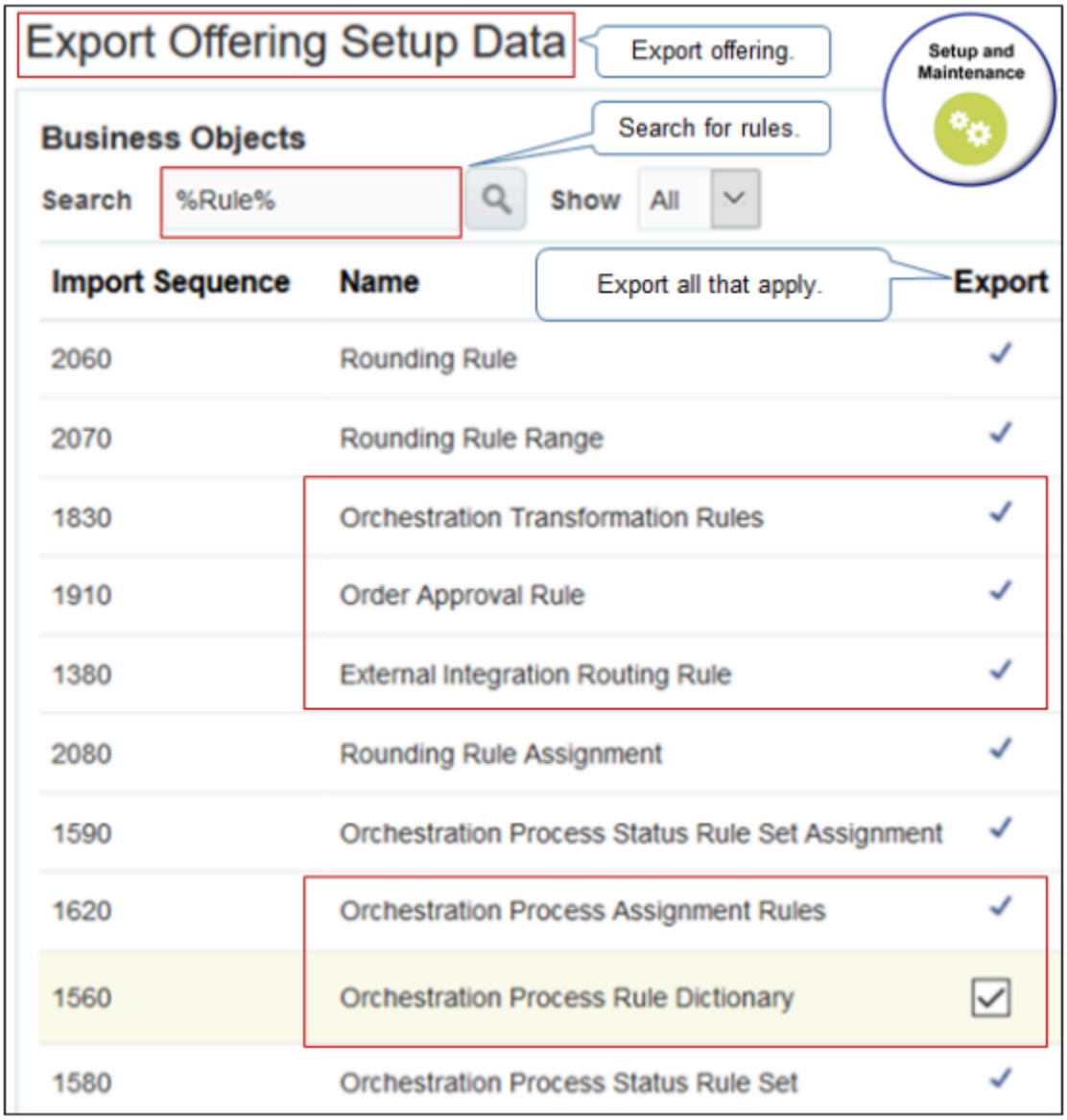
Use Advanced Mode and Tree Mode in Oracle Business Rules

Use Advanced Mode and Tree Mode to help simplify creating and managing your rule. These modes help structure your rule to make sure it contains the correct logic structure, and to filter the choices that are available to you to help make sure the rule references objects correctly according to structure.



Migrate Between Environments

Specify types of rules to migrate.



Note

- In the Setup and Maintenance work area, click **Actions > Export > Create New Export**.
- Export the business objects that you need to run your business rule. For example:
    - Transformation Rules
    - Approval Rules
    - Routing Rules
    - Assignment Rules
    - Orchestration Process Rule Dictionary

For details, see.

- Export Offering Setup

- *Migrate Business Rules in Order Management*

Related Topics

- Migrate Business Rules in Order Management

Apply Logic in Business Rules

Use guidelines to help you create a business rule.

Create Rules That Don't Conflict

- Make sure your rules won't cause a conflict at run time.

  For example, assume you write rule x that assigns a fulfillment line with item type of server to one process, and you write rule y that assigns server item ABC to another process. A conflict might happen. You must revise rule x so it doesn't specify server item ABC. Oracle Business Rules provide details about potential conflicts after you save the rule.
- If your rule compares one data object to another data object of the same type, then use advanced mode and create two variables, one variable to represent each data object.

  You must specify that variable one is not the same as variable two. Set up this test.

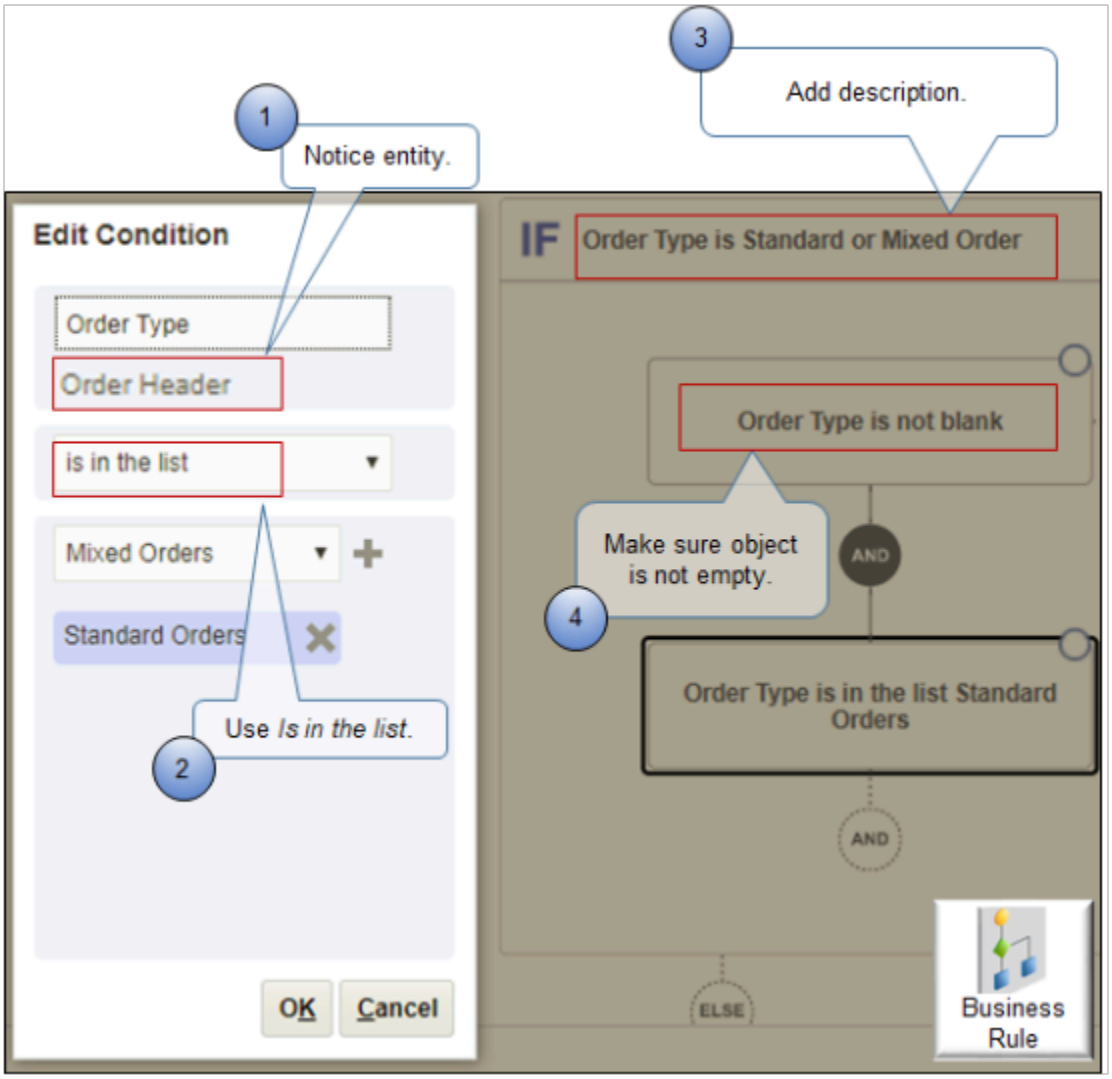
  ```
  variable1 is not variable2
  ```

  If you don't set up this test, then Oracle Business Rules will use variable1 and variable 2 to represent the same object instance when it determines the rule to apply.
- Make sure each rule evaluates to a single result. For example, if you write two rules for a fulfillment task, then make sure only one rule evaluates to true. Assume you write two rules.

  ```
  If Task Type = Shipment return Shipment
  If Task Type = Shipment and Customer = GOLD return Third Party Shipment
  ```

  These rules look different but they each evaluate to true when Task Type equals Shipment and Customer equals GOLD. To avoid this problem, write If, Then, Else rules in Visual Information Builder to make sure your rules are mutually exclusive.

Use Rule Logic in Visual Information Builder



Note

1. **Notice the entity**. You enter the attribute, such as Order Type, and then the entity that contains the attribute, such as Order Header, displays immediately under the attribute.

   Most attributes on the order header are different from attributes on the order line. For example, the order header includes the Order Type attribute but the order line doesn't. However, the order header and the order line contain similar or identical attributes. For example, the order header and the order line each include attribute Shipping Instruction. If your rule must reference Shipping Instruction on the order header, then make sure you use the order header entity.
2. **Is in the List**. Use Is in the List to examine the list of values that the attribute references instead of writing an OR condition. For example, the list of values for Order Type includes Mixed Orders and Standard Orders. If you don't use Is in the List, then it might be necessary to write a series of OR conditions. For example:

   ```
   If Order Type equals Mixed Orders, or If Order Type equals Standard Orders
   ```
3. **Add the description**. Description for the IF statement is optional. However, you can use it to quickly scan your rule logic without having to drill down into the rule to determine the purpose of the statement. You can also add a description for the Then clause.
4. **Make sure the object isn't empty**.

Note: If your rule depends on a value, and if the rule references an attribute that doesn't contain a value, then it might fail with unpredictable results, such as a error. For example:

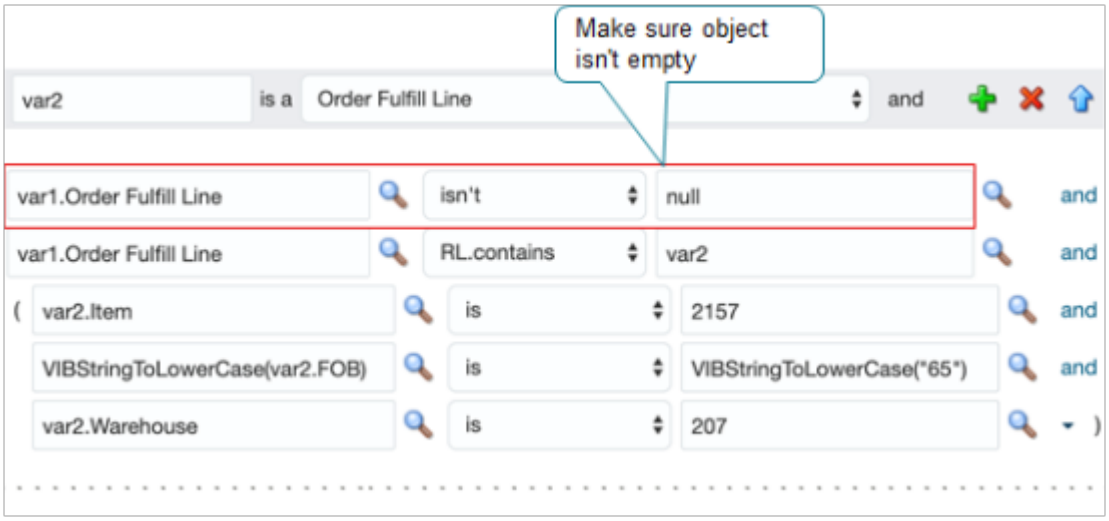
```
Manage Process Assignment Rules for Sales OrdersView row with key oracle.jbo.Key[null ] is not
 found in __DimensionVO__For__Reference__For_DCL__
```

or

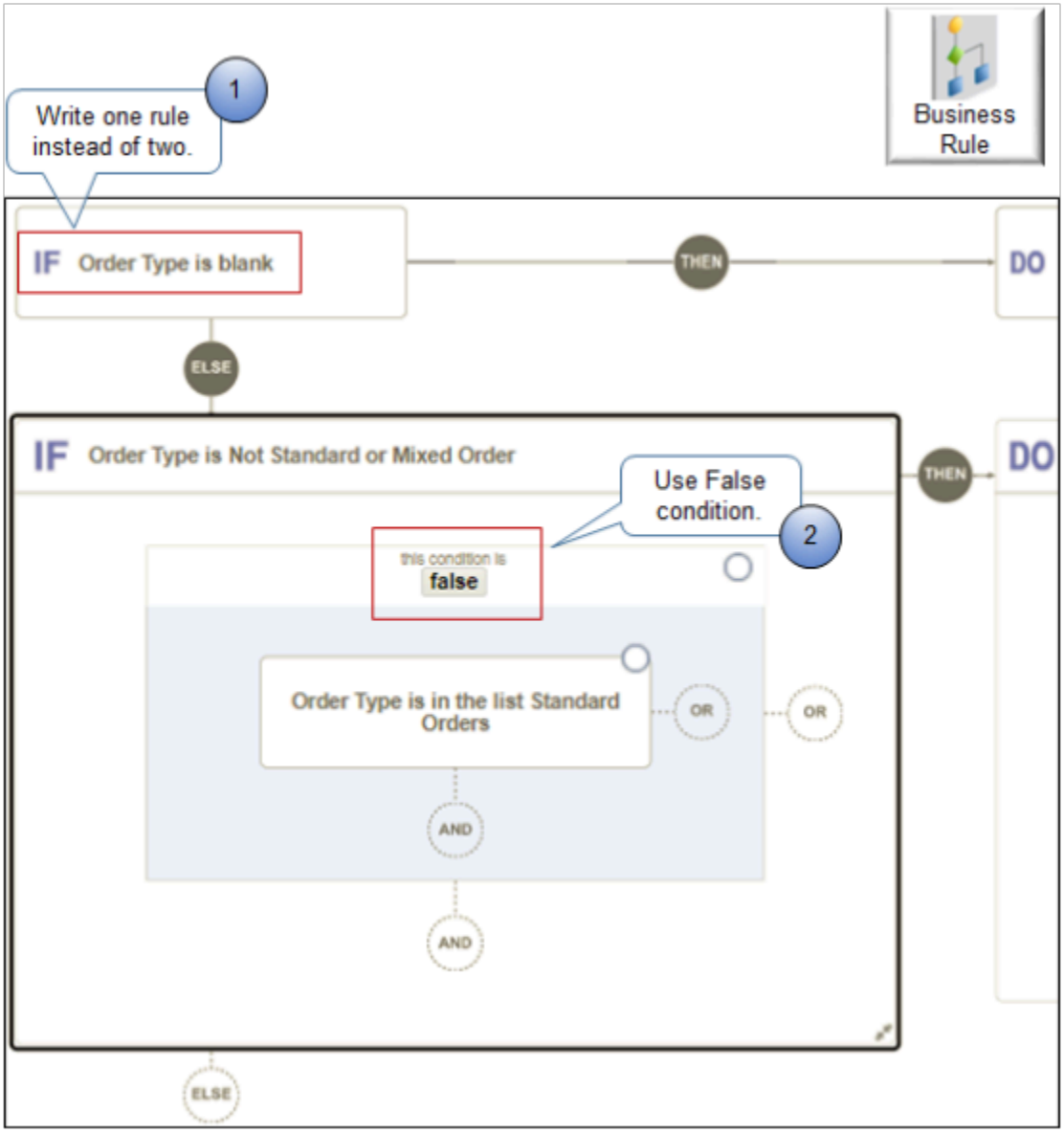
```
Attempted to invoke method "longValue" in class "java.lang.Long" on a null object
 reference. at line 29 column 13 in /Ruleset(PreTransformationRS)/Rule(HQVMISubinvDefault)/
Pattern(v0_HeaderVO)/Test[8]
```

Avoid this problem. Make sure the value of any attribute that your rule references isn't empty.

Here's an example of how to do it in Oracle Business Rules.



Use rule logic in Visual Information Builder to simplify creating and managing your rule.

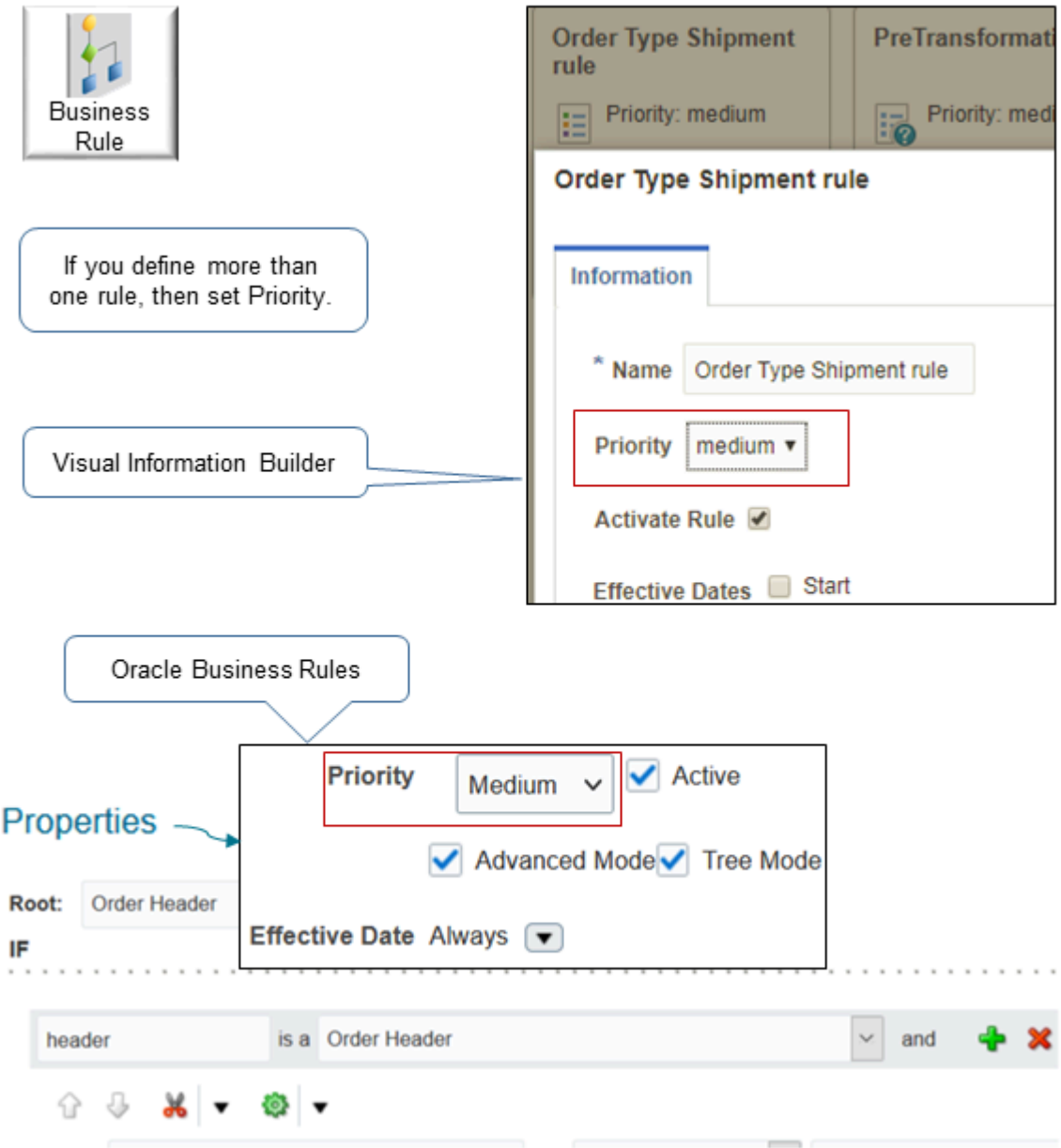


Note

1. Structure your rule.
    - Put all your logic in one rule instead of spreading it across more than one rule.
    - Write one rule that includes more than one IF_THEN_ELSE statement instead of writing more than one rule.
    - Add an IF THEN ELSE structure.
    - Nest your rules, such as adding another IF after an ELSE.
2. To look for values that aren't in the list, write a rule that looks for values in the list, then add a False condition. Use False directly in the IF statement to implement a NOT condition rather than adding another rule.

Set the Priority

If you set up more than one rule to meet the same business requirement, then set the priority.

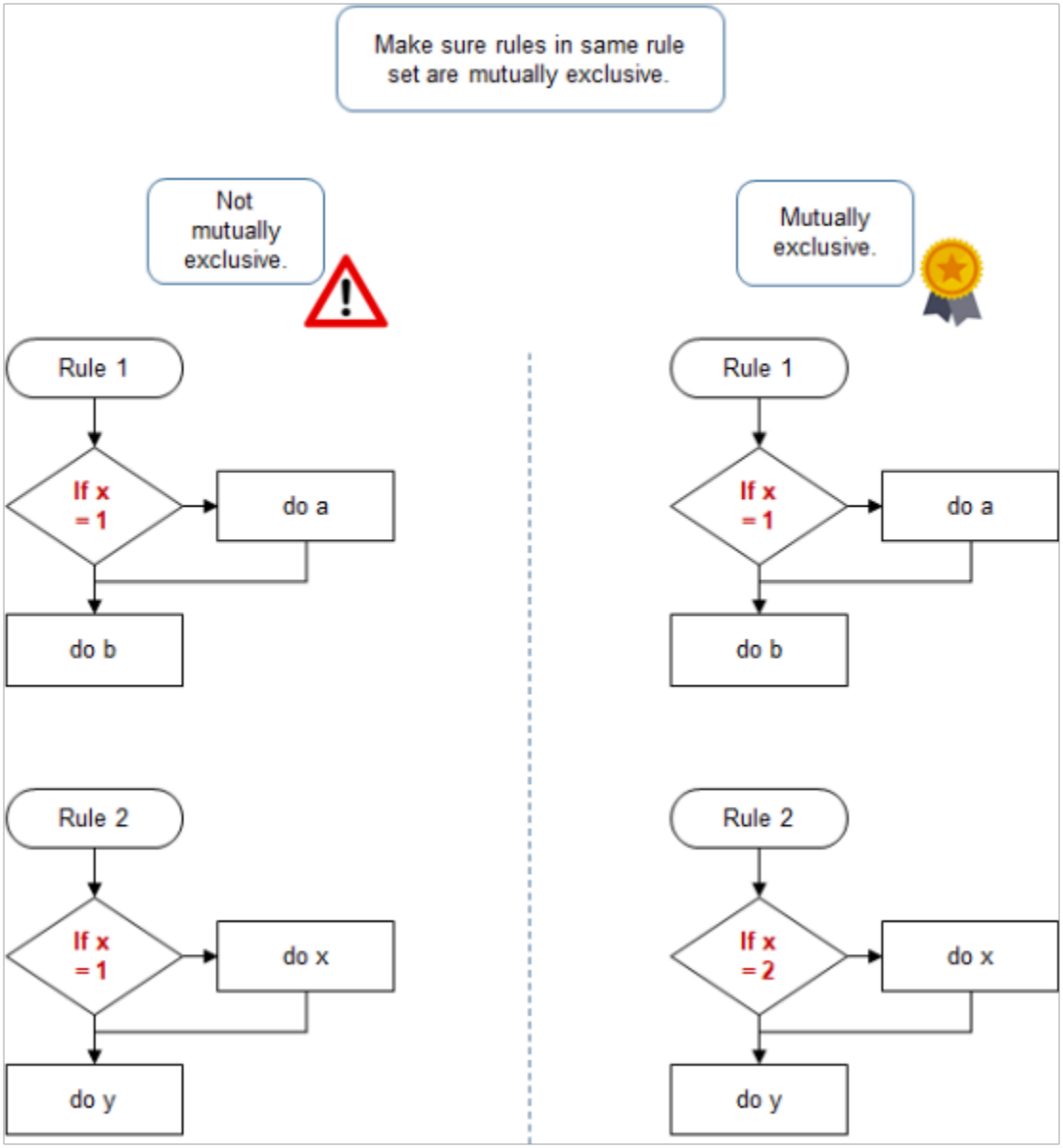


The priority determines the sequence that Order Management uses when it applies the rule. Assume you set priority to Medium for rule x and to High for rule y. Assume the conditions in the If statements in rules x and y evaluate to true. Order Management will apply rule y on the orchestration process first, and then will apply rule x.

For details, see *Guidelines for Pausing Orchestration Processes*.

Make Sure Rules Are Mutually Exclusive

Make sure the rules in a single rule set are mutually exclusive.



A rule set is a group of rules that run together to achieve a business objective. For example, assume you create a pause rule and a release rule on a pause task. The pause rule and release rule together constitute one rule set. Oracle Business Rules runs all the rules in a rule set. To avoid logic problems, make sure no two rules can be true at the same time, or false at the same time.

Create If Statements

- Use an If Then format that uses natural language when you first begin writing your rule. For example:

  ```
  If the item type is server, then add extra packing to the shipping instructions.
  ```

- Remove dependencies where the result of one rule changes the If statement of another rule in a way that causes infinite looping.

- Consider the outcome under a variety of data inputs. The If Then statement is the most common statement. You can also use If Then Else in advanced mode.
- Add the equivalent of an Else clause so your rule can handle the situation where no condition is met. To avoid an interruption in processing, this is particularly important when your rule assigns an orchestration process or routes processing to a fulfillment task.
- Use a bucket set to simplify If logic in the decision table. Use parameters on actions to provide different values for the same attribute in each rule.
- Use a decision table when you require more than one rule that uses the same set of If statements and Then statements.

Create If Statements in Oracle Business Rules

Use different conditions in Oracle Business Rules depending on the update that you're using.

Old Updates	Current Update	Description
For Each Case	Each	Repeat for each of the row that satisfies the conditions of the If statement.
`There is a case where`	At least one	At least one of the rows satisfies the conditions of the If statement.
`There is no case where`	None	None of the rows satisfies the conditions of the If statement.
Aggregate	Aggregate	Aggregate fulfillment lines before you send them to your fulfillment system.

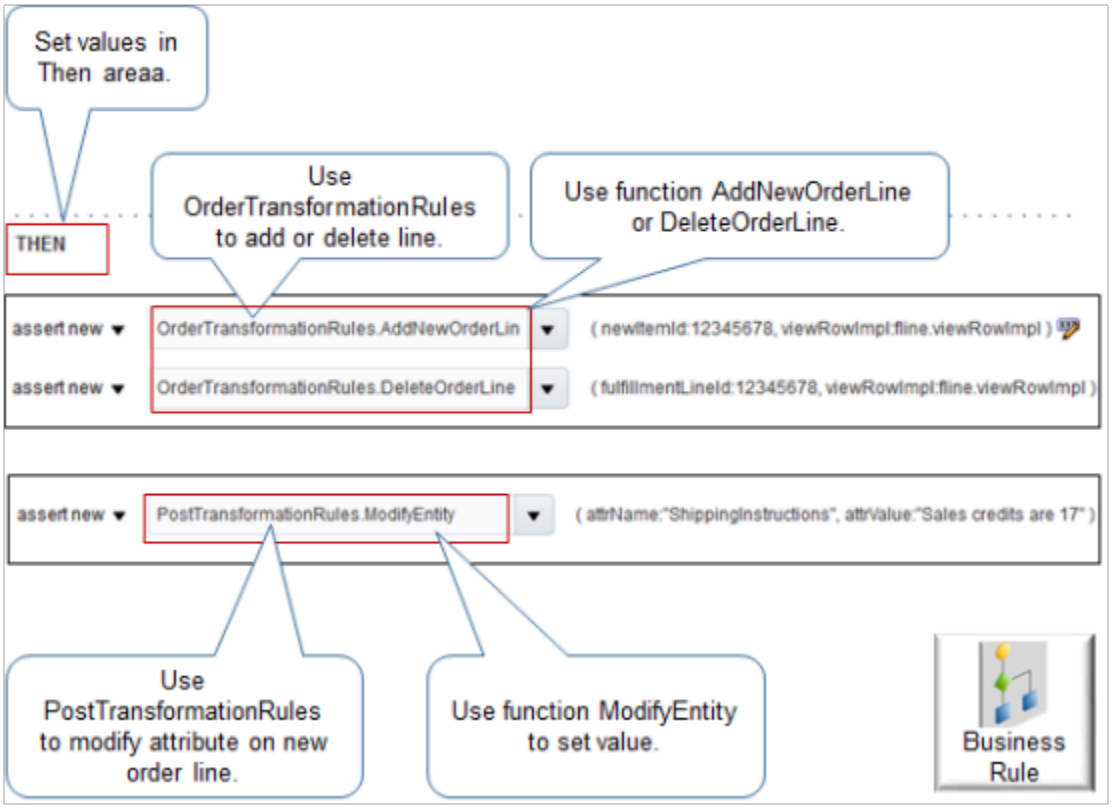
Related Topics

- Guidelines for Pausing Orchestration Processes

Create Different Types of Business Rules

Apply guidelines to help you create different types of business rules.

Transformation Rule



Note

- Use the Then statement to set the values.
- Use a product transformation rule to add a new order line or to delete an existing one.

  Here's some code that adds a new fulfillment line for item 12345678.

  ```
  OrderTransformationRules.AddNewOrderLine (newItemId:12345678, viewRowImpl:fline.viewRowImpl)
  ```

  This code deletes the fulfillment line that references item 12345678.

  ```
  OrderTransformationRules.DeleteOrderLine (fulfillmentLineId:12345678, viewRowImpl:fline.viewRowImpl)
  ```

- Use a posttransformation rule to modify an attribute on a new order line.
- Use the ModifyEntity function to set a value.

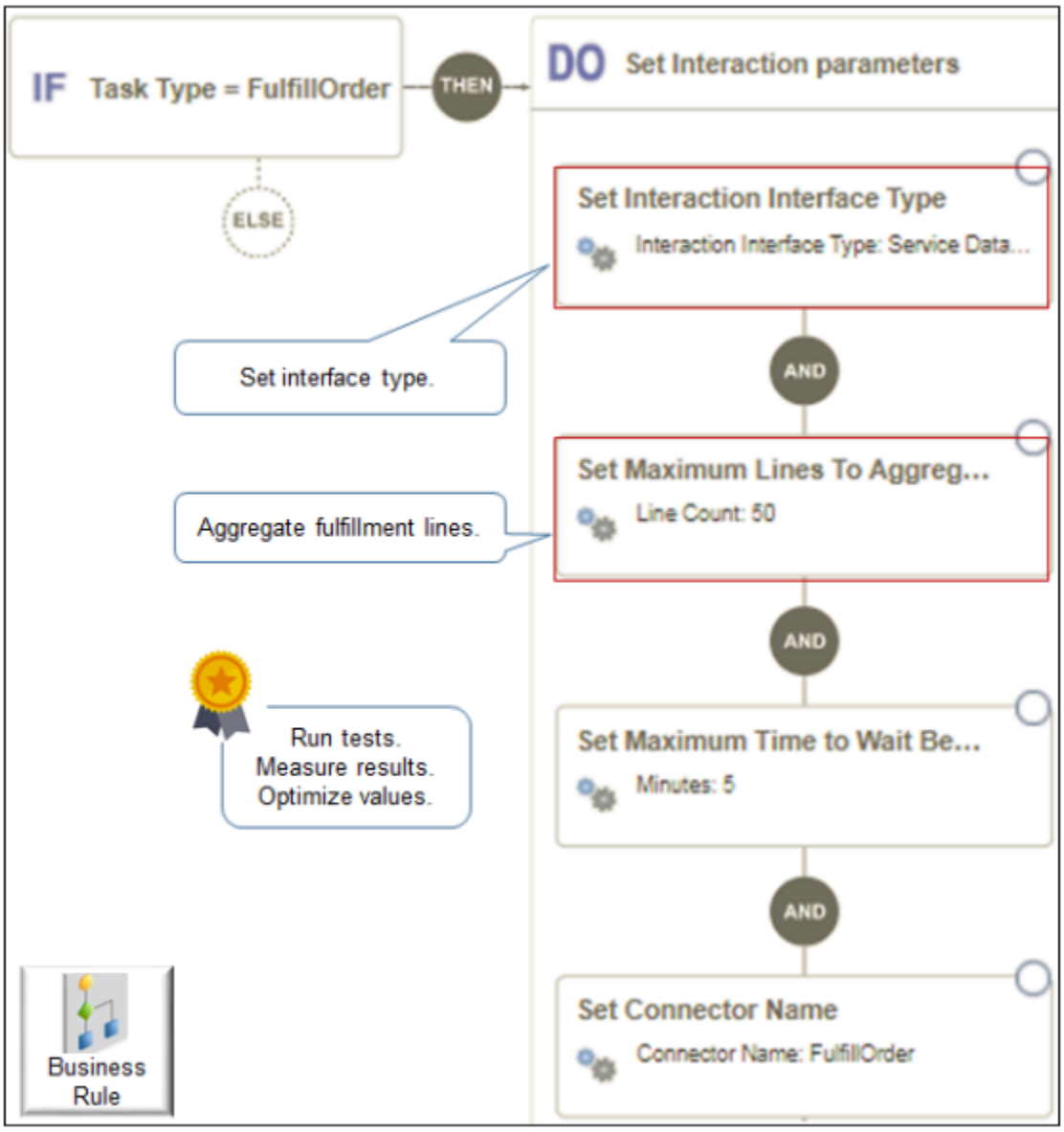
  This example code modifies the ShippingInstructions attribute. It sets the contents of the attribute to the text `Sales credits are 17.`

  ```
  PostTransformationRules.ModifyEntity(attrName:ShippingInstructions, attrValue: "Sales credits are 17.")
  ```

- Make sure you use each rule only to do work for its intended design. For example, use a product transformation rule only to transform an item. Don't use it to implement some other functionality.
- If the order header and the order line contain the same attribute, such as the Freight Terms attribute, and if you need to transform the value on the line, then use a posttransformation rule. Don't use a pretransformation rule. Order Management cascades values from the header to the line when you click Submit, so it will replace the

value that you set with a pretransformation rule with the value from the header. But a posttransformation rule runs after you click Submit and after Order Management cascades values.

Interface Routing Rule



Note

- Use parameters, such as Maximum Lines to Aggregate, to aggregate the lines you send to your fulfillment system.
- For details, see *Manage Routing Rules*.
- Run several tests and measure the results. Optimize values for each parameter after each test.

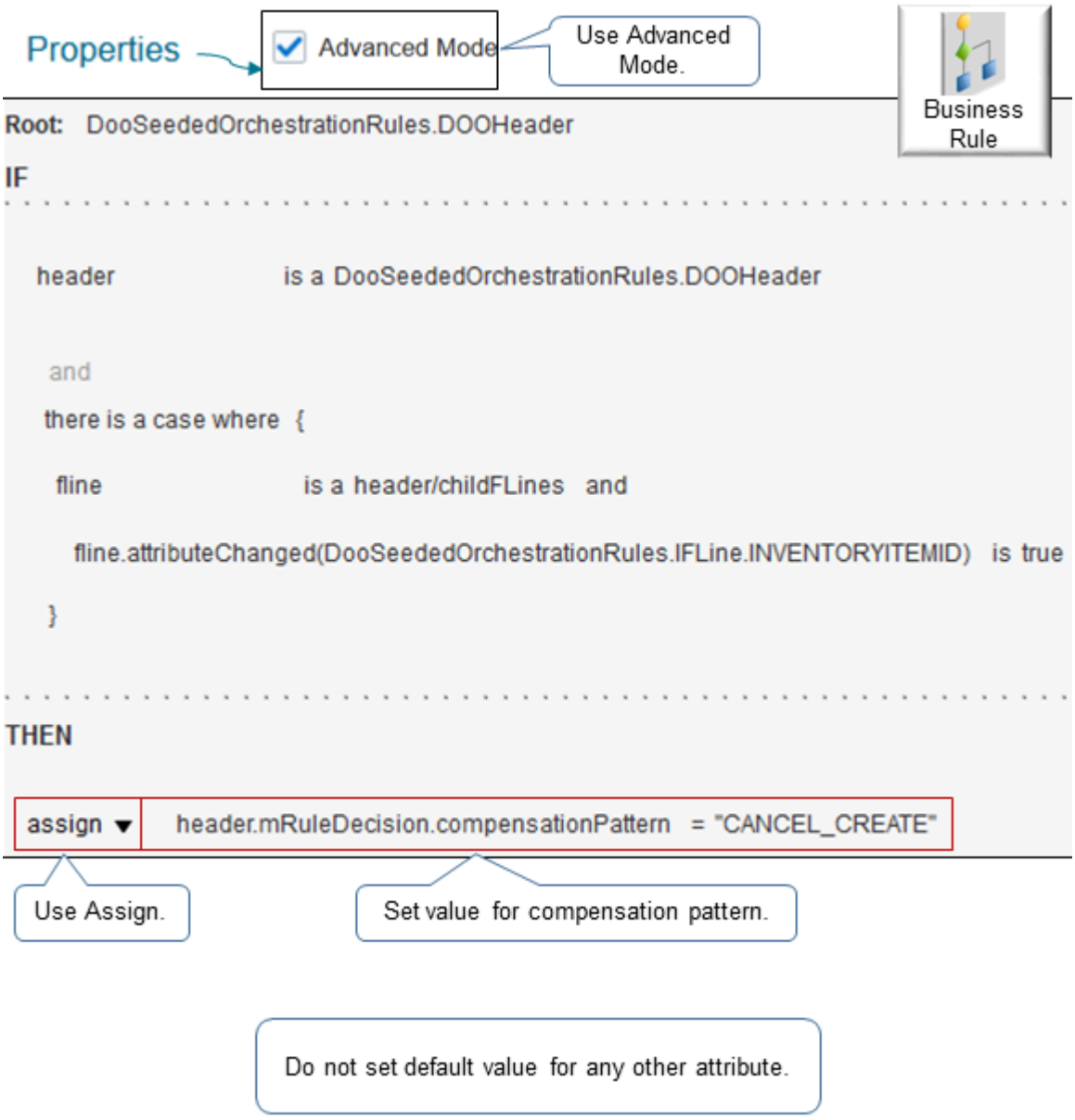
- Don't create a routing rule that depends on a pause task and that sets the connector. For example, if you use Visual Information Builder, don't create a rule that's similar to.

  ```
  If Task Type is equal to pause, then set connector name to Fusion-Reservation.
  ```

  If you use Oracle Business Rules, don't create a rule that's similar to.

  ```
  If header.Task Type is "Pause", then Set Connector Name ( "Fusion-Reservation" )
  ```

Compensation Pattern



Note

- Use Advanced Mode.
- Use the Assign action.
- Assign a value for the compensation pattern. For example:

```
Assign header.mRuleDecision.compensationPattern = "CANCEL_CREATE"
```

- Don't use this rule to set the default value for any other attribute.

Cost of Change Rule

Note

- Use Advanced Mode.
- Use the Assign action.
- Assign a numeric value, such as 10 to costOfChange. For example:

```
assign Header.mRuleDecision.costOfChange = 10
```

- Don't use this rule to set the default value for any other attribute.

Use Business Rules in Orchestration Processes

Use guidelines to help you create a business rule in an orchestration process.

For details, see *Guidelines for Setting Up Orchestration Process Steps*.

Lead-Time Expression Rule

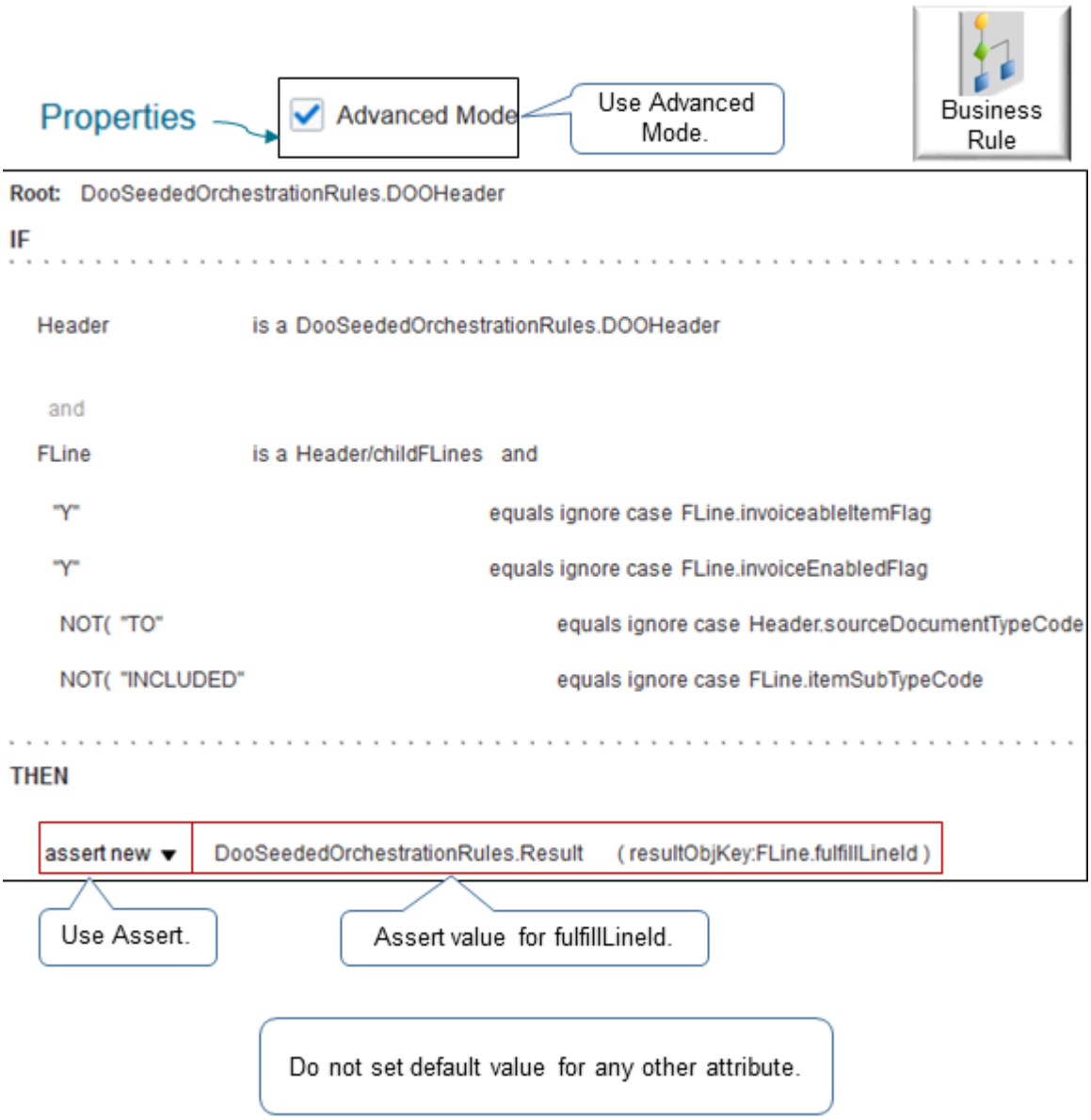
Note

- Use Advanced Mode.
- Use the Assign action.
- Assign a datetime value. For example:

```
assign Header.mRuleDecision.leadTime = (Fline.scheduleShipDate.time - CurrentDate.date.timeInMillis)/
(1000*60*60*24)
```

- Don't use this rule to set the default value for any other attribute.

Line-Selection Rule



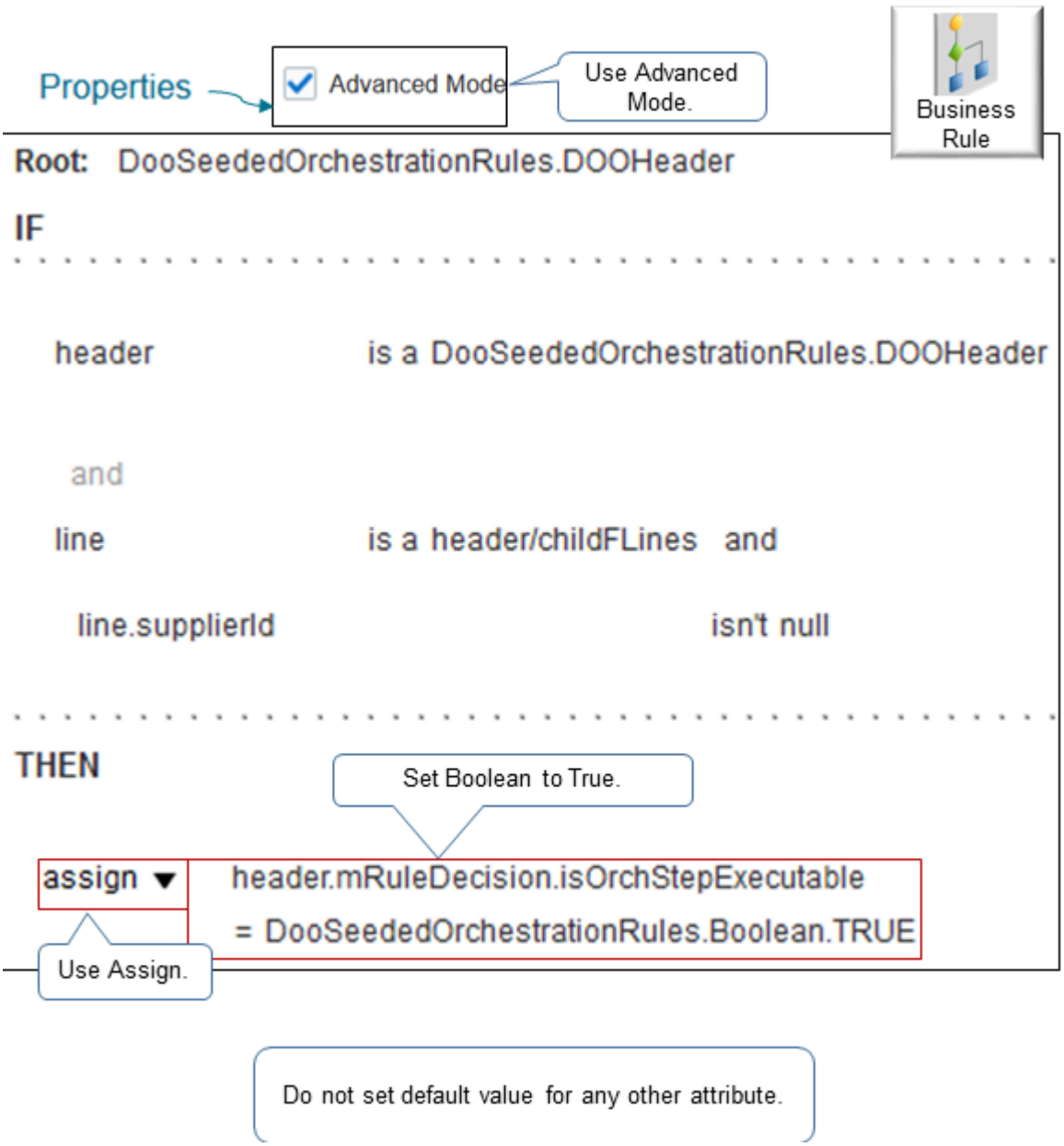
Note

- Use Advanced Mode.
- Use the Assert action.
- Assert value for fulfillLineId. For example:

```
assert DooSeededOrchestrationRules.Result (resultObjKey.Fline.fulfillLineId)
```

- Don't use this rule to set the default value for any other attribute.

Branching Condition Rule



Note

- Use Advanced Mode.
- Use the Assign action.
- Assign the Boolean to True. For example:

  ```
  assign Header.mRuleDecision.isOrchStepExecutable = DooSeededOrchestrationRules.Boolean.TRUE
  ```

- Don't use this rule to set the default value for any other attribute.

Pause Tasks

Set up a pause task on an orchestration process. For details, see *Guidelines for Pausing Orchestration Processes*.

Related Topics

- Guidelines for Setting Up Orchestration Process Steps

Use Business Rules When You Can't Use Extensions

Create a business rule when an Order Management extension doesn't meet your requirements.

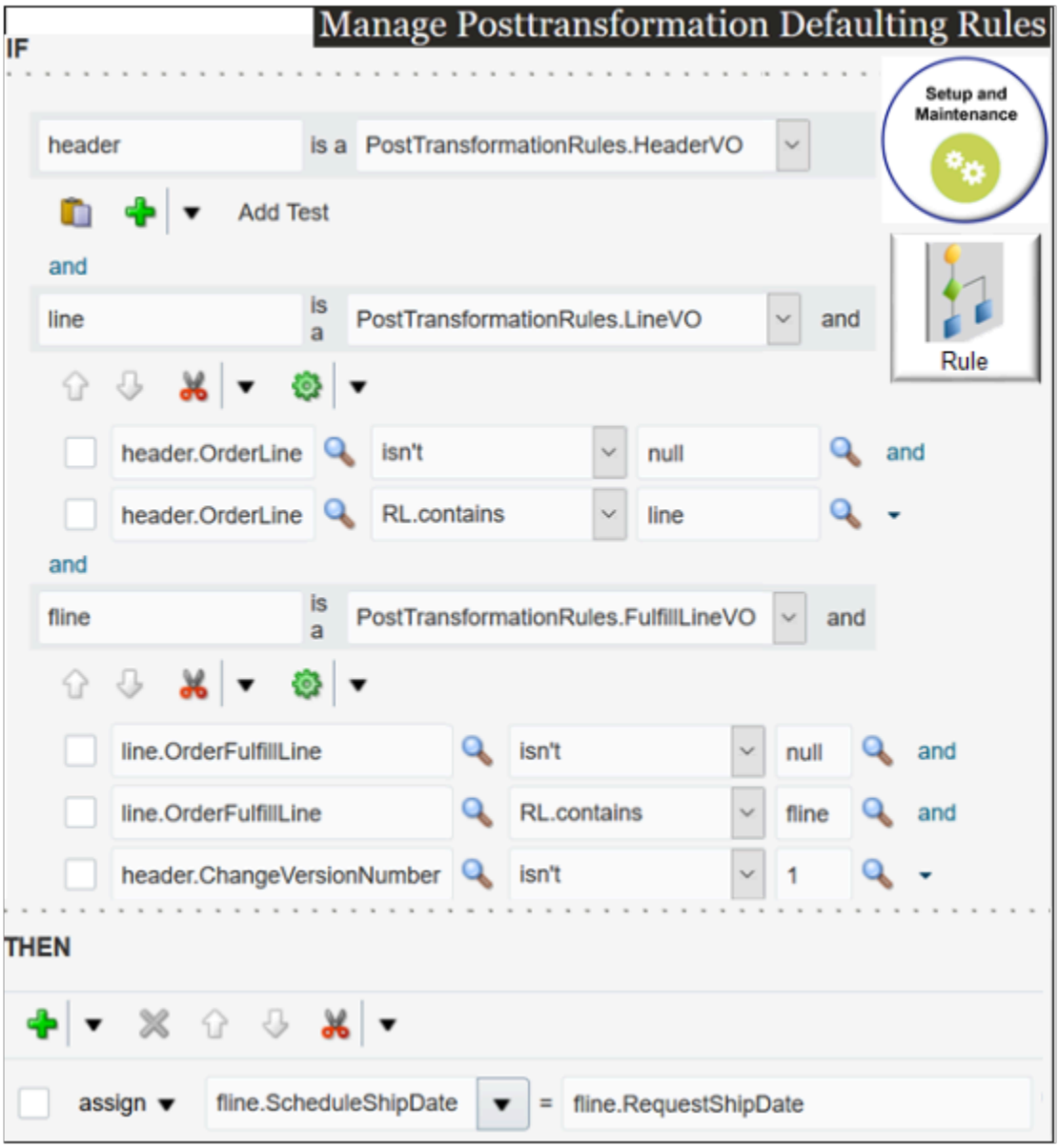
For example, you can't use an Order Management extension to update the ScheduleShipDate attribute when you revise a sales order.

Assume you must set the value of the ScheduleShipDate attribute to the value that the RequestShipDate attribute contains when you create an order revision.

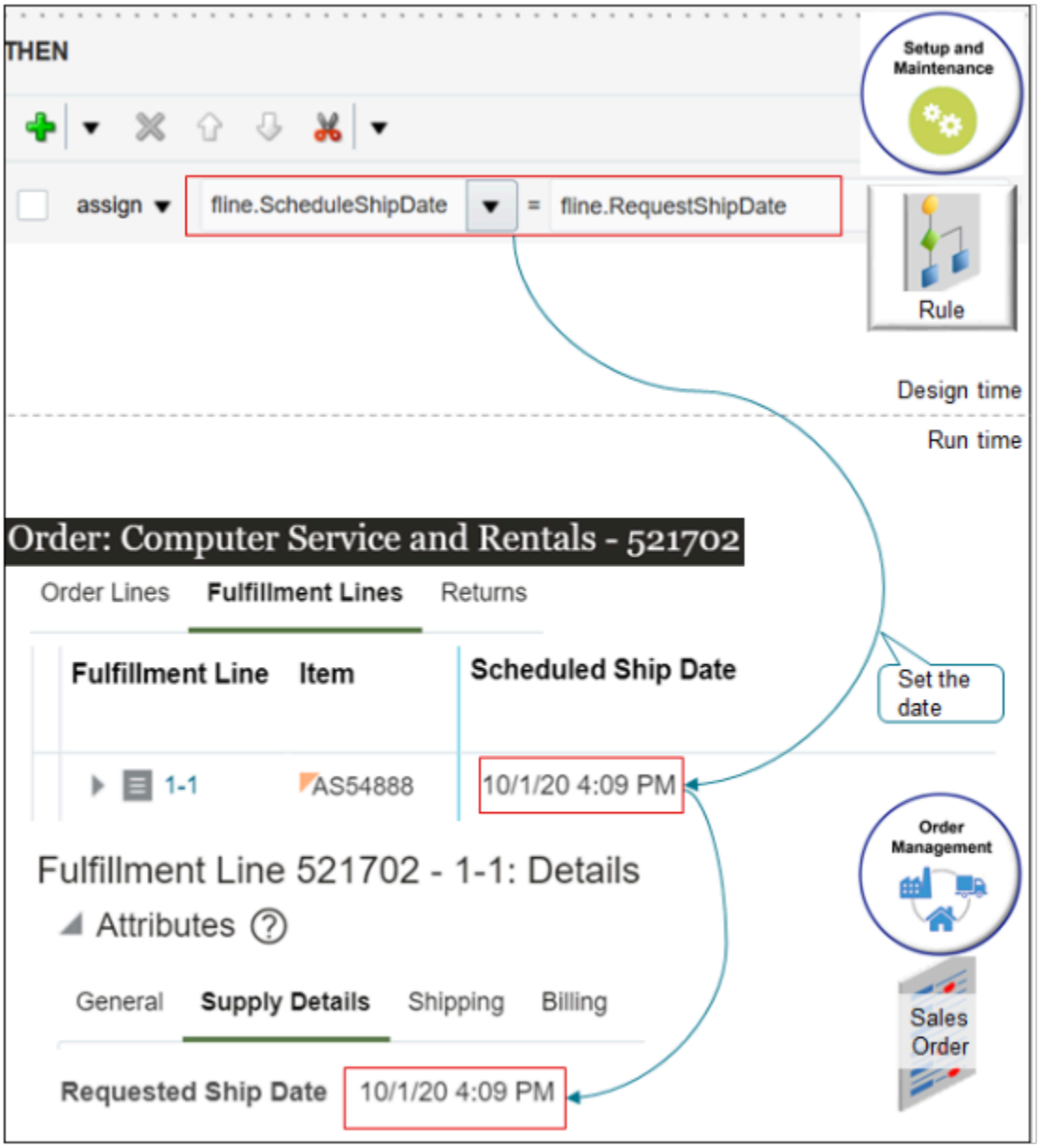
Summary of the Setup

1. Create a posttransformation rule.
2. Test your setup.

Create a Posttransformation Rule



Here's how the rule works at run time.



Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules
2. On the Manage Posttransformation Defaulting Rules page, create a new rule.
3. Add a check mark to the Advanced Mode option.

4. In the If area, set the conditions.

Code	Description
`header is a PosttransformationRules.Header`	Declare the Header variable into the PosttransformationRules dictionary. Get values for attributes of the order header that the orchestration process is currently processing from the header virtual object (VO), then store them in the Header variable.
`line is a PosttransformationRules.LineVO`	Declare the line variable into the PosttransformationRules dictionary. Get values for attributes of the order line that the orchestration process is currently processing from the line virtual object (VO), then store them in the line variable.
`header.OrderLine isn't null`	Make sure the order line contains a value.
`header.OrderLine RL.contains line`	Declare the OrderLine variable into the rules language (RL) dictionary, then set the value of OrderLine to the value that the line variable contains. This condition makes sure the OrderLine variable references the fulfillment line that the orchestration process is currently processing. It also makes sure you correctly declare the variable into the dictionary. You use it to examine all lines in the sales order, not only the line that the current orchestration process instance is processing.
`fline is a PosttransformationRules.Fulfil`	Declare the fline variable into the PosttransformationRules dictionary. Get values for attributes of the fulfillment line that the orchestration process is currently processing from the virtual object (VO) for the fulfillment line, then store them in the fline variable.
`line.OrderFulfillLine isn't null`	Make sure the line.OrderFulfillLine fact contains a value.
`line.OrderFulfillLine RL.contains fline`	Make sure the line.OrderFulfillLine fact contains data from the fline variable.
`header.ChangeVersionNumber isn't 1`	Proceed to the Then statement only if you're revising the sales order. If the ChangeVersionNumber attribute on the order header equals 1, then the sales order isn't a revision.

5. In the Then area, add an Assign action.

Code	Description
`fline.ScheduleShipDate = fline.RequestShipDate`	Set the value of the ScheduleShipDate attribute in the fline variable to the value that the RequestShipDate attribute contains in the fline variable.

6. Click **Save > Release**.

For details about how to create a business rule, see *Overview of Using Business Rules With Order Management*.

Test Your Setup

1. Go to the Order Management work area, create, then submit a sales order.

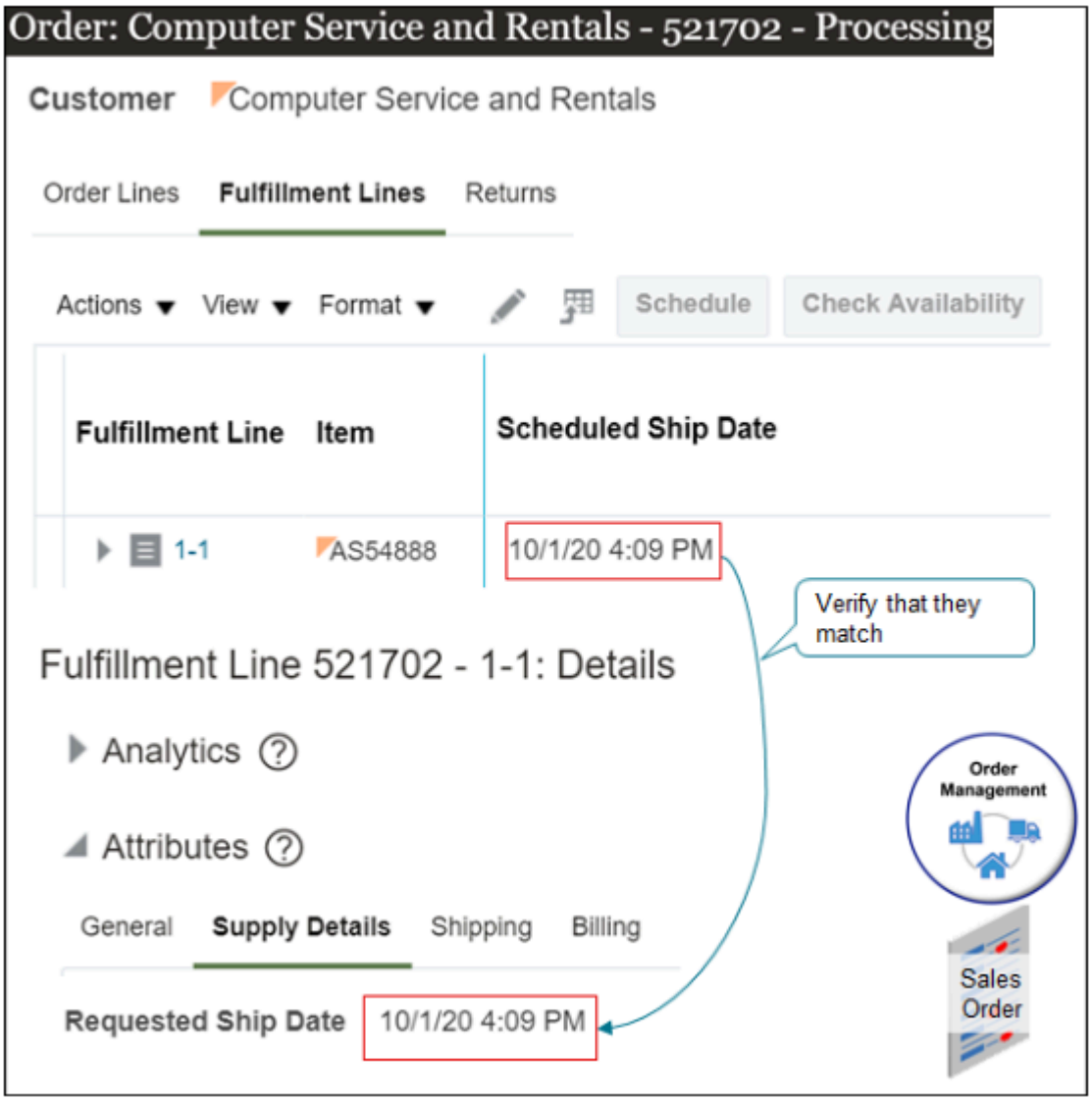
Attribute	Value
Business Unit	Vision Operations

Assume the order number is 521702.

2. Revise sales order 521702, then submit your revision.
3. On the Order page, click **Fulfillment Lines**.
4. On the order line, examine the value of the Scheduled Ship Date attribute.
5. In the Attributes area, click **Supply Details**.

6. Examine the value of the Requested Ship Date attribute, and verify that it matches the Scheduled Ship Date.

   For example:



Use REST API

If you don't want to use a business rule or an extension to set the scheduled ship date, you can use a patch action on the salesOrdersForOrderHub REST API. For example:

```
{
 "lines":[
 {
 "SourceTransactionLineId": "1",
 "SourceTransactionScheduleId": "1",
 "OverrideScheduleDateFlag": true,
 "ScheduleShipDate": "2019-10-25T05:59:59+00:00",
 "ShippingInstructions":"REST NEW TEST",
 "RequestedFulfillmentOrganizationCode": "M1"
 }
 ]
```

```
}
```

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click **Sales Orders for Order Hub > Update One Sales Order**.

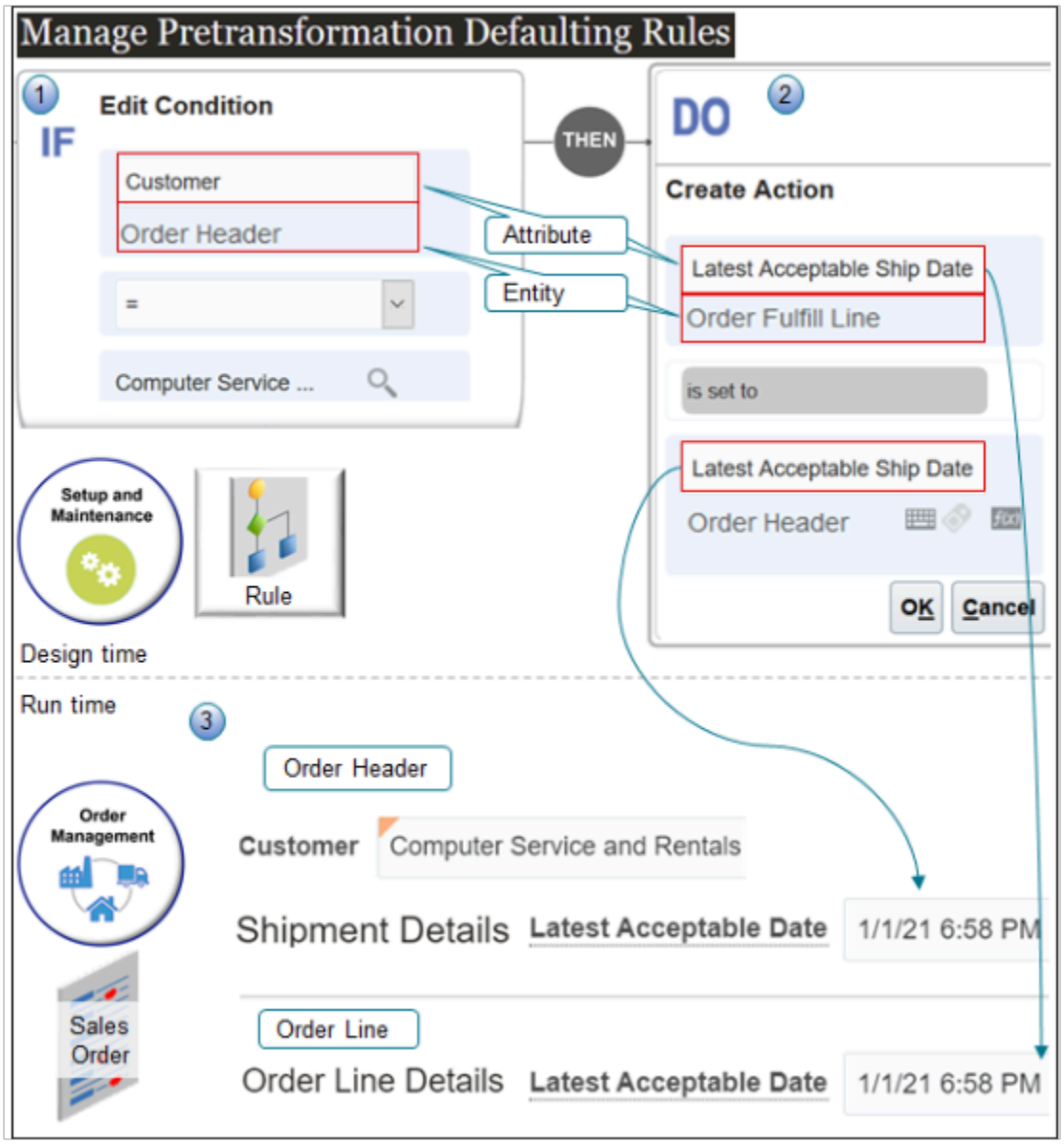
Related Topics

- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules
- Import Shipping Method
- Use SQL to Query Order Management Data

Reference Attributes in Business Rules

You can reference an attribute from a variety of sources in a business rule.

For example:



What the Numbers Mean

1. The condition references an attribute that resides in an entity, such as the Customer attribute in the order header entity.

   ```
   If the Customer attribute in the Order Header entity equals Computer Service and Rentals
   ```

2. The action references attributes and entities.

   ```
   Set the Latest Acceptable Ship Date attribute in the Order Fulfill Line entity to the same value that
    the Latest Acceptable Ship Date attribute in the Order Header entity contains.
   ```

3. Order Management applies the rule at run time. Here's the pseudocode.

   ```
   The Customer attribute in the Order Header entity equals Computer Service and Rentals, and the Latest
    Acceptable Ship Date attribute in the Order Header entity contains 1/1/21 6:58 PM, so set the Latest
    Acceptable Ship Date attribute in the Order Fulfill Line entity to 1/1/21 6:58 PM.
   ```

General Guidelines

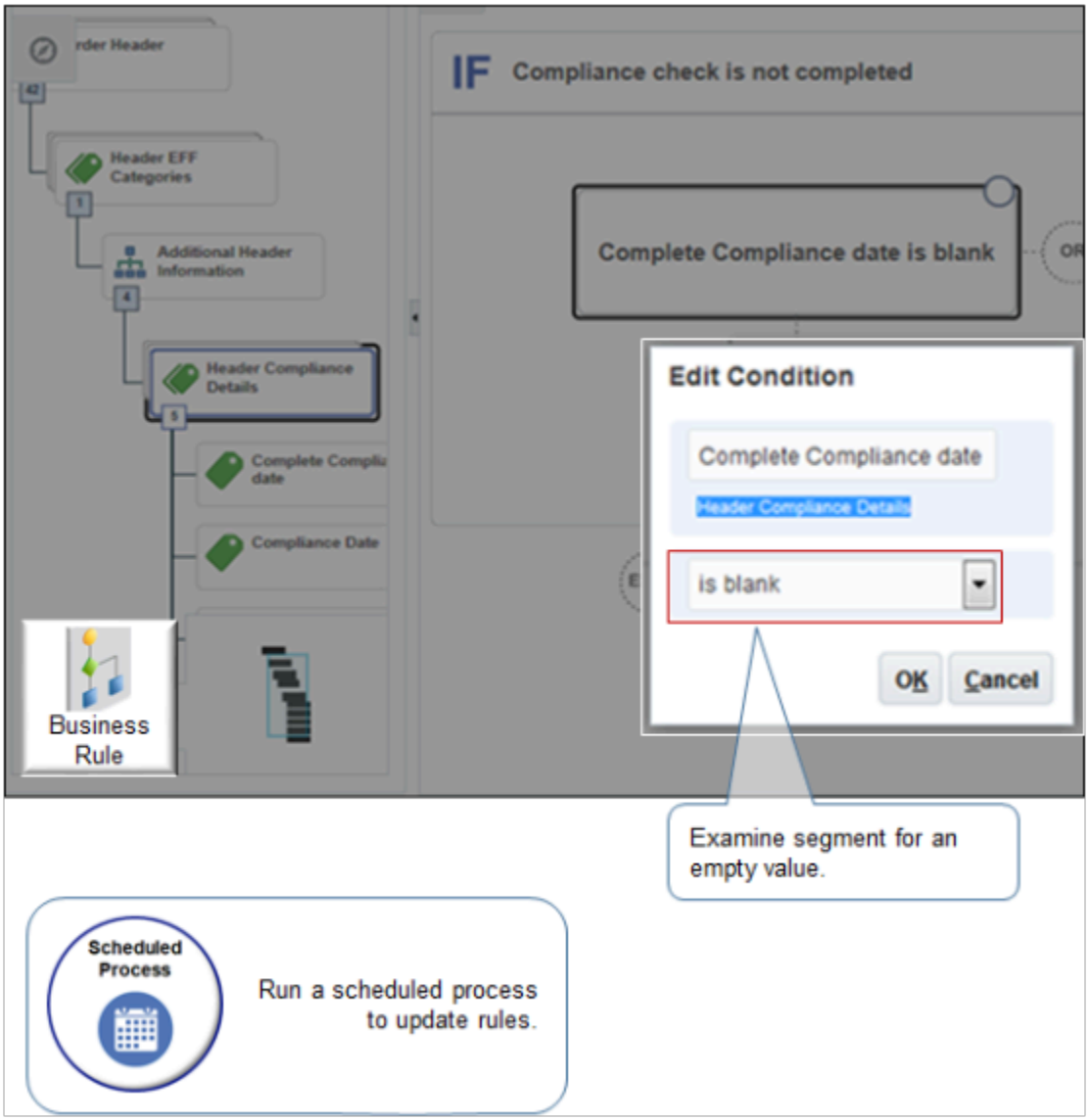
Note

- Identify the attributes that you will reference in the rule. For example, Item Type and Shipping Instructions are each an attribute on the fulfillment line. For details about Order Management tables that these attributes reference, go to *Tables and Views for Oracle Fusion Cloud SCM*, then examine the Order Management chapter.
- Add a default, catch all rule that can handle the situation where you must set an attribute value.
- Make sure your rule can handle an attribute that doesn't contain a value, even for attributes that normally do contain a value.

  > **Note:** If the attribute doesn't contain a value, and if your rule does a calculation that requires a value, then it might create an error or result in a null pointer exception at runtime.

  For example, an Order Entry Specialist might or might not set the value for an extensible flexfield because adding a value is optional. If your rule references an extensible flexfield, and if this flexfield doesn't contain a value at run time, then write your rule so it populates the flexfield with a default value that your rule can use during calculations.
- You can reference a variety of attributes and entities.

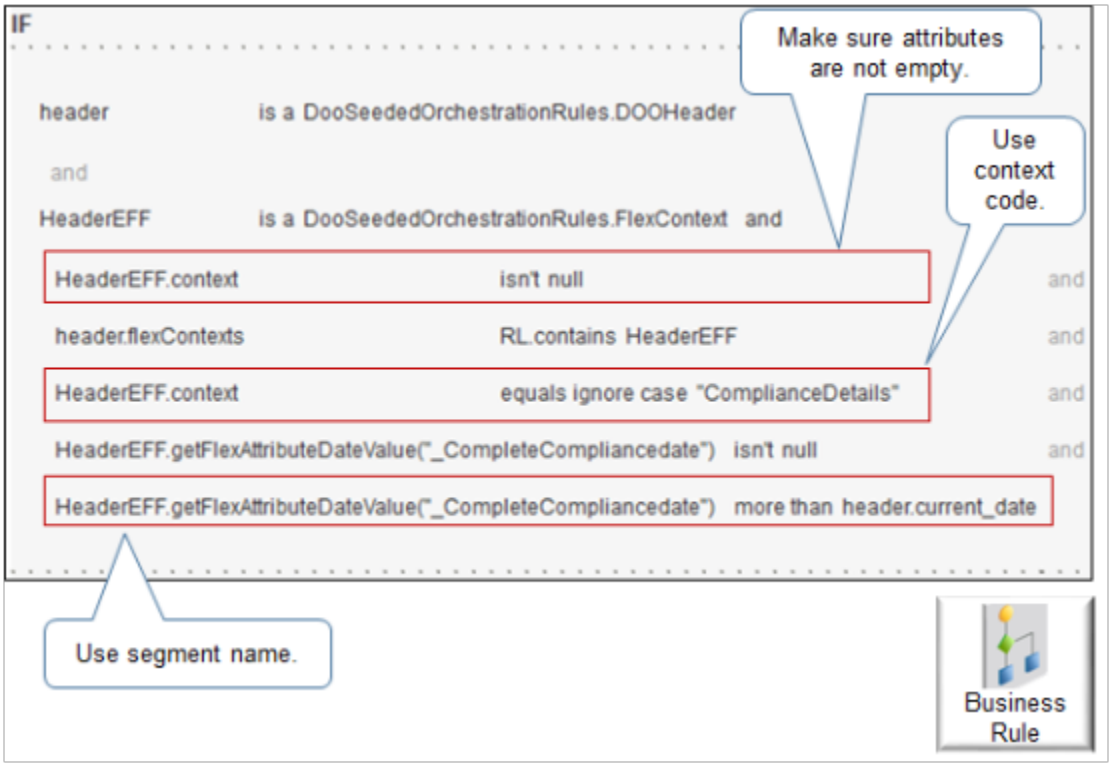
Reference Extensible Flexfields



Apply guidelines when you reference an extensible flexfield in Visual Information Builder.

- Its not necessary to examine the context for an empty value.
- Your rule must examine the segment for an empty value.
- Run the *Publish Extensible Flexfield Attributes* scheduled process whether you use Visual Information Builder or Oracle Business Rules. This scheduled process updates the rule definitions. You must run it each time you set up a new or modify an existing extensible flexfield so you can view and use them in the rule editors.

Apply guidelines when you reference an extensible flexfield in Oracle Business Rules.



Note

- Use `isn't null` to make sure the value of any attribute or object, such as the segment or context, that your rule references isn't empty. If your rule references an empty attribute, but depends on a value, then it might fail with unpredictable results.
- Reference the context code, such as `HeaderEFF.context`.
- Reference the segment name, such as `HeaderEFF.getFlexAttributeDateValue("_CompleteCompliancedate")`.
- If you reference a value set, then Oracle Business Rules uses the VALUE column, by default. If the value set.
    - **Contains a value for ID**. The editor uses the ID, and you must write the rule so it uses the ID.
    - **Doesn't contain a value for ID**. The editor uses the value.

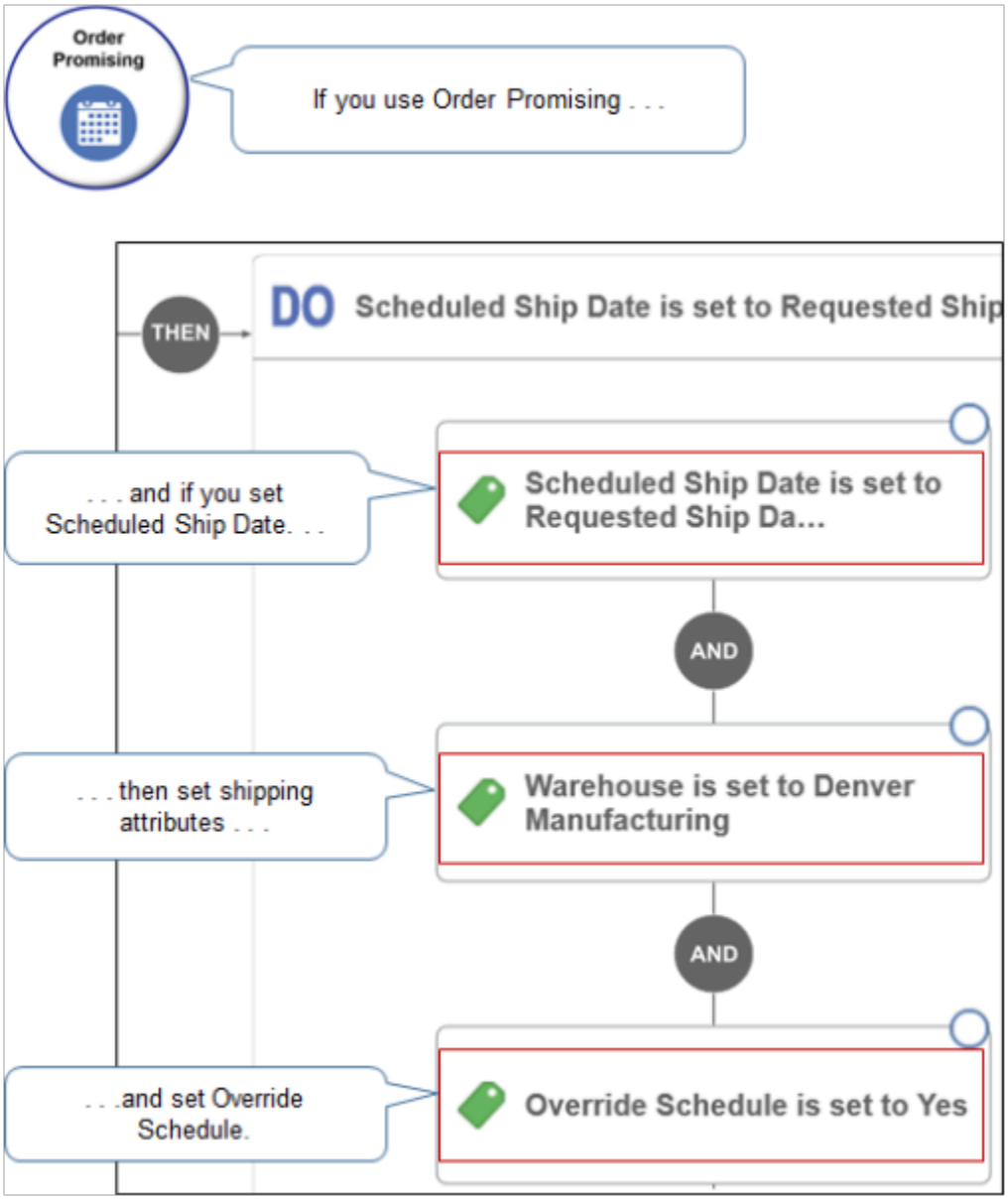
  For example, if you write a rule that references a payment term, then you must make sure the value set includes a value for ID, and your rule must reference the ID.

Get more information.

- Learn how to identify the context code and segment name. For details, see *Identify Flexfield Contexts and Category Codes for Your Business Rules*.
- Use Extensible flexfields. For details, see *Overview of Using Extensible Flexfields in Order Management*.

Reference Order Promising

If you use Order Promising, then apply these guidelines.

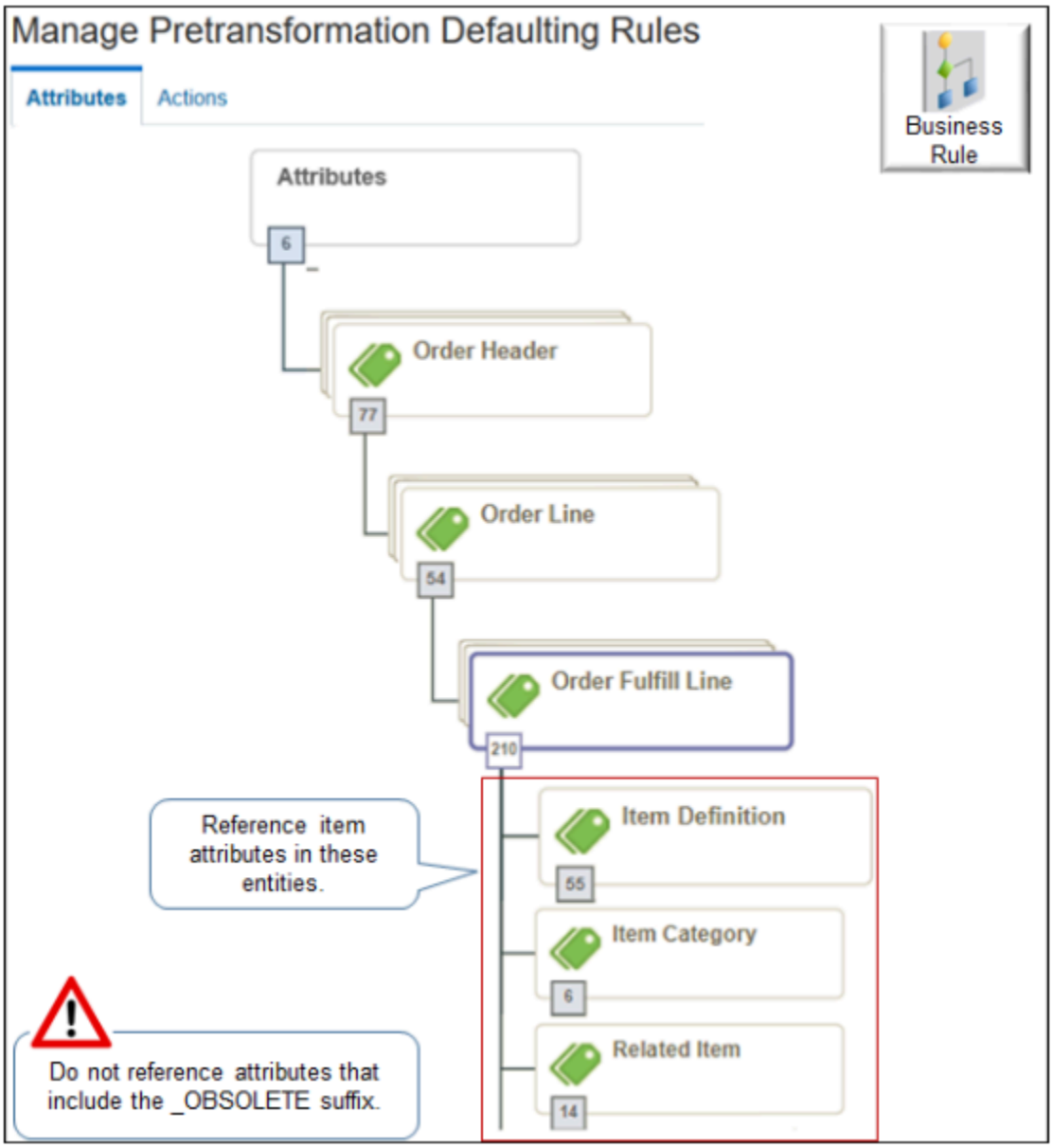


Note

- Order Promising uses the Scheduled Ship Date attribute on the sales order only if you set the Override Schedule attribute to Yes. If you create a rule that references Scheduled Ship Date, then make sure you override the schedule.
- Specify the warehouse, or specify the supplier and supplier site. It isn't necessary to specify scheduled arrival date.
- If you don't make these settings, then Order Promising ignores the scheduled ship date that the rule sets.

Reference Item Attributes

You can reference an item attribute in an entity in Visual Information Builder.



Note

- You can reference an item attribute from these entities.
    - Item Definition
    - Item Category
    - Related Item
- These entities are children of the Order Fulfill Line entity, grandchildren of Order Line, and great grandchildren of Order Header.
- You can reference an item attribute in Visual Information Builder or Oracle Business Rules.
- Don't reference an item attribute that ends with suffix _OBSOLETE in the attribute name.

Related Topics

- Overview of Using Extensible Flexfields in Order Management
- Overview of Setting Up Extensible Flexfields in Order Management
- Identify Flexfield Contexts and Category Codes for Your Business Rules

Migrate Business Rules in Order Management

Migrate your business rules across different instances of Order Management. For example, migrate business rules from your test environment to your production environment.

Here are the types of rules you can migrate.

- Approval
- Routing
- Pretransformation
- Posttransformation
- Transformation
- Product transformation
- Process assignment

Note

- Business rules that you add to an orchestration process are part of the orchestration process definition. You use the Functional Setup Manager to export these definitions from one environment and to import them into another environment.
- In most situations, an upgrade or patch won't affect your rules, with these exceptions.
    - The upgrade or patch might overwrite modifications you make to a predefined rule or decision table.
    - The upgrade or patch might change rule behavior, and this change might affect the behavior of your business rules.

Migrate business rules in Order Management.

1. Make sure the source environment and the target environment are on the same update.

   You can't migrate rules across different updates. To get your rules into a new update, upgrade the source environment so its on the same update as the target environment, then migrate your rules.
2. Make sure each of your rule specifies attribute values that exist.

Example Rule	Description
Assume you create rule in Oracle Business Rules. `If Business Unit equals "203", then set FulfillOrdID to "207"`	If Business Unit 203 or FulfillOrdID 207 doesn't exist in your target environment, then the migration will fail. To avoid this error, make sure these entities exist in your target environment.

Example Rule	Description
Assume you create a rule in Visual Information Builder. `If Business Unit equals "Vision Operations" then set Warehouse to "Singapore Distribution"`	If Business Unit Vision Operations or Warehouse Singapore Distribution doesn't exist in your target environment, then the migration will fail. To avoid this error, make sure these entities exist in your target environment.

3. Make sure each rule operation that uses `Starts with` or `Contains` includes the full attribute value or ID value.
4. For each attribute that your rule references in the source environment, make sure you also set up each of these attributes in the target environment.
   For example, if a rule in your source environment references the Primary Salesperson attribute, then make sure the target environment also includes Primary Salesperson.
5. Make sure none of the rules you're about to migrate contain errors.
   - Test each rule in the source instance.
   - Make sure your rules specify values that use the same data type, and that the attribute can contain the data type.
   - The migration examines each rule in the source that you specify to migrate. If any of these rules contain an error, then the migration will fail.
6. Make sure the rules you migrate replace the functionality that rules in your target environment requires. Migration deletes all rules in the target environment before it does the migration, including rules you don't set up in the source. For example, if the target includes rules x, y, and z, and if the source includes rules x and y, then the migration deletes rules x, y, and z from the target, then copies rules x and y from the source to the target. If the target depends on rule z, then copy rule z to the source before you do the migration.
7. Make sure you release each rule you must migrate.
   If you create a business rule in the source environment but don't release it, then the migration won't migrate the rule. Migration only migrates rules you save and release.
8. If your rule references an extensible flexfield, then make sure you set up each of flexfield in the same way in the source environment and in the target environment.
   The Manage Configuration Packages page doesn't examine your extensible flexfield setup. You must manually examine your setup.
9. Migrate your rules.
   - Go to the Setup and Maintenance work area.
   - On the Setup page, click **Tasks > Manage Configuration Packages**.
   - On the Manage Configuration Packages page, search for your configuration package.
   - In the search results, click the **row** that includes your configuration package, then click **Export Setup Data**.
   - On the Export Setup Data page, click **Next > Submit**.
     The Setup and Maintenance work area exports business rules from your source environment and imports them into your target environment.

Related Topics

- Overview of Using Business Rules With Order Management
- Copy Setups Between Instances of Order Management
- Export Setup Data Using Implementation Project

Oracle Business Rules

Use the Business Rules Editor

Use an editor to edit Oracle Business Rules.

Summary of the Steps

1. Access the rule editor.
2. Add the If statement.
3. Add the Then statement.

Access the Rule Editor

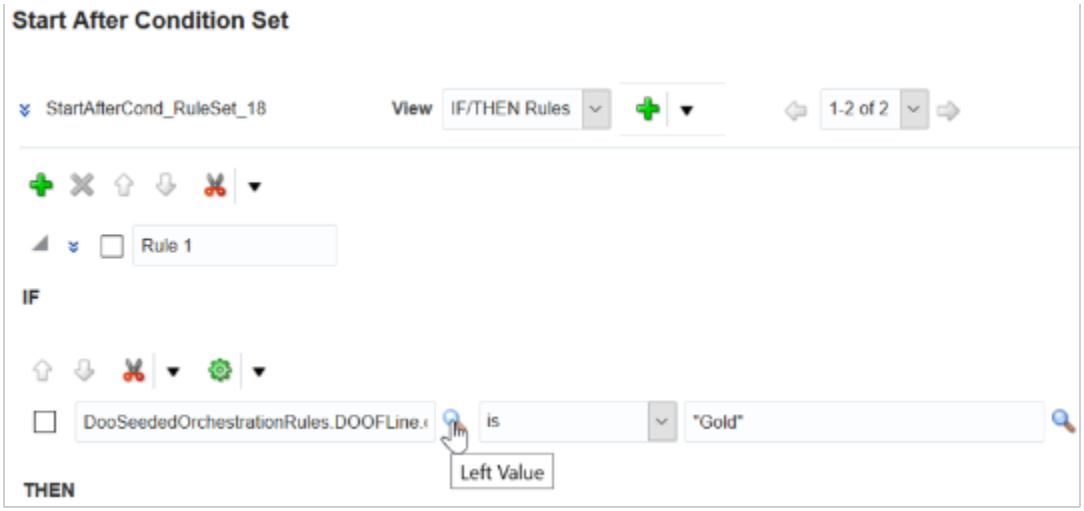
1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
2. Open one of these pages.
    - Manage Orchestration Process Definitions
    - Manage Product Transformation Rules

    The page you access depends on the set up you must do. For this example, you modify an orchestration process definition, so open Manage Orchestration Process Definitions.
3. Access the rule editor.

    You access the rule editor in different ways, depending on the page you use. For this example, you click **Click for Rule** in the Pause Rule column on the step of an orchestration process. Examine an example. For details, see *Pause Orchestration Processes Until Events Happen*.

Add the If Statement

You will create this statement.

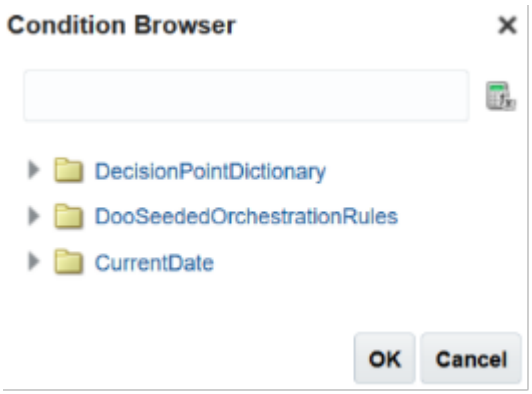


Do it.

1. Click the magnifying glass (Left Value) to access the Condition Browser.

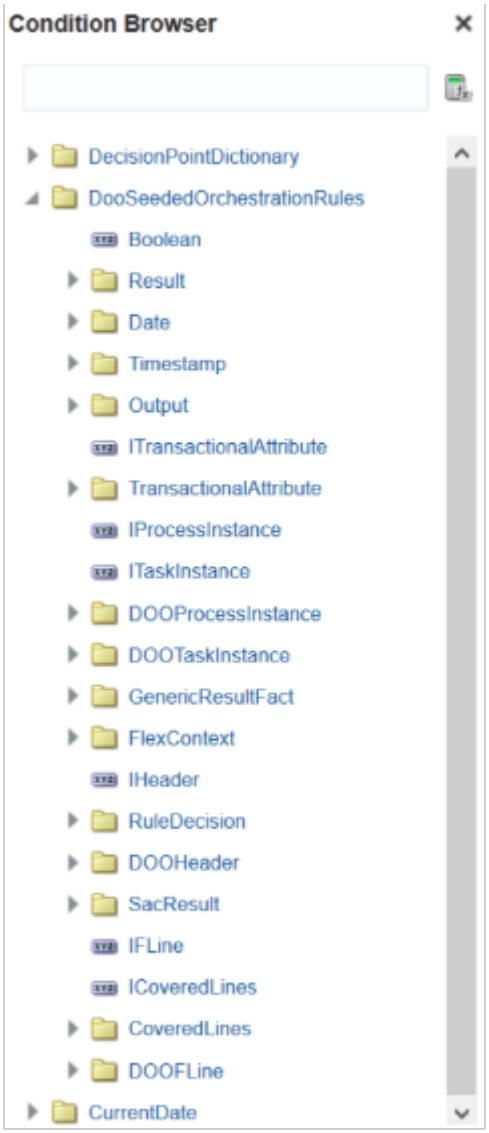
   The Condition Browser displays objects that come predefined in the dictionary. It also displays objects that you assert into the dictionary while you create your rule.

   The browser filters the items it displays according to where you access it. This behavior helps make sure you only select objects that make sense in the context where you access the browser. For example, here's the dictionaries that the browser displays when you create a start-after condition.



2. Expand the DooSeededOrchestrationRules dictionary.

   The browser displays the objects that this dictionary contains, including facts, variables, and so on.

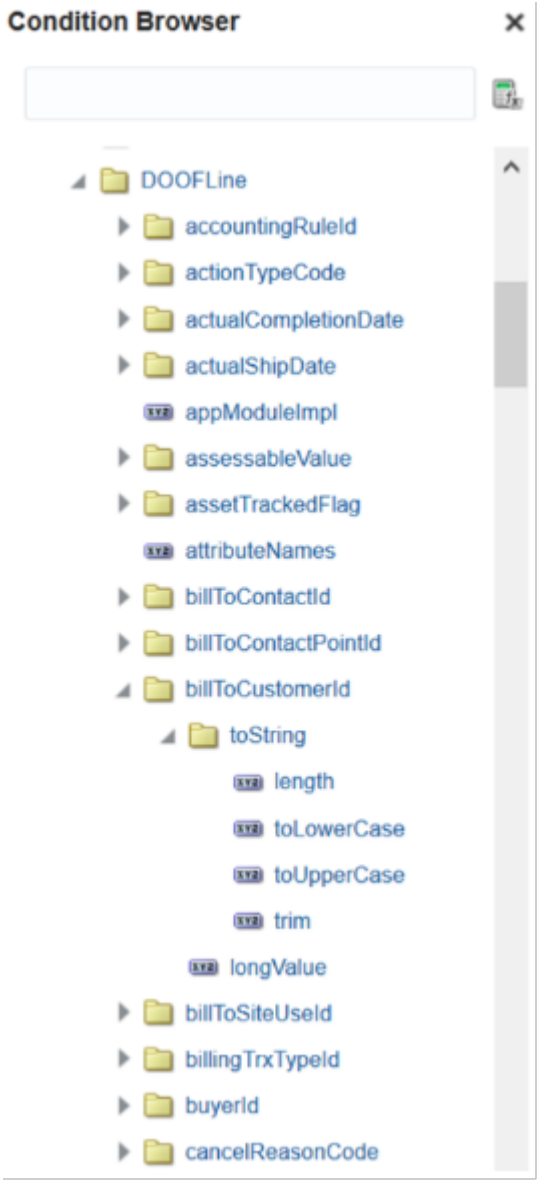


   You usually use only some of the objects that are available in the dictionary. For example, here are the objects you typically use in the DooSeededOrchestrationRules dictionary.

   - Result
   - Date
   - `Timestamp`
   - DOOHeader
   - SacResult
   - DOOFLine

   Get assistance from Oracle to use other objects or other dictionaries.

3. In DooSeededOrchestrationRules, expand DOOFLine, then notice the attributes. These attributes display on the order fulfillment line in the Order Management work area.

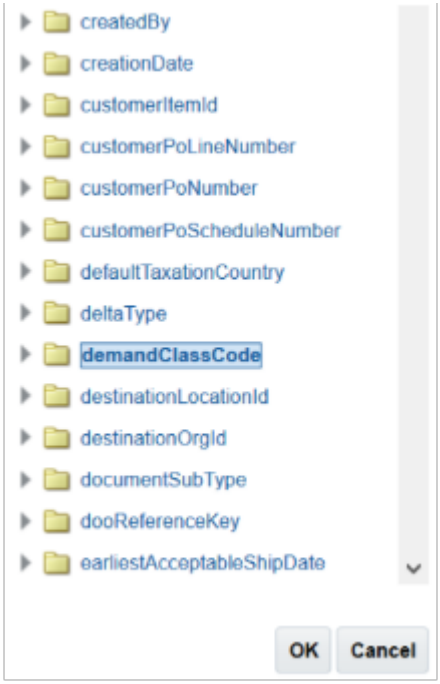


where

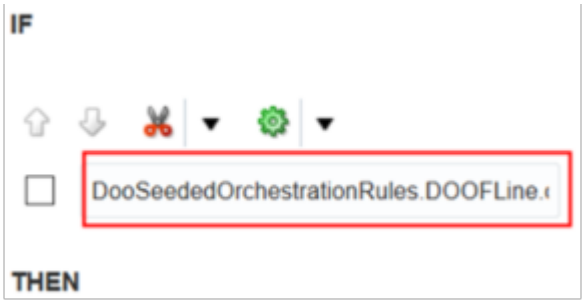
Browser Text	Description
DOOFLine	Fact in the DooSeededOrchestrationRules dictionary
DOO	Abbreviation for distributed order orchestration, which is an earlier name for order orchestration.

Browser Text	Description
FLine	Abbreviation for fulfillment line.
billToCustomerId	Fulfillment line attribute. This attribute stores an identification number for the customer. Its the same number that displays in the Order Management work area.
toString	Includes conversions you can make on the attribute.

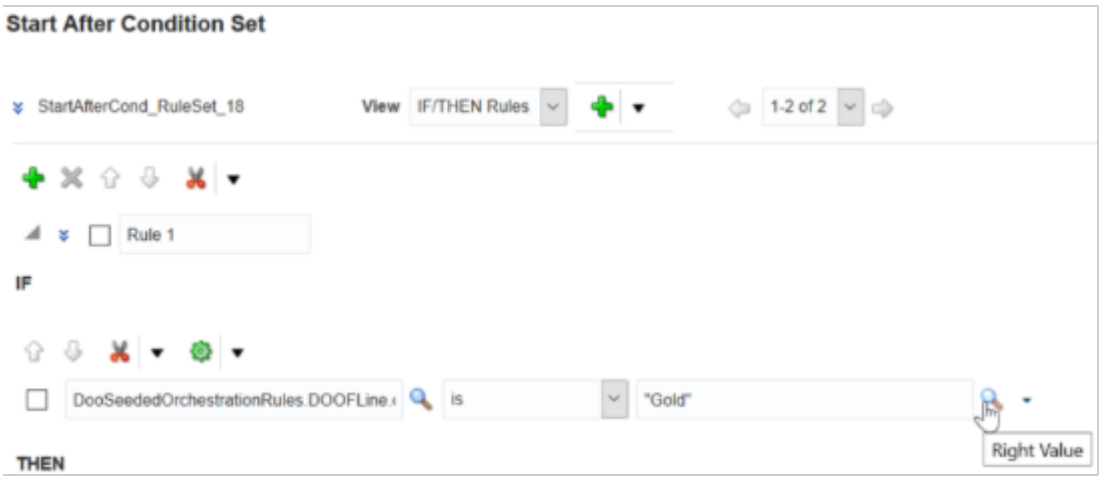
A sales order uses the demandClassCode attribute to allow you to classify the customer. Assume your rule must examine the value of demandClassCode, so scroll down, click **demandClassCode**, then click **OK**.



The rule editor adds the attribute to the statement.

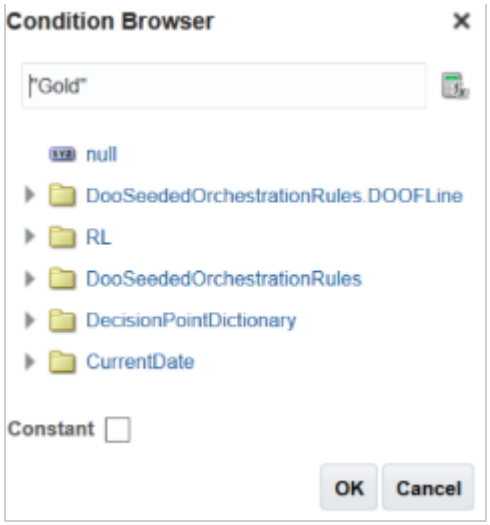


4. Click **Right Value** (the magnifying glass) to access the Condition Browser.



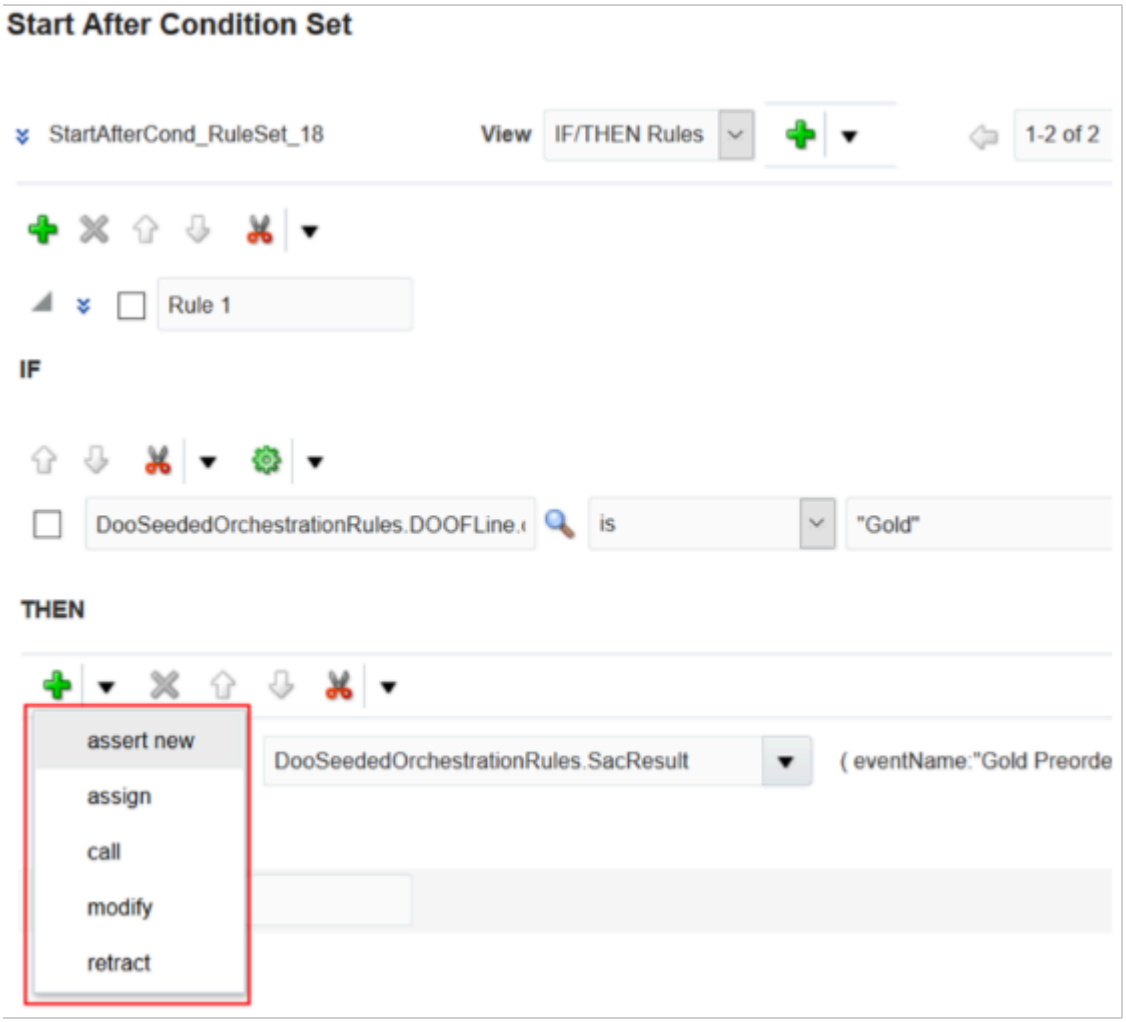
The Right Value window specifies the value that the If statement uses to determine whether its True. Similar to Left Value, you can expand DooSeededOrchestrationRules to access objects in the dictionary. You can also specify a string.

For this example, enter `"GOLD"` in the window, then click **OK**. You must enclose the string with double quotation marks.



Add the Then Statement

1. In the Then area, click **Add Action > Asset New**.

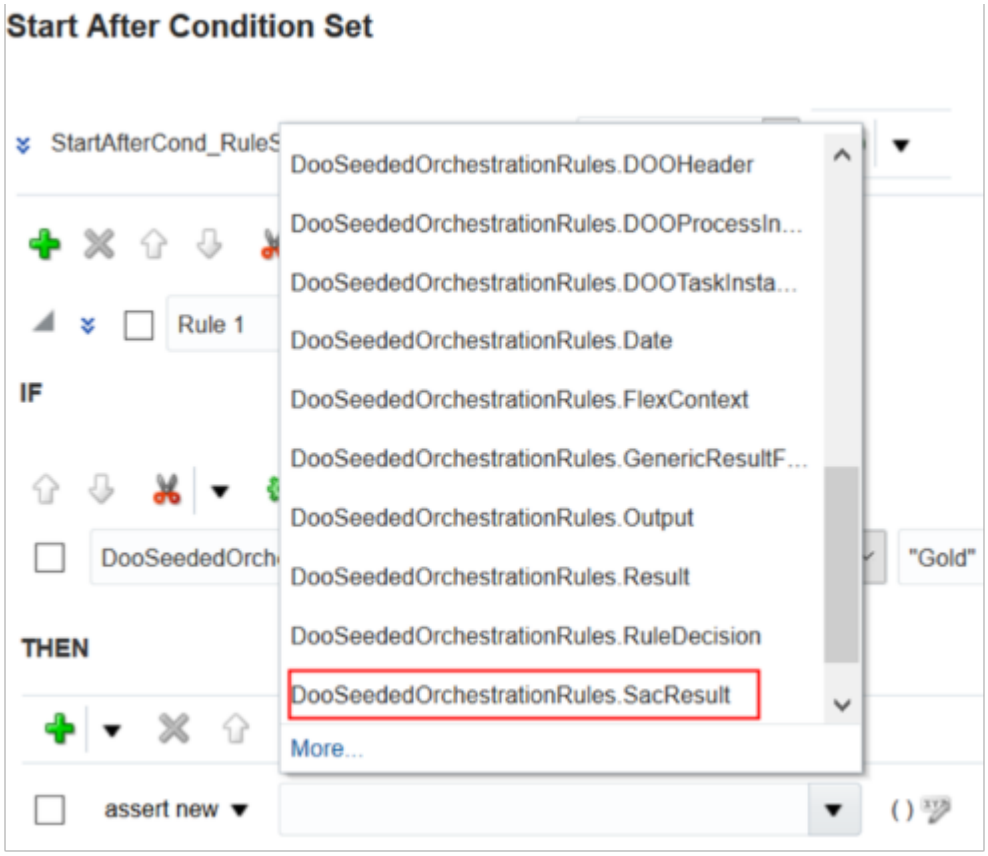


Select an action. For this example, you will assert the Then statement as a new fact into the dictionary.

Action	Description
Assert	Add a fact into the dictionary. You can also reassert an object. A reassert updates the dictionary so it reflects the new state of the object whether or not you changed the object. Reassert doesn't create a fact. If your rule requires more than one fact for a fact type, then you must assert each of these facts as a separate object.
Assert New	Add a new fact into the dictionary.

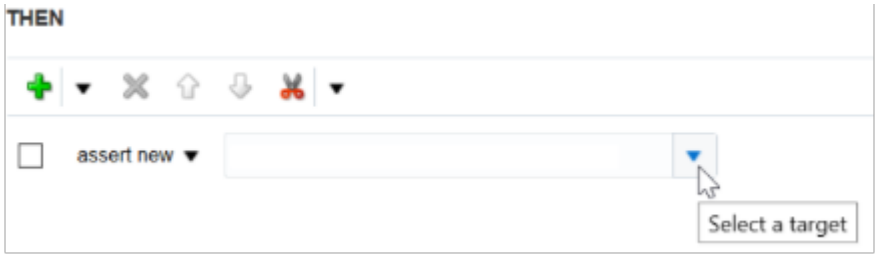
Action	Description
Assign	Assign a value to a variable in the dictionary.
Call	Call a function from the dictionary.
Modify	Modify a data value that's associated with a matched fact in the dictionary.
Retract	Remove a fact from the dictionary.
Other Actions	Advanced mode displays more actions, but you don't use most of them. For details, see the How to Use Advanced Mode Action Forms section in *User's Guide for Oracle Business Rules*.

2. Click **Select a Target**.

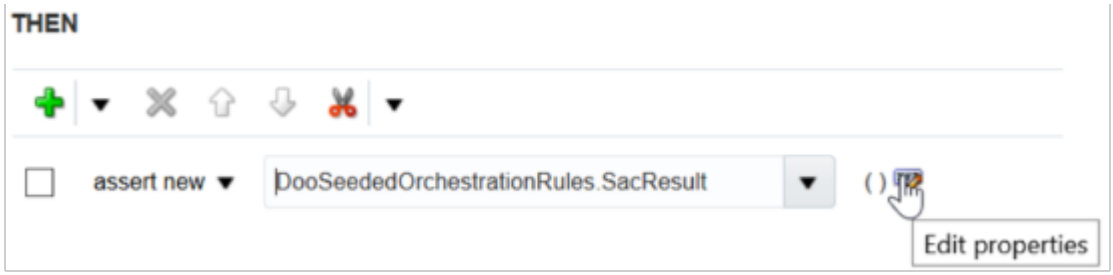


The target specifies the object this rule will modify if it evaluates to true.

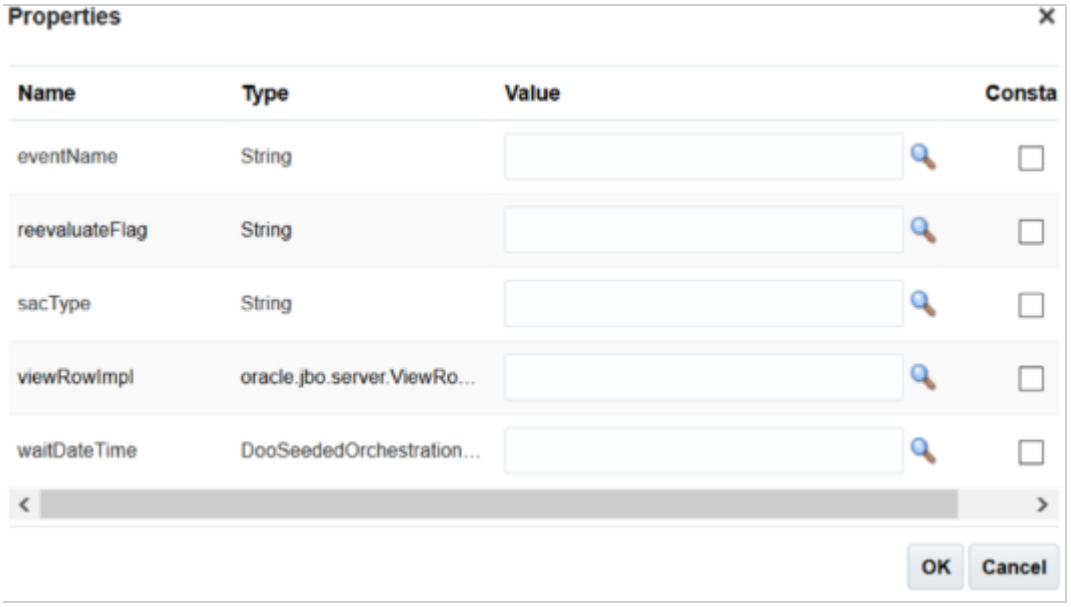
For this example, select object SacResult, where **sac** means **start after condition**. The rule will modify this value.



3. Click **Edit Properties**.



If you click Edit Properties, the rules editor gets the properties of the object that you selected as the target, then displays them in the Properties dialog. In this example, it displays the properties of SacResult.

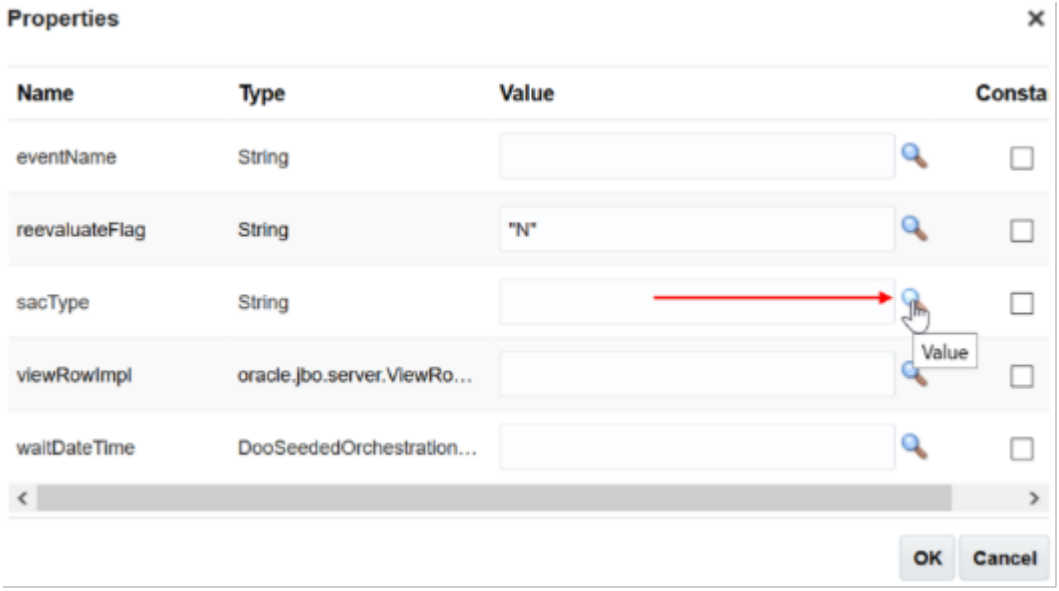


4. In the reevaluateFlag row, in the Value column, enter `"N"`. You must include the double quotation marks.

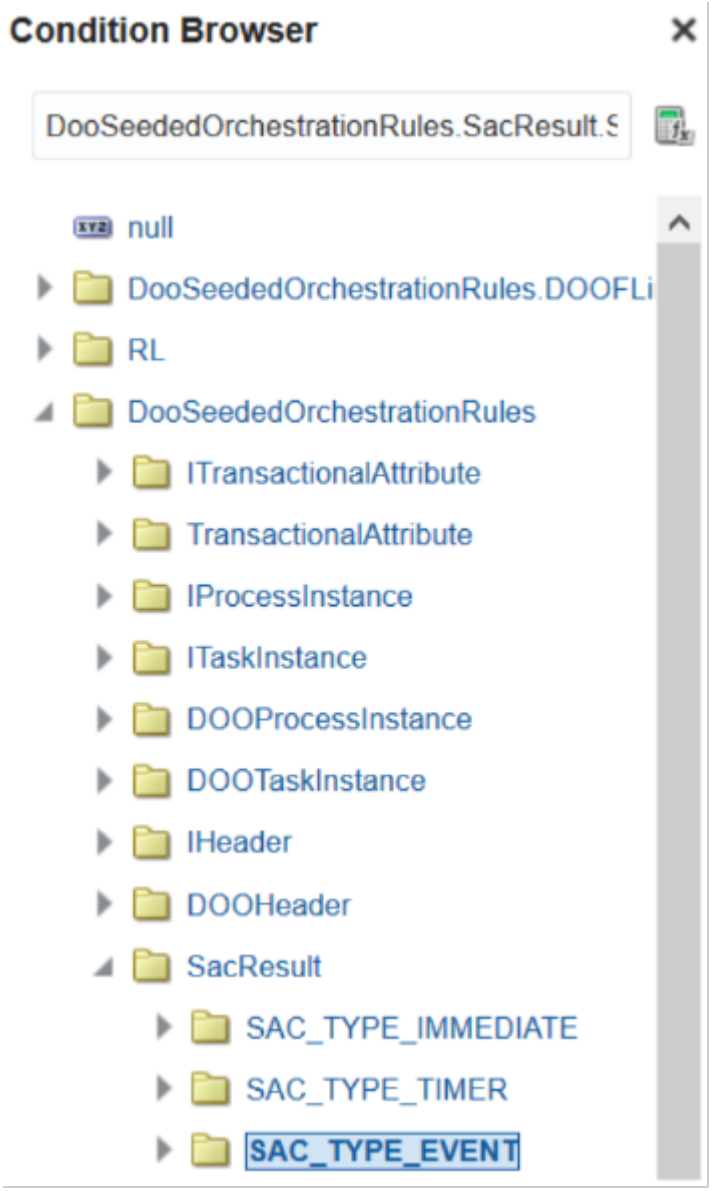
   Each target contains a different set of properties, so you must set them differently depending on your business requirements. In this example, reevaluateFlag specifies to evaluate the result of the start after condition again after the rule runs. You don't want this reevaluation, so set reevaluateFlag to N.
5. In the Value column, click **Value** (the magnifying glass).

   Use the sacType property to specify the action to take when the If statement evaluates to true. In this example, you must pause the orchestration process while it waits for an event to happen. The EVENT pause type causes

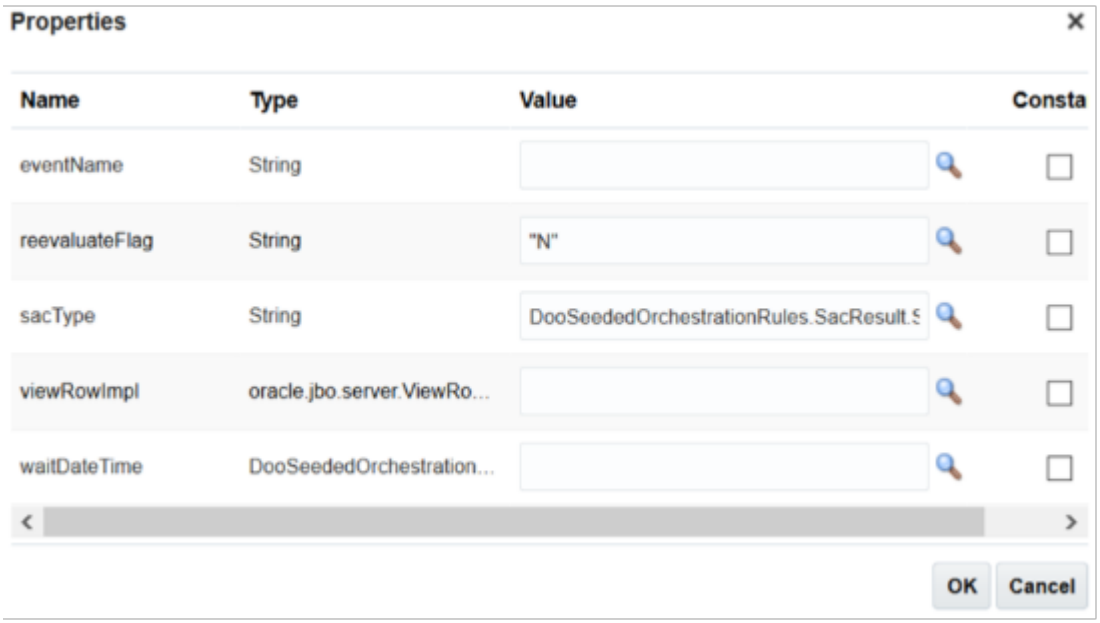
the orchestration process to pause, so you set sacType. Learn how to set properties for the start after condition. For details, see *Pause Orchestration Processes Until Events Happen*.



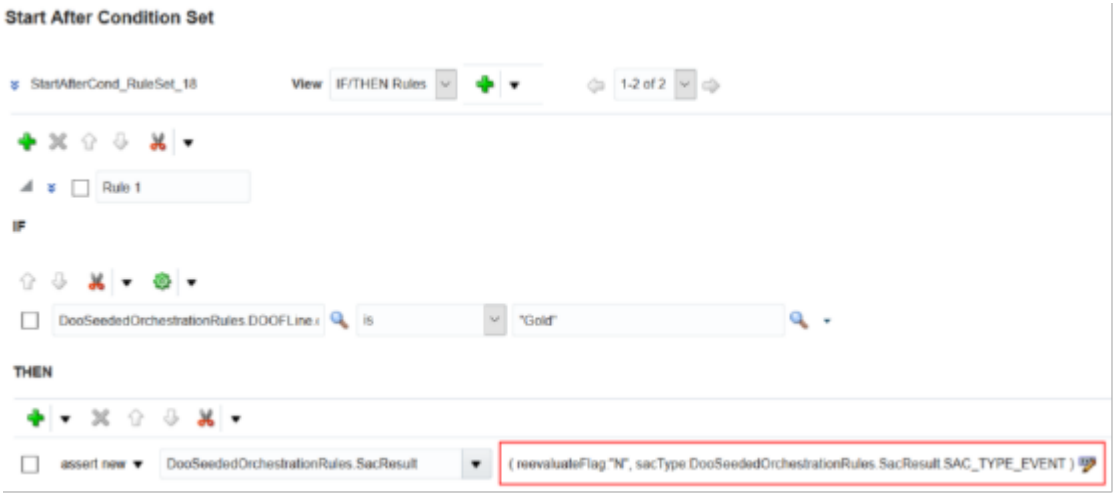
6. In the Condition Browser, expand **DooSeededOrchestrationRules > SacResult**, click **SAC_TYPE_EVENT**, then click **OK**.



7. Verify that the Properties dialog displays your settings, then click **OK**.



Notice that the Then area displays a string representation of the properties you set. Use this string to quickly assess the action the rule will do. Here's the complete business rule.



Related Topics

- Use Case to Pause for an Event
- Use Holds to Stop Orchestration Processes

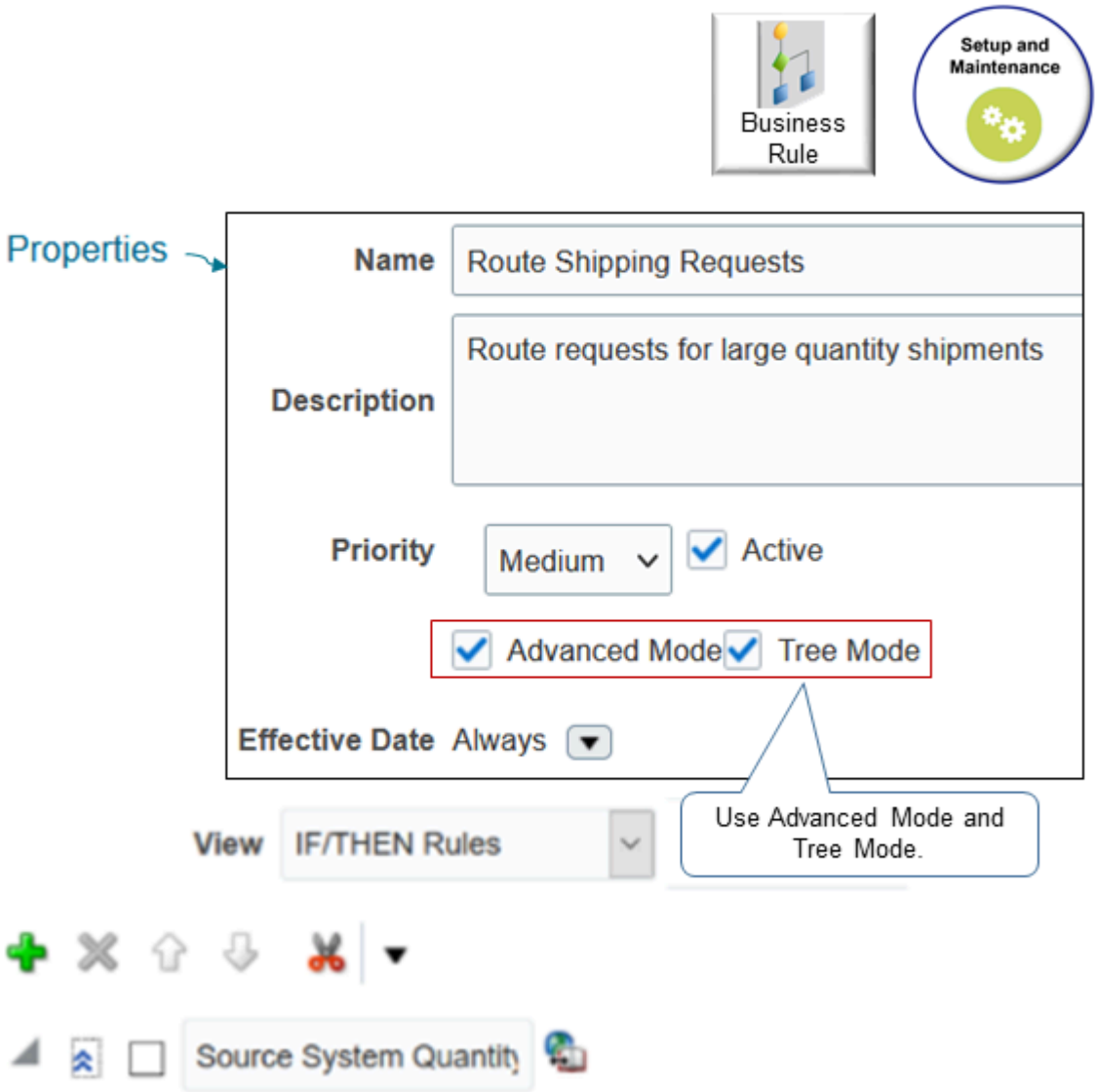
Use Advanced Mode and Tree Mode in Business Rules

Use advanced mode in a business rule to set priority, set variables, define collections, and so on.

- Set the rule priority when you must process more than one rule.
- Set the variable name for an object so the rule can compare two instances of an object or so you can create a hierarchical relationship.
- Set up collections of objects in one business rule.
- Use complex rule logic, such as If Then Else, or more than one If statement.
- Use Tree Mode to maintain the object hierarchy.

Use Advanced Mode

Here's the top part of a rule that uses Advanced Mode.



Here are the attributes that are available in the rule header when you use Advanced Mode.

Attribute	Description
Effective Date	Determines when the business rule is available to use.
Priority	Determines when to run each rule when you create a set of rules. • You use Priority only rarely. • Priority isn't an absolute value because Order Management might run a rule more than one time in a single session. Also, input values to the rule might change, and the sequence that Order Management uses to activate them might also affect priority. • You can't use Priority across more than one set of rules.
Active	Makes the business rule available for use. If Active doesn't contain a check mark, then Order Management doesn't include it during validation, even if Order Management previously released it as an active rule.

Examine examples that use Advanced Mode. For details, see *Get Data from Product Information Management*.

Use Tree Mode

A business rule uses facts to evaluate rules at run time.

- The business rule uses the data model from Order Management to identify each fact it must use.
- A fact doesn't include data that resides outside of Order Management.
- Use facts for transformation, process assignment, and routing rules according to business components in Oracle Application Development Framework (ADF), specifically view objects (vo) that expose the data model.
- View objects provide a hierarchical view of transactional data, such as a sales order. However, each business rule converts them into rule language (rl) facts. Facts in rule language don't include a hierarchy. You don't need to manage rule language facts, but this conversion might affect how you create your rule.
- If the fact is a business component in Application Development Framework, and if your rule uses more than one object type, then you must make sure your rule reestablishes the view object (vo) hierarchy. Reestablishing the hierarchy makes sure each rule performs at an optimal level.

Properties of Business Component Facts in the Application Development Framework

Property	Description
ViewRowImpl	References a row in a view object. You use ViewRowImpl to access a row in a database table. You typically specify this property in the result. Its required in a transformation rule. Its optional for other business rules. The Business Components for Java framework instantiates an object of ViewRowImpl for each record that the view object query returns for the row.
key_values	References the oracle.rules.sdk2.decisionpoint.KeyChain object. You can use this property to get the set of key values for this row and the parent rows of this row. This property is optional.

You can use Advanced Mode with an explicit join to reestablish the hierarchy. You can also use Tree Mode to more clearly represent the hierarchy in the rule editor.

Maintain the Object Hierarchy

You must maintain the object hierarchy when you use advanced mode or tree mode.

Advanced Mode	Advanced Mode and Tree Mode
If you use only Advanced Mode, and if the If statements and Then statements in your rule will process a public view object on the sales order, then your rule must make sure it joins each public view object so it accurately represents the sales order hierarchy. For example: • Join the header public view object with the line public view object. • Join the line public view object with the fulfillment line in the public view object. • And so on.	If you use Advanced Mode and Tree Mode, then you can select objects in the existing hierarchy, such as Header, Line, and FulfillLine. If you use an extensible flexfield, then you might need to create more joins.

Note

- If you use Tree Mode, then also use Advanced Mode because Advanced Mode helps you to maintain the hierarchy.
- If you use only Advanced Mode to maintain the object hierarchy, then you must explicitly specify the hierarchy. View an example. For details, see *Get Data from Product Information Management*.
- If you use Tree Mode to set the root of an object hierarchy, then the rules editor displays objects only at the root level or below the root level while you create the rule. This filtering helps to make sure you correctly maintain the object hierarchy.
- Tree Mode uses the forward slash (/) to indicate hierarchy.

View an example that uses Tree Mode to maintain the object hierarchy and that uses the forward slash. For details, see *Use Extensible Flexfields in Line-Selection Rules*.

Related Topics

- Use Extensible Flexfields in Line-Selection Rules
- Get Data from Product Information Management

Use Decision Tables and Bucket Sets in Business Rules

Use a decision table to create a set of IF statements and display them as a table.

If you must analyze more than one combination of attribute values, then the decision table is more compact and intuitive to use than a number of individual IF statements. The decision table doesn't require you to use all If statements and Then statements in every rule.

Note that you can't add a bucket set to a fact.

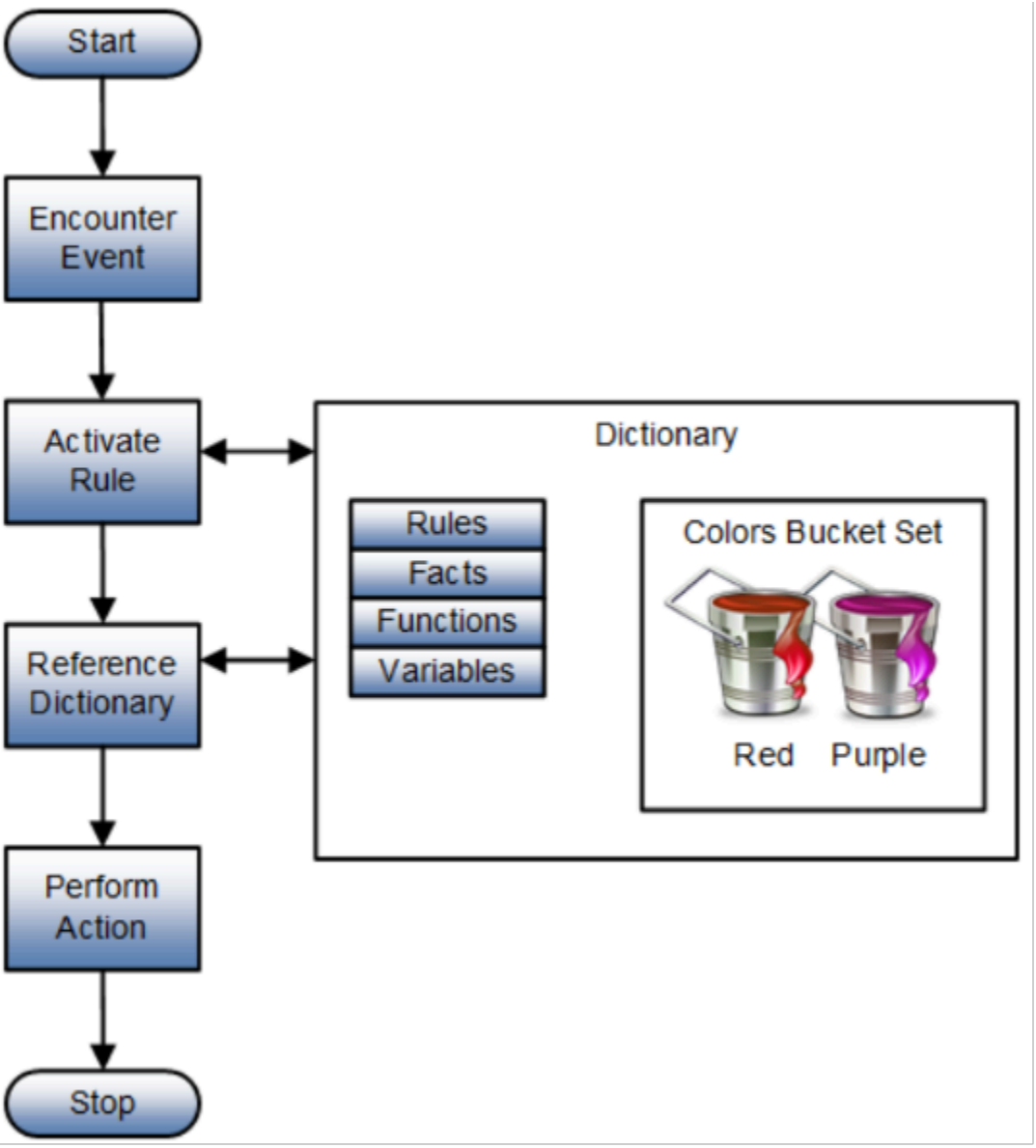
Examine some examples that use a decision table and bucket set. For details, see *Create Transformation Rules* and *Assign Orchestration Processes*

Bucket Sets

A **bucket** is a container that sets up a list of values or a range of values that the If statement uses to determine whether its true.

A **bucket set** is a container that you can use to hold the overall range of values that a group of buckets defines. Use it to constrain the values that the If statement and the facts in a decision table will consider.

Assume you create a Colors bucket set that includes string values of `red` and `purple`. You can write a rule for each color. You can also write one rule for blue and another rule for purple.

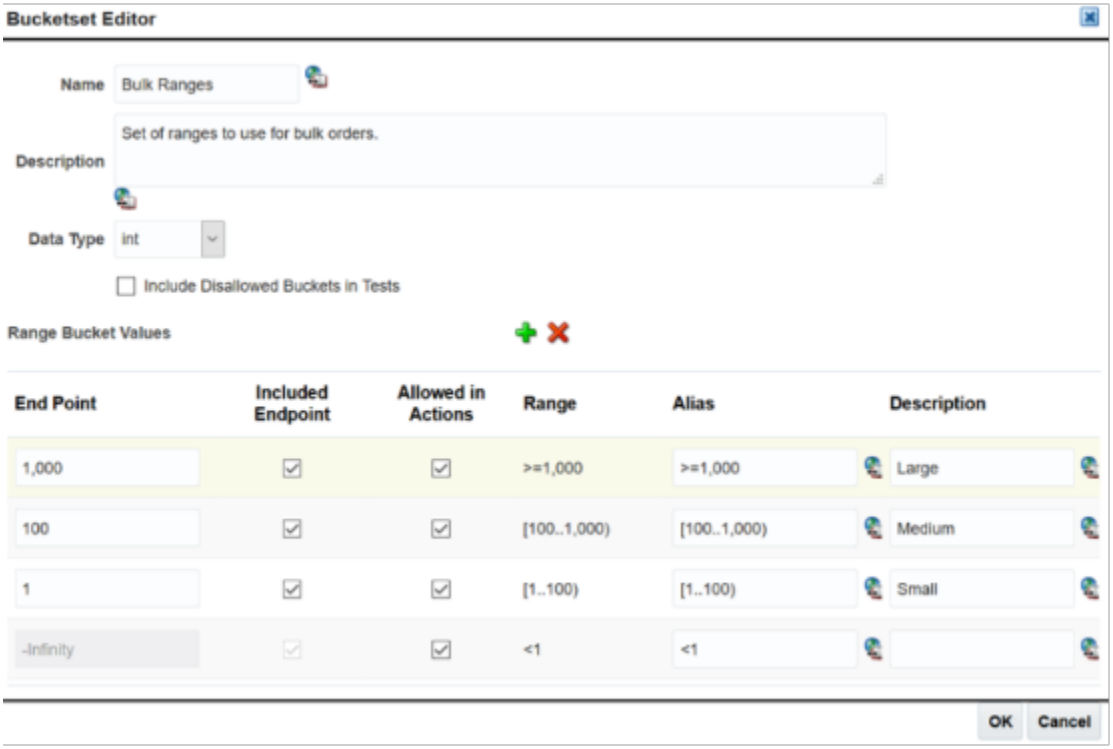


Or assume you create a Numbers bucket set that uses integers instead of strings. You can create buckets in the Numbers bucket set. Here are the buckets.

- Less than 1
- 1 to 10
- Greater than 10

You can create a bucket set that specifies aliases for data that might be difficult to recognize, such as identification numbers. You can use the toString function in the If statement to convert the value from the Long data type to a String alias, such as a product code. The bucket set will map each identification number to the product code.

Here's an example bucket set that groups ranges of values into a large, a medium, or a small bucket.



Use Predefined Bucket Sets

Here they are.

- ORACancelReason
- ORACarrier
- ORAConversionType
- ORACurrencies
- ORADemandClasses
- ORAFOB
- ORAFreightTerms
- ORAInvoicingRules
- ORAModeOfTransport

- ORAPaymentTerms
- ORAReceiptMethods
- ORAReturnReason
- ORAReturnReason
- ORASalesCreditTypes
- ORAServiceLevel
- ORAShipmentPriority
- ORASupplier
- ORASupplierSite
- ORATaxClassificationCodes
- ORATaxExemptionReason
- ORAUOM
- ORAWarehouses
- RAAccountingRules

Add Data to Your Bucket Sets

Run a scheduled process to populate your bucket sets with data.

Assume you need to populate your bucket set only with data that's related to your Computer Service and Rentals customer.

1. Go to the Scheduled Processes work area.
2. Click **Schedule New Process**, then search for the Generate Bucket Sets scheduled process.
3. In the dialog that displays, set the value.

Parameter	Value
Bucket Set Name	You must leave this parameter empty. It is no longer supported.
View Object Name	You must leave this parameter empty. It is no longer supported.
Where Clause	You must leave this parameter empty. It is no longer supported.
Refresh Collected Data	Set a value. o **Yes**. Keep your bucket set up to date with data in the entities that the bucket set references. o **No**. Don't use this value.

4. Click **Submit**.

Related Topics

- Create Transformation Rules
- Assign Orchestration Processes
- Guidelines for Using Extensions to Get Data from Oracle Applications

Calculate Dates in Business Rules

You can calculate a date in a business rule.

- A business rule doesn't come predefined with functions that do date arithmetic, such as **`Scheduled Ship Date minus 1`**.
- To avoid data type and object hierarchy problems, you can't use a business rule to do date arithmetic with a fact. However, you can create a rule that does date arithmetic in an orchestration process.
- A date attribute use the time stamp data type.

Here's an example that uses an If statement to set up **`fline`** as the alias for the fulfillment line. It compares the time of the scheduled ship date to the current date, and uses milliseconds in the comparison conversion.

```
If fline.scheduleShipDate.time>=CurrentDate.date.timeInMillis
```

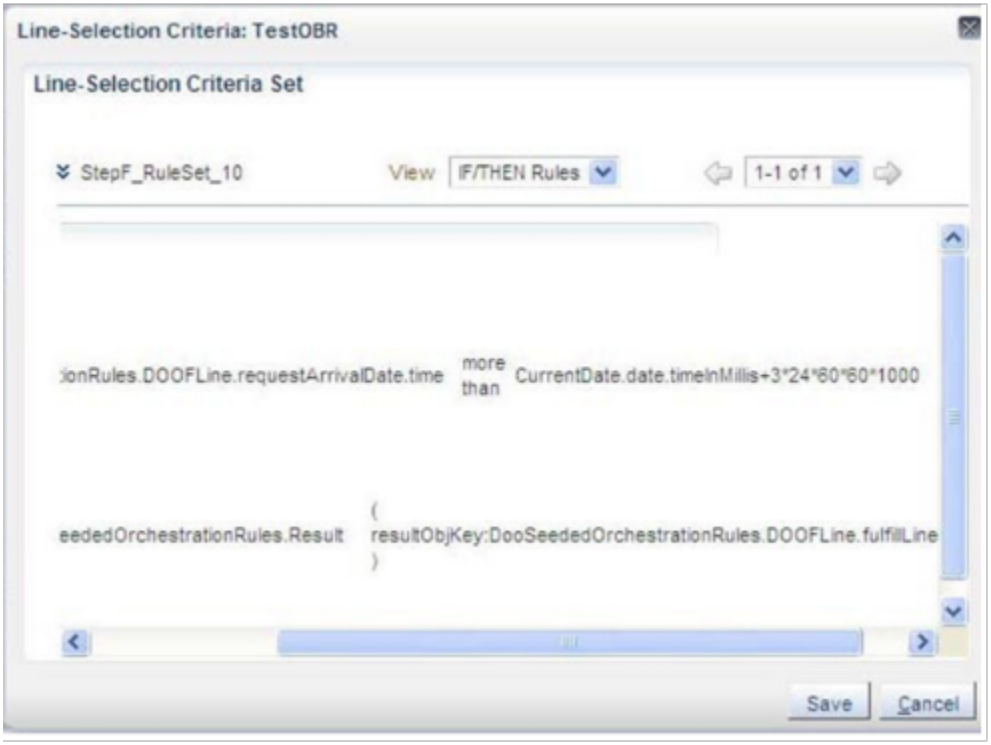
In another example, here's some code that calculates the scheduled arrival date to happen three days after the current date.

```
If fline.scheduleArivalDate.time>= CurrentDate.date.timeInMillis+3*24*60*60*1000
```

where

- 3 is the number of days.
- 24 is the number of hours in one day.
- 60 is the number of minutes in one hour.
- 60 is the number of seconds in one hour.
- 1000 is the number of milliseconds in one second.

Here's a dialog that sets up a business rule for the line selection criteria in an orchestration process.



Note

Rule Part	Code
If the requested arrival date happens earlier than three days after the current date, then skip this step.	`DooSeededOrchestrationRules.DOOFLine.requestArrivalDate.time more than CurrentDate.date.timeInMillis+3*24*60*60*1000`
Action	Assert New `DooSeededOrchestrationRules.Result(DooSeededOrchestrationRules.DOOFLine.fulfillLineId`

If you use this kind of calculation in the THEN clause, then make sure your rule doesn't modify or overwrite the original value. You might need to assign a temporary attribute to hold the calculation before your rule populates the result object.

Create Temporary Attributes

Use a Timestamp object to create a temporary attribute, such as to provide an offset to a data attribute. For example, **three days before the scheduled ship date**. The Assign New action creates a new instance of the attribute, provides an alias, and sets the initial value.

Here's an example of a lead time expression in a business rule on an orchestration process. `fline` represents the fulfillment line object and DateTime is the temporary attribute.

Action	Code
Assign New	`DooSeededOrchestrationRules.Timestamp DateTime = fline.scheduleShipDate`
Modify	`DateTime(time:DateTime.time-(2*24*60*60*1000))`
Assert New	`DooSeededOrchestrationRules.Result(resultObj:DateTime)`

Use Date Functions in Advanced Mode

Function	Example
`before(Timestamp|java.util.Date)`	`If fline.RequestArrivalDate.before(fline.ScheduleArrivalDate) is true`
`after(Timestamp|java.util.Date)`	`If fline.RequestArrivalDate.after(fline.ScheduleArrivalDate) is true`
`compareTo(Object|Timestamp|java.util.Date)`	`If fline.RequestArrivalDate.compareTo(fline.ScheduleArrivalDate) is -1`
`equals(Object|Timestamp)`	`If fline.RequestArrivalDate.equals(fline.ScheduleArrivalDate) is true`

Note

- Use these functions for each attribute that's a Timestamp data type, including a temporary attribute that you declare with the Timestamp object.
- You can manually enter the function after you select the attribute. You can also open the Expression Builder, click the Functions tab, then open the folder next to the date attribute.
- The format for each function, except compareTo, is true or false, and this format doesn't require quotation marks.
- Here's the format that compareTo uses.
    - `-1`. The object to the left of compareTo happens before the object to the right of compareTo.
    - `0`. The object to the left of compareTo happens at the same time as the object to the right of compareTo.
    - **1**. The object to the left of compareTo happens after the object to the right of compareTo.

Manage Errors and Conflicts in Business Rules

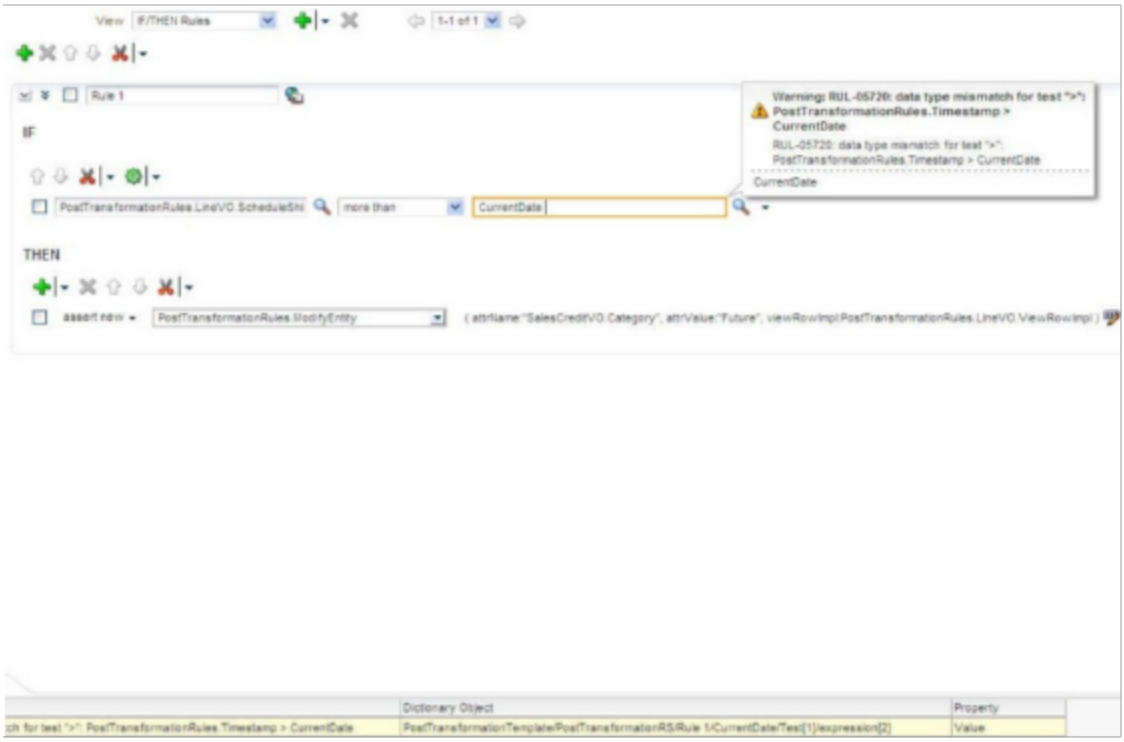
Manage errors and conflicts that might happen with your business rule.

Manage Errors

Each business rule validates your syntax when you save your rule. It displays errors in the error pane at the bottom of the page or dialog.

- Click Validate to validate your rule before you save it.
- Click the arrow on the bottom right corner of the page or dialog to examine the error.
- Double-click an error message to open the rule that contains the error and to automatically highlight the object that the error affects.

For example:



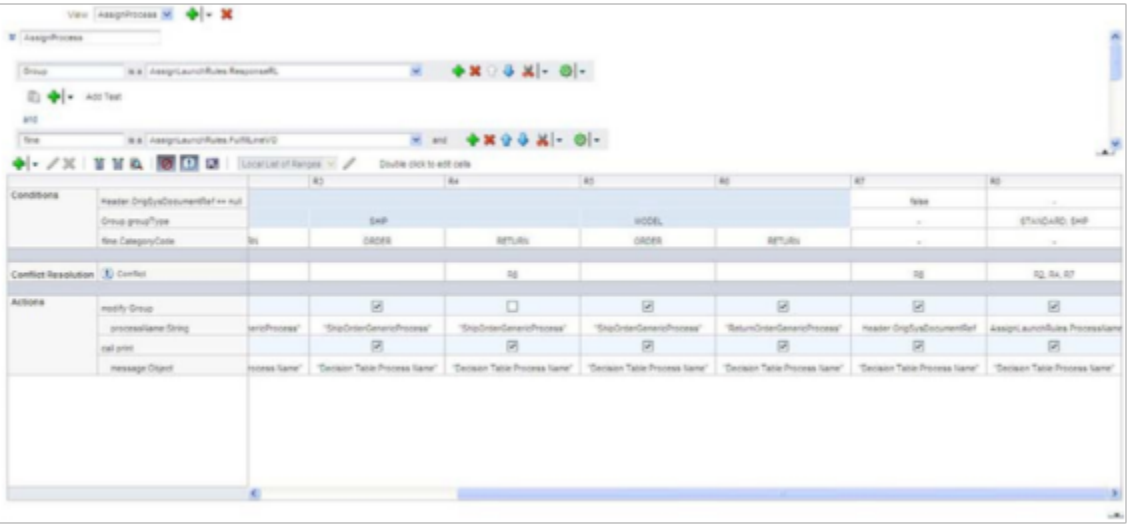
Manage Conflicts

Each business rule provides conflict management for your decision tables.

- The business rule displays conflicts between the If statement and the Then statement.
- You can use the exclamation point (!) on the tool bar to toggle visibility of the conflict.
- The business rule identifies each rule that the conflict affects, including rules that depend on a conflicted rule. For example, if rule A is conflicted, and if Rule B depends on Rule A, then Rule B also displays as conflicted.
- An icon on the decision table toolbar identifies a missing set up in your business rule. For example, if you add bucket set elements but don't use them in your rule, then the page displays an icon.
- The page might display the rules you must add to correct the set up. You can select a rule to add it to your set up, but you aren't required to select them. Instead, you can create a different rule to correct the set up.

  This feature is useful when you create a bucket set that includes a large number of values. Each bucket set allows you to select Otherwise when it isn't necessary to individually specify each selection.

Here's an example where a new rule is in conflict with three other rules.



Functions You Can Use in Business Rules

Use a function in your business rule to get details or to do a calculation.

Here are the functions that you can use in a product transformation, orchestration process assignment, or routing rule. You can't use them on an orchestration process.

Attribute	Description
ModifyEntity	Modify an entity. Use this function with a pretransformation rule, transformation rule, or posttransformation rule. Use these arguments. • attrName(String) • attrValue(Object) • viewRowImpl(oracle.jbo.server.ViewRowImpl) Note • If you don't use the list of values to set attrName(), then you must enclose the attribute that you provide with quotation marks. • The name of the attribute is the required data type, not the value of the attribute at run time. • Make sure your rule populates the ModifyEntity arguments when your rule uses them to do an action. • ModifyEntity behavior is different from the Modify action you use on menus on the Manage Product Transformation Rules page.
AddNewOrderLine	Add an order line to a sales order. Use this function only with a transformation rule.

Attribute	Description
	Use these arguments. • newItemId(Long) • viewRowImpl(oracle.jbo.server.ViewRowImpl)
DeleteOrderLine	Delete an order line from a sales order. Use this function only with a transformation rule. Arguments. • fulfillmentLineId(Long) • viewRowImpl(oracle.jbo.server.ViewRowImpl) For example: • Delete fulfillment line 1 and fulfillment line 2

Note that viewRowImpl(oracle.jbo.server.ViewRowImpl) represents the current object instance. For example, a fulfillment line record.

Functions You Can Use with Rules On an Orchestration Process

Use these functions in a rule that you create on an orchestration process. You can't use them with a product transformation, orchestration process assignment, or routing rule.

These functions work at the attribute level. You can find them on the Functions tab of the Expression Builder dialog as the first entries under each object.

You can also use predefined date functions on each date attribute.

Date Functions You Can Use On an Orchestration Process

Function	Description
getAdjustedDate(Timestamp, Double)	Add or subtract the value of argument Double (in days) from the Timestamp argument, depending on the sign, and then return the result in the Timestamp argument.
subtractFromDate(Timestamp, Timestamp)	Subtract the value of the second Timestamp argument from the first Timestamp argument. If the second Timestamp argument is empty, then subtractFromDate subtracts the current date from the first Timestamp argument. subtractFromDate returns the number of days as a BigDecimal() value.
getConvertedDate(Object)	Convert a type of date object in the datetime data type.
current_date	Get the system date as a datetime data type.

Change Management Functions You Can Use On an Orchestration Process

Function	Description
hasChanges()	Determine whether extensible flexfield values in the change order are different from values that the previous version of the sales order contains.
getAttribute(String)	Get the name of an attribute. Use getAttribute on the order header, fulfillment line, orchestration process, or task.
getTransactionalAttribute(String)	Get the name of a transactional attribute. You can use getTransactionalAttribute only on a fulfillment line.
getFlexContext(String)	Gets the name of the extensible flexfield context. Use getFlexContext only on a fulfillment line.

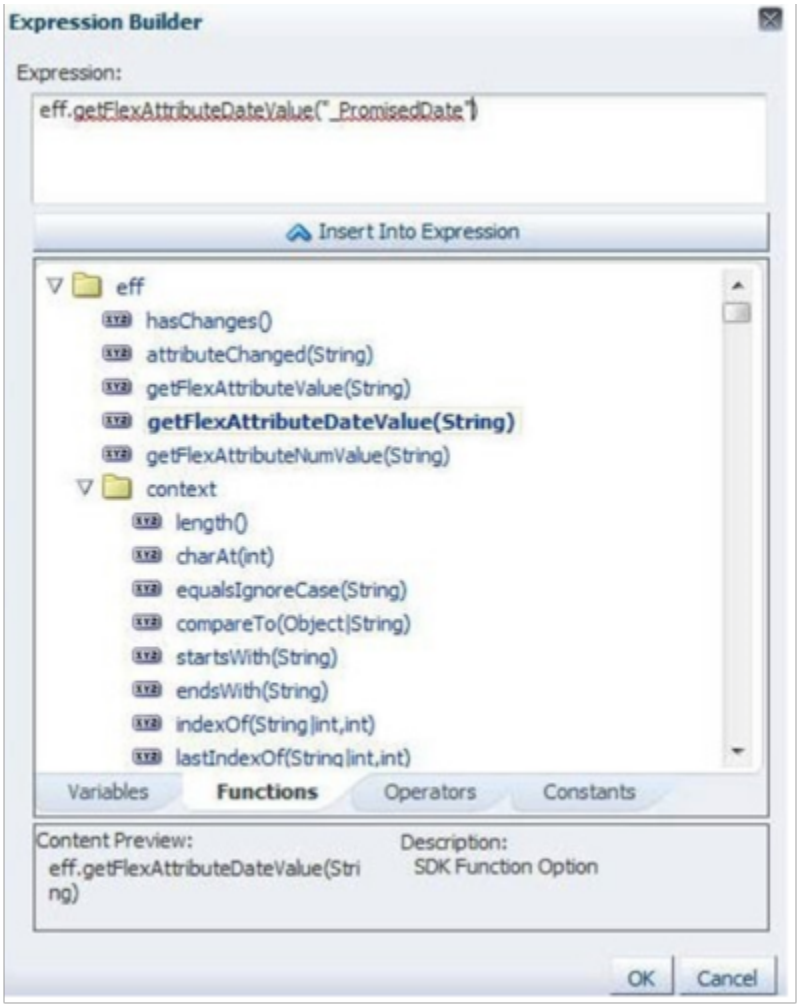
You can also use these functions. You must prefix the attribute name with an underscore (_), and you must enclose the attribute name in quotation marks.

Function	Description
attributeChanged(String)	Determine whether the value of an attribute in the change order is different from the value that the previous version of the sales order contains. Use it on the order header, fulfillment line, or extensible flexfield context.
getFlexAttributeValue(String)	Get the value of an extensible flexfield attribute of data type String. Use it only on an extensible flexfield context.
getFlexAttributeDateValue(String)	Gets the value of an extensible flexfield attribute of data type Date in time stamp format. Use it only on an extensible flexfield context.
getFlexAttributeNumValue(String)	Get the value of an extensible flexfield attribute of data type Number. Use it only on an extensible flexfield context.

For details, see *Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications*.

Example of Choosing a Function

Use the Functions tab in the Expression Builder dialog to get a list of arguments you can use. For example, with method getFlexAttributeDateValue.



Visual Information Builder

Use Visual Information Builder

Use a drag-and-drop interface to create a business rule. Visualize data, visualize your business processes, and implement business logic.

Examine the Behavior of a Predefined Rule

Observe how the rule editor establishes a condition.

- If task type is shipment, then set the connector to Connect to Oracle Shipping

Do it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Integration Routing Rules for Sales Orders
2. On the Manage External Interface Routing Rules page, notice the predefined routing rules.
   Use this page to.
    - Activate or inactivate a rule.
    - Create a new rule.
    - Change the rule priority.
    - View, modify, duplicate, or delete a rule.
    - Publish active rules.
3. Right-click **Shipping Task Routing**, then click **Actions > Edit**.
4. Click **IF**, then notice the phrase `Task Type is Equal to Shipment`.
5. Click **DO**, then notice the phrase `Set Connector Name, Connector Name: Connect to Oracle Shipping System`.
6. Click **Close**.

Limitations

- You can't use a decision table. If you created a rule in an earlier version of Order Management that uses a decision table, and if you prefer to use Visual Information Builder to replace this rule, then use the IF-THEN-ELSE structure in Visual Information Builder to replace the decision table.
- Don't use Visual Information Builder **and** Oracle Business Rules at the same time in the same implementation. Use Visual Information Builder **or** Oracle Business Rules for all your rules.
- You can't create a global variable for use in a rule.
- You can't create collections of objects in conditions.
- Don't reference an object that includes the text `_Obsolete` when you create your rule. `_Obsolete` indicates that business rules might no longer support the object in the current or future update.

Related Topics

- Manage Routing Rules
- Manage Pretransformation Rules

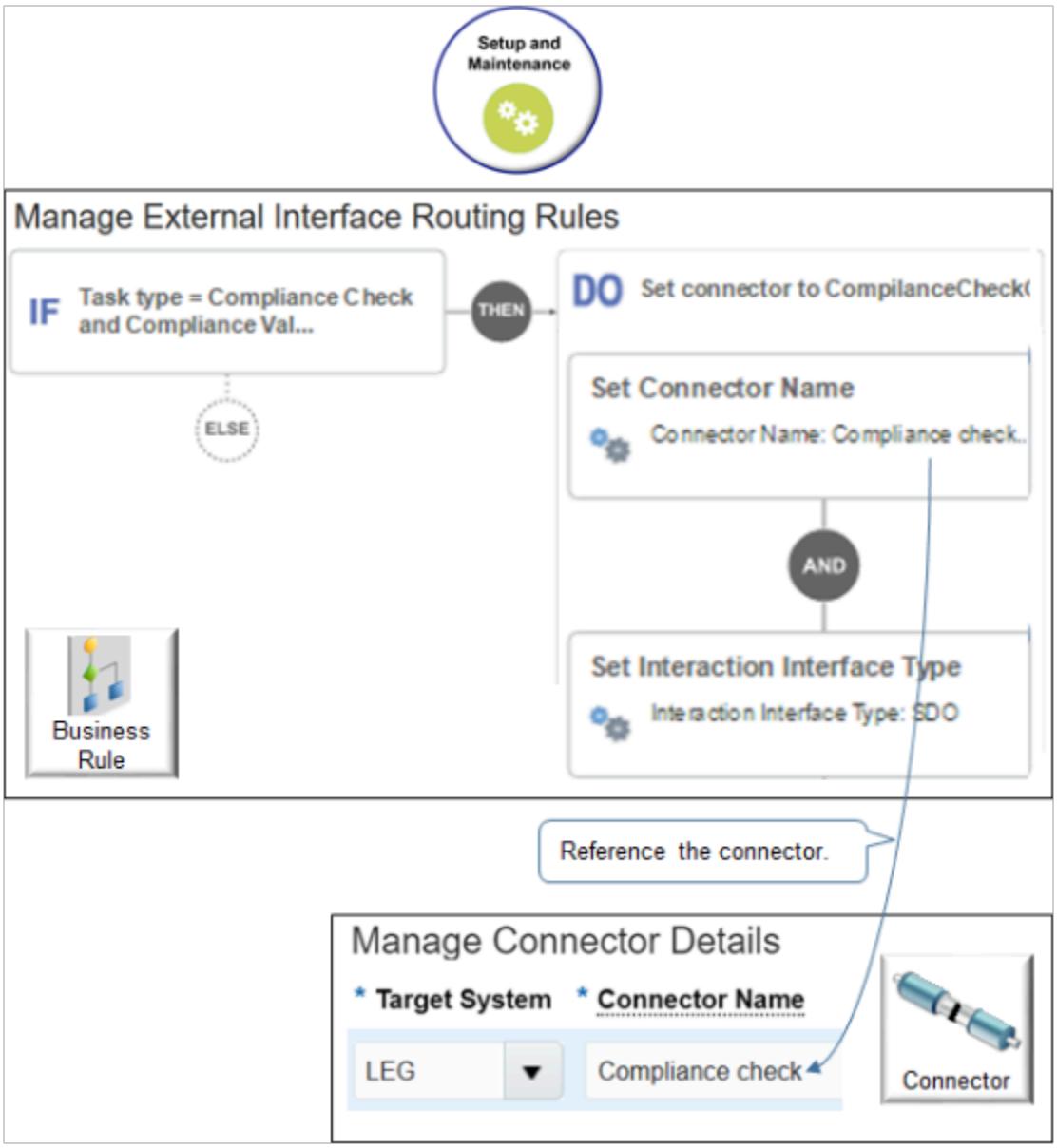
Manage Routing Rules

Use Visual Information Builder to create routing rules.

Assume you must create a routing rule.

- If the task type is Compliance Check, and if the compliance value is less than 100, then set the connector to ComplianceCheckConnector, and set the interface type to SDO.

Here's the rule that you will create.



Summary of the Steps

1. Create the If statement.
2. Create the Then statement.
3. Activate and publish your rule.

This topic uses example values. You might need different values, depending on your business requirements.

Create the If Statement

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Integration Routing Rules for Sales Orders
2. On the Manage External Interface Routing Rules page, click **Create New Rule**.
3. Set the values.

Attribute	Value
Name	Routing Rule for Compliance Check
Description	Route sales orders according to results of the compliance check.

4. Name the If statement so it describes the meaning of the condition.
    - In the If area, click **Enter Description**.
    - In the Enter Description dialog, enter `Task Type = Compliance Check and Compliance Value Is Less Than 100`, then click **OK**.
5. Click **New Condition** (the dashed circle in the IF area).
6. In the Create Condition dialog, enter `Task`, wait a moment, then click **Task Type (Order Header)**.

   Notice that the dialog displays suggestions when you wait after you finish typing.
7. In the Create Condition dialog, click **Search**.
8. In the Search dialog, enter `Compliance`, then click **Search**.
9. Select **Compliance Check**, click **OK > OK**.

   Notice that area IF displays the condition.
    - `Task Type is Equal to Compliance Check`
10. Click **AND**.
11. In the Create Condition dialog, enter `Compli`, then wait a moment.

    Notice that the dialog displays a number of attributes, such as **Compliance Date (Header Compliance Details)**. Extensible flexfields that you set up on the order header provide values for these attributes. The order header displays them as attributes in the Order Management work area.
12. Click **Compliance Value (Header Compliance Details)**.
13. Change the operator from **=** (equal), to **<** (less than).
14. Change the value from `0.0`, to `100`, then click **OK**.

    Notice that the IF area displays the condition.
    - `Task Type is equal to Compliance Check AND Compliance Value < 100`

Create the THEN Statement

1. Click **THEN > DO**.
2. Name the DO so it reflects the meaning of the action.
    - In the DO area, click **Enter Description**.
    - In the Enter Description dialog, enter `Set connector to ComplianceCheckConnector and interface type to SDO`, then click **OK**.
3. Click **New Action** (the dashed circle in the DO area), then click **Perform an Action**.
4. In the Create Action dialog, set the action to **Set Connector Name**, then click **Search**.
5. In the Search dialog, enter `%Compli%`, then click **Search**.

   The percentage symbols (%) are wildcards that search for all values before and after the text **Compli**. This search returns services that the Manage Connector Details page contains, and that contain the text **Compli**.
6. In the Description area, click **Compliance check system conn**, then click **OK**.
7. In the Create Action dialog, click **OK**.
8. In the DO area, click **AND > Perform an Action**.
9. In the Create Action dialog, set the action to **Set Interaction Interface Type**.
10. Set Interaction Interface Type to **SDO**.

    If you are a new customer, then you must set Interaction Interface Type to SDO for any interaction interface that you set up. Use EBM only for backward compatibility to a prior update.
11. Click **Save and Close**.

Activate and Publish Your Rule

1. On the Manage External Interface Routing Rules page, notice that the Active indicator for the Routing Rule for Compliance Check rule is grey.
2. Right-click **Routing Rule for Compliance Check**, then click **Actions > Edit**.
3. In the Routing Rule for Compliance Check dialog, add a check mark to the **Activate Rule** option, then click **Save and Close**.

   Notice that the Active indicator for Routing Rule for Compliance Check is green.
4. Click **Publish**.

   Order Management publishes each of the rules that are active on the Manage External Interface Routing Rules page.

Related Topics

- Use Visual Information Builder
- Route Requests from Order Management to Fulfillment Systems

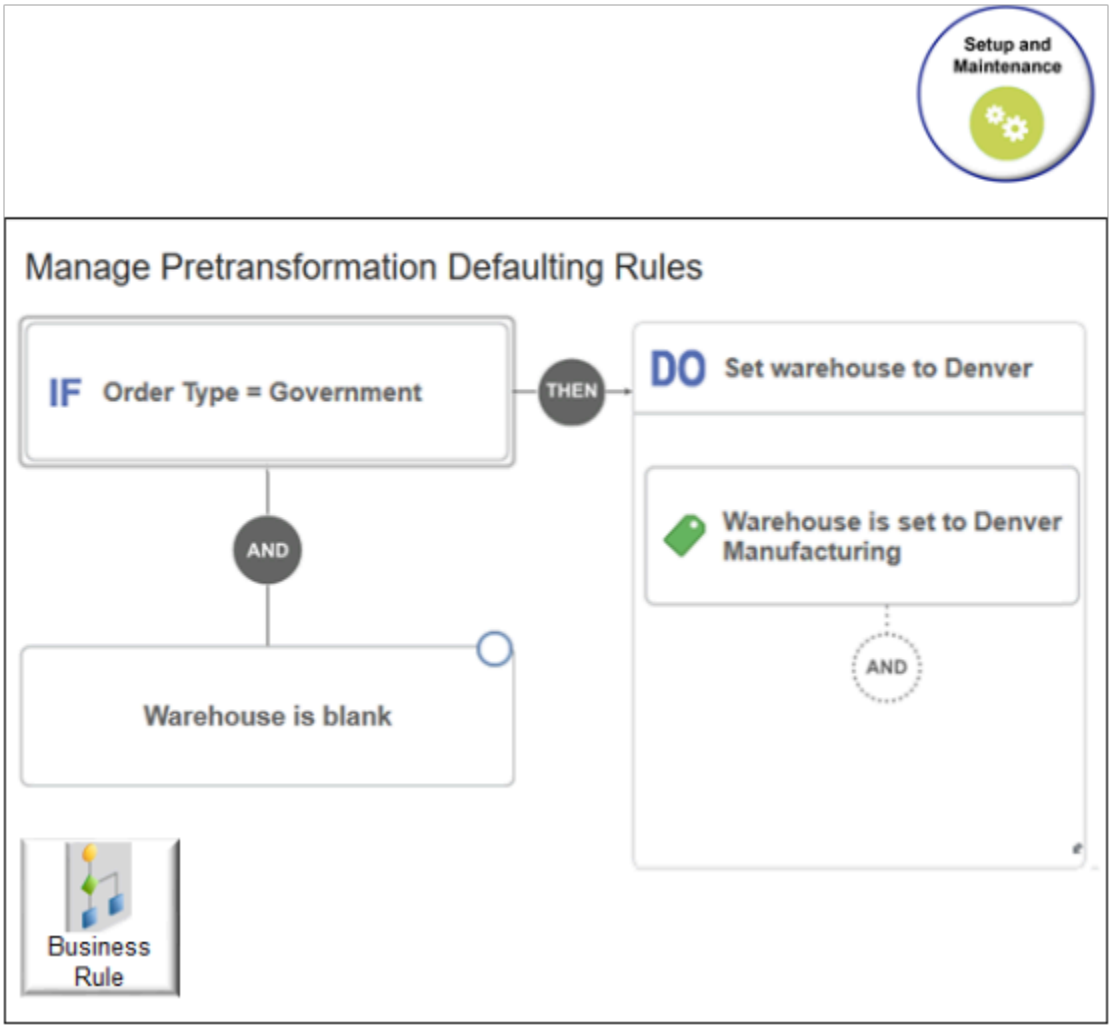
Manage Pretransformation Rules

Use Visual Information Builder to manage a pretransformation rule that sets the default value of an attribute according to the value of another attribute.

You will create a pretransformation rule.

- If order type is Government, then set warehouse to Denver Manufacturing.

For example:



Summary of the Steps

1. Create the If statement.
2. Create the Then statement.
3. Activate and publish your rule.

This topic uses example values. You might need different values, depending on your business requirements.

Create the If Statement

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Pretransformation Rules for Sales Orders

2. On the Manage Pretransformation Defaulting Rules page, click **Create New Rule**.
3. Set the values.

Attribute	Value
Name	Pretransformation Default Rule for Warehouse
Description	Use warehouse Denver Manufacturing for government sales orders.

4. Name the If statement so it describes the meaning of the condition.
    - Click **Enter Description**.
    - In the Enter Description dialog, enter `Order Type = Government`, then click **OK**.
5. Click **New Condition** (the dashed circle in the IF area).
6. In the Create Condition dialog, enter `order type`, wait a moment, then click **Order Type (Order Header)**.

   Notice that the dialog displays suggestions when you wait after you finish typing.
7. Accept the **is equal to** condition, set order type to Government, then click **OK**.

   The IF area displays the condition.
    - `Order Type is equal to Government`
8. Click **And**.
9. In the Create Condition dialog, enter `ware`, wait a moment, then click **Warehouse (Order Fulfill Line)**.
10. Change = to `Is Blank`, then click **OK**.

    The IF area displays the condition.
    - `Warehouse is blank`

      Make sure the attribute you set in a pretransformation rule is empty before you set the value. This test helps to make sure the rule doesn't evaluate or change a value that the rule or the Order Entry Specialist already set.

Create the THEN Statement

1. Click **THEN > DO**.
2. Name the DO so it describes the meaning of the action.
    - Click **Enter Description**.
    - In the Enter Description dialog, enter `Set warehouse to Denver`, then click **OK**.
3. Click **New Action** (the dashed circle in the DO area).
4. In the Create Action dialog, enter `ware`, wait a moment, then choose **Warehouse (Order Fulfill Line)**.
5. In the Create Action dialog, click **Search**.
6. In the Search dialog, enter `Denver`, then click **Search**.
7. Under Warehouse Name, click **Denver Manufacturing**, then click **OK**.
8. In the Create Action dialog, click **OK**.

   The DO area displays the condition.

- `Warehouse is set to Denver Manufacturing`

9. Click **Save and Close**.

Activate and Publish Your Rule

1. On the Manage Pretransformation Rules for Sales Orders page, notice that the Active indicator for the Pretransformation Default Rule for Warehouse rule is grey.
2. Click anywhere in the Pretransformation Default Rule for Warehouse.
3. In the Pretransformation Default Rule for Warehouse dialog, add a check mark to the **Activate Rule** option, then click **Save and Close**.

   Notice that the Active indicator for the Pretransformation Default Rule for Warehouse rule is green.
4. Click **Publish**.

   Order Management publishes each of the rules that are active on the Manage Pretransformation Defaulting Rules page.

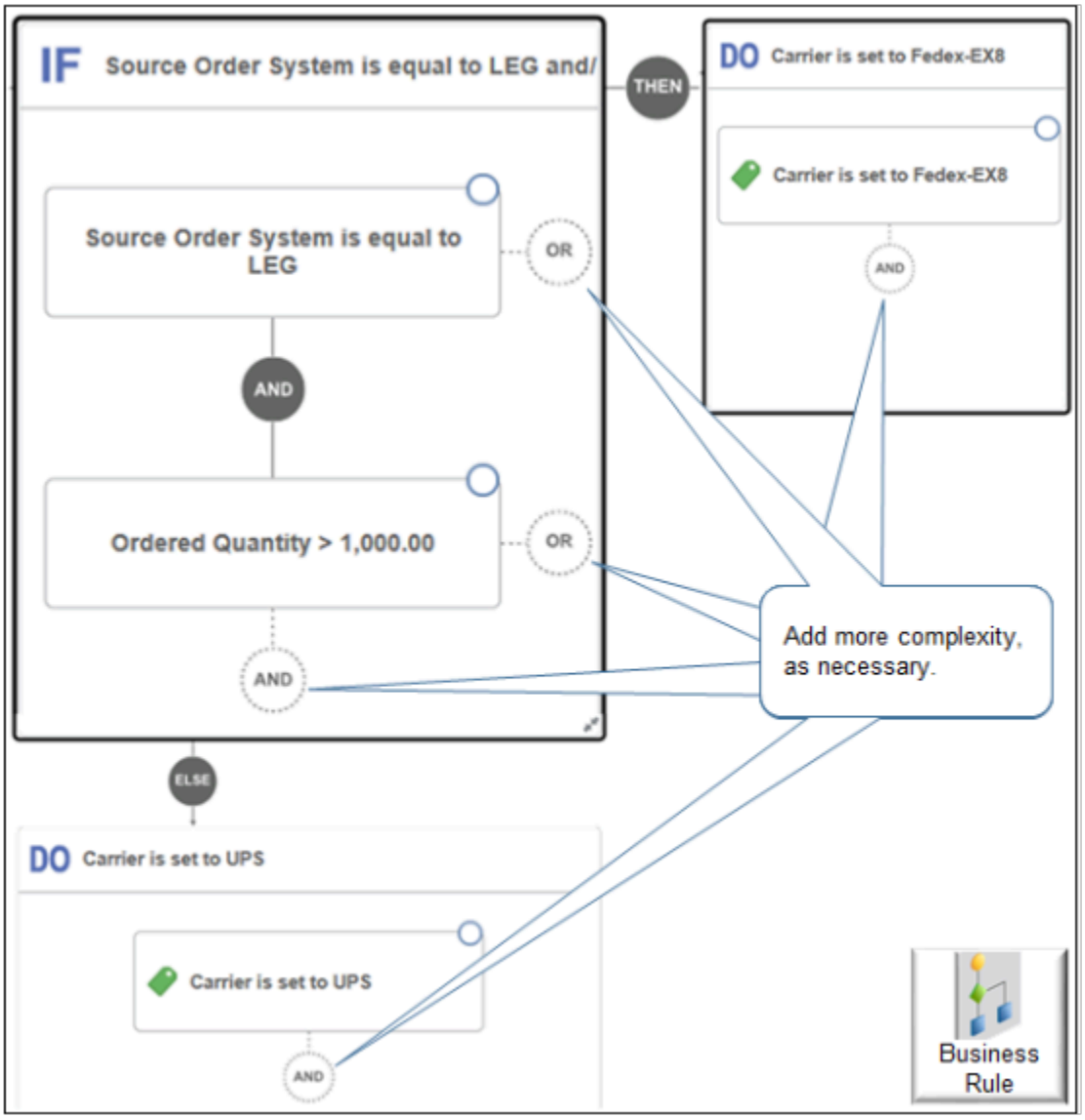
A More Complex Example

Add more complexity to your rule. For example, allow your fulfillment system to use one carrier for large source orders you create in your legacy system, and another carrier for all other sales orders.

- If the source system is a legacy system, and if the quantity is more than 1,000, and then use carrier Fedex, else use carrier UPS.

In your shipping environment, assume you typically use Fedex (Federal Express) to deliver small to medium size packages, and you use UPS (United Parcel Service) to deliver large packages.

You can add the OR, ELSE, THEN, and AND conditions to meet your business requirements.



Related Topics

- Use Visual Information Builder
- Overview of Transformation Rules
- How Order Management Transforms Source Orders Into Sales Orders
- How Order Management Transforms Order Lines Into Fulfillment Lines

8 Process Sales Orders

Control Behavior

Use Order Profiles to Control Order Management Behavior

Manage predefined profile values to control behavior in Oracle Order Management.

Most order profiles include predefined values so you don't have to set them up unless you need different values to meet your requirements.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Profiles
2. On the Manage Order Profiles page, in the Profile Option area, click **Search**.
3. In the search results, in the Profile Options list, click the **profile** that you must edit.
4. In the Profile Values list, add or delete values, as necessary.

Profile Option	Description
Aggregate According to Number of Order Lines That Changed	This option aggregates order lines according to the number of order lines that changed. The default value is Yes. For details, see the Aggregate Fulfillment Lines subtopic in For details, see the Aggregate Fulfillment Lines subtopic in *Set Up Features, Manage Change, and More for Drop Ship*. Set it to No to aggregate lines according to time or to the number of lines in a routing rule. For details, see *Aggregate Requests That Order Management Sends to Your Fulfillment System*. This option only applies on order lines in a drop shipment. It applies to changes that happen on the source order.
Aggregator Hold Timeout Period in Minutes	The aggregator applies a hold on the purchase order in Procurement to prevent other processes from updating the order while aggregating. Specify the number of minutes for the hold operation to wait before timing out while aggregating order lines. • The default value is 120 minutes. • The minimum value is 30 minutes. The hold operation will wait for 30 minutes even if you set the value to less than 30. For example, if you set the value to 15, then the hold operation will wait for 30 minutes. Use the aggregator profiles together. If you don't enable Aggregate According to Number of Order Lines That Changed, then Order Management ignores Aggregator Hold Timeout Period in Minutes.
Currency Conversion Type	Specify the value to use when converting a currency in the Order Management work area. This value is a conversion type.

Profile Option	Description
Display Currency	Specify the currency that Order Management will display in the Order Management work area, by default.
Fulfillment Task Retries	See the section below in this topic.
Hours to Wait Before Allowing Date Changes on Fulfillment Lines	See the section below in this topic.
Map Sales Credit from Parent to Child	Map the sales credit on the parent configured item to the sales credit on child configure options that don't already have a credit. Consider an example. • Assume you have a configured item named AS54888 Desktop Computer, and its the parent. • The keyboard, monitor, and mouse are AS54888's children. • The keyboard and the monitor don't already have a credit, but the mouse does already have a credit of 20. If you enable this option, and if you set the sales credit to 10 for the AS54888 on the sales order, then Order Management will set the sales credit for the keyboard and monitor to 10, and the credit for the mouse will remain at 20.
Number of Orchestration Processes to Start Concurrently for Large Sales Orders	Specify the number of orchestration process instances to start concurrently for your sales order. You can use this profile to improve performance for your large sales orders. Assume you use the predefined ShipOrderGenericProcess orchestration process, your sales order has 5,000 lines, and you set this profile to: • 100. Order Management will create 50 separate batches for the sales order, and each batch will process 100 lines. Order Management will process each batch sequentially, so it will run no more than 100 instances of ShipOrderGenericProcess concurrently. • 1. You will have 5, 000 instances of ShipOrderGenericProcess all running at the same time, and this will degrade performance. The default value is 100. We recommend that you start with the default value, modify it and monitor performance in a test environment until you achieve the desired performance, then set the value in your production environment.
Percent of Order Lines in Error in Each Hour	Specify the percent of lines that are in error that Order Management should attempt to fix in each hour. See the section later in this topic.
Populate Split Lines with Values from Original Line	If you enable this option, and if the user splits a line in the Split Fulfillment Line dialog, then Order Management will populate values on the split line with values from original line. Note • Order Management will populate values for the Warehouse, Supplier, Supplier Site, Shipping Method, and Demand Class attributes. • If the value of any of these attributes is empty on the original line, then it will be empty on the split line too. • Your users can remove the populated value and select some other value, or leave the attribute empty.

Profile Option	Description
	• This profile option only affects lines that your user manually splits in the Split Fulfillment Line dialog. It doesn't affect splits that Order Management automatically does through a scheduling task or a fulfillment task.
Required Overview Status Filter	Specify the default customer to use when filtering the summary of status data on the Overview page of the Order Management work area. It allows your users to view summary data for only one customer at a time. It removes the All option. Order Management provides no value, by default. To improve performance, you can enter a customer identification number.
Respond Immediately on Start of Submission Request	Specify when the Sales Order for Order Hub REST API sends a response to each request that it receives. • **Enabled**. Send a response when starting the order submit. • **Not Enabled**. Send a response after finishing the order submit. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.
Retain Sales Order Number During Import	For details, see *Keep Your Order Number When You Import*.
Send Fulfillment Details for Drop Shipments to Accounts Receivable	This option comes predefined with the Site level disabled. If you enable it, then Order Management will send fulfillment details for drop shipments to Accounts Receivable. It will send the waybill number, bill of lading, and so on.
Skip Availability When Searching for Item	Improve performance when the user searches for an item on the catalog line. If you set this option to Yes, then Order Management doesn't send a request to Global Order Promising to determine whether the item is available. If you don't use Global Order Promising, then set this option to Yes.
Skip PricingTotals and Pricing Validation	Improve performance. Don't validate the price and don't calculate the total price of the sales order when the user clicks Add on the catalog line. Order Management will calculate the sales order total only when the user saves the order as a draft or clicks Submit.
Total Number of Order Lines in Each Hour	Specify the average number of order lines that you typically process in one hour.
Truncate Attribute Values for Accounts Receivable	See the section below in this topic.
User Request Waiting Period in Seconds	Specify the number of seconds to wait after an action finishes. This time allows each asynchronous web service to finish before displaying a confirmation message or a warning message in the Order Management work area. The default value is 5.
Use Global Order Promising to Recalculate Dates in Order Management	See the section below in this topic.

Fulfillment Task Retries

Set the number of times that Order Management should rerun a fulfillment task before it sends a reply to your fulfillment system when lines are locked or there's a problem with a wait task.

You must enter a value in the Profile Value attribute. Use this format:

task_type #times_to_retry,task_type#times_to_retry,task_type#times_to_retry,task_type#times_to_retry

where

- *task_type* identifies the type of fulfillment task.
- *times_to_retry* specifies the number of times to retry the task.
- You can specify task types in any sequence. For example, you can specify All first, and then Shipment, Procurement, and finally Generic.

Specify the Task Type

Use these values for *task_type*.

Value	Description
Shipment	Retry all fulfillment tasks that involve the Shipment task type.
DOO_Procurement	Retry all fulfillment tasks that involve the Procurement task type.
DOO_$GENERIC$	This is a profile value. It will retry all the fulfillment tasks that you create. For details about fulfillment tasks that you create, see *Create Your Own Task Type*.
DOO_ALL	This is a profile value. It will retry all predefined fulfillment tasks and all custom tasks that you create.

Note the format requirements. You must:

- Include the DOO_ prefix for GENERIC and ALL.
- Use all upper case letters for DOO_GENERIC and DOO_ALL.
- Enclose DOO_GENERIC and DOO_ALL with dollar signs ($ $). For example, DOO_ALL and DOO_$GENERIC$. The dollar signs are wildcards.
- Use the comma to separate each value.

Don't include the space character anywhere in your code.

Specify Predefined Task Types

You must match the name of each predefined task type exactly, including capitalization. Some predefined task types include the DOO_ prefix, but others don't. Here's how you can determine the format that you need:

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Task Types
2. On the Manage Task Types page, notice the value in the Task Type column. You must use this value. For example:
    - Schedule doesn't have a DOO prefix, so don't include that prefix with the Schedule task type.
    - DOO_Procurement has a DOO prefix, so you must include that prefix with the DOO_Procurement task type.

Specify the Times to Retry

Use these guidelines for *times_to_retry*.

- Specify a minimum value of 0 and a maximum value of 9.
- If you specify a value that's greater than 9, then Order Management will still use 9. For example, set the retry to 10, and Order Management will attempt to rerun the fulfillment task 9 times before it replies with an error.
- If Order Management can't process the fulfillment task after the number of times that you specify, then it will send an error to your fulfillment system.
- The value that you specify depends on your server's performance. If you find that specifying a higher value degrades performance, then use a lower value.

Example

Consider an example.

```
Shipment#5,DOO_Procurement#4,DOO_$GENERIC$#7,DOO_$ALL$#3
```

In this example, Order Management will retry:

- Shipment tasks 5 times
- Procurement tasks 4 times
- Generic tasks 7 times
- All tasks except Shipment, Procurement, and Generic tasks 3 times

Consider Priority

DOO_ALL has the lowest priority. If you provide a value for Shipment, DOO_Procurement, or DOO_GENERIC, and you also provide a value for DOO_ALL, then Order Management uses the value that you provide in Shipment, DOO_Procurement, or DOO_GENERIC, and then DOO_ALL.

Assume you provide:

```
DOO_$ALL$#4,DOO_$GENERIC$#6,Shipment#5
```

Order Management will retry all the fulfillment tasks that you create 6 times, all the shipment tasks 5 times, and all other tasks 4 times.

Notes

- The profile option code is ORA_DOO_FULFILLMENT_TASK_RETRIES.
- You might not receive a reply for some fulfillment tasks. For example, Order Management will attempt to retry a fulfillment task in a drop ship flow but it won't send a reply.
- A response can be immediate or delayed. Having a delayed response doesn't necessarily mean there's a problem. For example, Order Management might be waiting for a shipping task to finish, so it can't send a response until that task is done. Or it might be processing a number of other lines and can't process the fulfillment task until its done with those other lines. For details about the delayed response, see *Overview of Connecting Order Management to Your Fulfillment System*and *Connect Order Management to Your Fulfillment System*.

Hours to Wait Before Allowing Date Changes on Fulfillment Lines

Specify the number of hours to wait before allowing the Order Entry Specialist to update attributes on the fulfillment line for a drop shipment.

Order Management allows the user to modify attributes when one of these conditions is true.

- Oracle Procurement hasn't created the purchase requisition.
- The time that you specify in this profile has elapsed.
- The Scheduled Ship Date or the Scheduled Arrival Date has occurred.

The default value is 4 hours.

If you add an order line that includes a drop shipment and submit the sales order, then Order Management sends the fulfillment line to Oracle Procurement, and Procurement starts the flow that creates a purchase requisition for the item but it doesn't send updates to Order Management until it finishes the flow. To keep the fulfillment line consistent with the purchase requisition, Order Management doesn't allow you to modify the fulfillment line until it receives the requisition from Procurement. Procurement usually creates the requisition in a few minutes or less, but in some situations the response might be delayed or Order Management might not receive any response.

At some point, you might need to edit attribute values on the fulfillment line to reprocess the requisition. Use this profile option to specify how long to wait. For example, if you learn through experience that some responses from Procurement take as long as 1 hour to finish, but never more than 2 hours, then set this profile to 3. That way, the Order Entry Specialist can't modify attributes while the response is delayed.

Percent of Order Lines in Error in Each Hour

Use the Total Number of Order Lines in Each Hour profile and the Percent of Error Lines to Process Each Hour profile together to help avoid performance problems that might happen when you run the *Recover Errors* scheduled process.

Assume you set these values.

Profile Option	Value
Total Number of Order Lines in Each Hour	25000
Percent of Order Lines in Error in Each Hour	10

2,500 is 10% of 25,000, so the Recover Errors scheduled process will attempt to recover about 2,500 order lines that are in error each hour. If the actual number of lines that are in error is significantly more than 2,500, say about 5,000, then the Recover Errors scheduled process will attempt to recover the lines in about 2 hours.

Set Percent of Order Lines in Error in Each Hour conservatively, such as 10%. Set it to a value that works for a typical business day where you don’t expect a lot of lines will go into error.

Increase Percent of Order Lines in Error in Each Hour only when you must recover lines that are in error at a much faster rate than usual. For example, something happens that causes errors to accumulate, such as your server goes down. If you increase it:

- Increase it gradually, observe performance, then use a value that meets your business requirements.
  Consider the time of day when you run the Recover Errors process. Try to run it when you don't have other activities running that consume a lot of resources on your server.
- Monitor performance. Adjust the values of these profiles as necessary according to the performance that you observe.

The values in this topic are example values. They aren't recommendations. You must use your own values to meet your business requirements.

For details, see *Fix Errors in All Sales Orders*.

Truncate Attribute Values for Accounts Receivable

Use the Truncate Attribute Values for Accounts Receivable profile to truncate some of the attribute values that Order Management sends to Accounts Receivable. You can set the profile to:

- **Yes**. Truncate the attribute values to the recommended length.
- **No**. Don't truncate any values.

Order Management sends these order line attributes to Accounts Receivable for billing.

Attribute	Maximum Recommended Length
Bill Of Lading Delivery Name	The value for each attribute must not exceed 30 characters.
Charge Component Explanation	The attribute's value must not exceed 240 characters.

If the value that Order Management sends for any of these attributes doesn't match the length that Accounts Receivable uses, then billing fails in Accounts Receivable. To avoid this problem, you must truncate the length so it doesn't exceed the recommended length.

- If you set the PreCreditCheckedFlag attribute to Y when you import or revise a sales order, and if the value in the Credit Check Authorization Number attribute in your import or revision exceeds 30 characters, then Order Management doesn't truncate the value in Credit Check Authorization Number but instead will reject the sales order.
- Order Management applies this behavior only when you integrate with Oracle Accounts Receivable. It doesn't apply when you integrate with some other accounts receivable application.
- Order Management validates the Credit Check Authorization Number attribute when you import the sales order regardless of how you set the profile.

The profile code is FOM_TRUNCATE_AR_ATTRIBUTE_VALUES.

Example

Consider an example where, if you enable the profile, then Order Management truncates the value so it doesn't exceed 30 characters.

Attribute	Profile Not Enabled	Profile Enabled

Bill Of Lading	This value has 57 characters:
	
	`Split300100580225779BOL:VisionManufa`
	`San Francisco`
		This value has 30 characters:
		
		`Split300100580225779BOL:Vision`

Delivery Name	This value has 57 characters:	This value has 30 characters:

Attribute	Profile Not Enabled	Profile Enabled
	`Split300100580225779DEL:VisionManufa San Francisco`	`Split300100580225779DEL:Vision`

If you enable the profile, then Order Management truncates the value so it doesn't exceed 240.

Attribute	Profile Not Enabled	Profile Enabled
Charge Component Explanation	This value has 262 characters: `TESTING DEPARTMENT COMPUTER SERVICE AND RENTALS 10-30-2023 AS54888 DESKTOP COMPUTER - Discount Rule COMPUTER SERVICE AND RENTALS DISCOUNT RULE defined for the item AS54888 DESKTOP COMPUTER applied from the discount list COMPUTER SERVICE AND RENTALS DISCOUNT LIST`	This value has 240 characters: `TESTING DEPARTMENT COMPUTER SERVICE AND RENTALS 10-30-2023 AS54888 DESKTOP COMPUTER - Discount Rule COMPUTER SERVICE AND RENTALS DISCOUNT RULE defined for the item AS54888 DESKTOP COMPUTER applied from the discount list COMPUTER SERVICE AND`

Use Global Order Promising to Recalculate Dates in Order Management

If you enable this option:

- And if you change the promised ship date or promised arrival date in Oracle Procurement
- Or if you change the work order date in Oracle Manufacturing
- Or if you change the requested date in Oracle Inventory

Then Order Management will use Global Order Promising to calculate the scheduled ship date or scheduled arrival date for sales orders in Order Management.

Specify the profile value to determine when to apply this behavior.

Value	Description
Back-to-Back	Apply this behavior only in back-to-back flows.
Drop Shipment	Apply this behavior only with sales orders that include a drop shipment.
Back-to-Back and Drop Shipment	Apply this behavior in back-to-back flows and with sales orders that include a drop shipment.
Neither Back-to-Back nor Drop Shipment	Don't apply this behavior on any sales orders. This is the default value.

Note

- Order Management doesn't send the revised scheduled ship date or scheduled arrival date to Oracle Procurement, so the dates on the sales order in Order Management won't match the dates on the purchase order in Procurement.
- You can use this feature only with orchestration processes that use Global Order Promising.
- You can use this option only for a single site.

- If the buyer manages transportation, then Order Management maps the Requested Delivery Date on the purchase order to the Scheduled Arrival Date on the sales order. Order Management and Global Order Promising ignore the value in Requested Delivery Date when recalculating the Scheduled Arrival Date.

Example When Buyer Manages Transportation

Buyer manages transportation

	Order Management				Procurement			
	Requested Ship Date	Requested Arrival Date	Scheduled Ship Date	Scheduled Arrival Date	Requested Ship Date	Promised Ship Date	Requested Delivery Date	Promised Delivery Date
Customer places order	3-25-21	3-30-21	-	3-30-21	3-25-21	3-25-21	3-30-21	-
Buyer changes Promised Ship Date to 3-30-2021	-	-	3-30-21	-	-	3-30-21	-	-
Update from Global Order Promising	-	-	3-30-21	4-6-21	-	3-30-21	-	-

Legend
Buyer Updates the Date
Global Order Promising Updates the Date

Buyer doesn't manage transportation

	Order Management				Procurement			
	Requested Ship Date	Requested Arrival Date	Scheduled Ship Date	Scheduled Arrival Date	Requested Ship Date	Promised Ship Date	Requested Delivery Date	Promised Delivery Date
Customer places order	2-25-21	-	2-25-21	2-28-21	-	-	2-28-21	2-28-21
Buyer changes Promised Delivery Date to 3-10-21	-	-	-	3-10-21	-	-	-	3-10-21
Update from Global Order Promising	-	-	3-7-21	3-10-21	-	-	-	3-10-21

Note

Buyer Manages Transportation	Buyer Doesn't Manage Transportation
Assume you set the transit lead time to 5 days in Global Order Promising.	Assume you set the transit lead time to 3 days in Global Order Promising.

Buyer Manages Transportation	Buyer Doesn't Manage Transportation
The buyer can revise the Promised Ship Date in Procurement.	The buyer can revise the Promised Delivery Date, but can't revise the Promised Ship Date in Procurement.
Global Order Promising updates the Scheduled Arrival Date.	Global Order Promising updates the Scheduled Ship Date.

Hierarchy of the Profile Options

You can set a profile option at different levels to specify a hierarchy.

We recommend that you set site values for your profile option before you set the values for the product or user.

Level	Priority	Example
Site	Lowest	Affects all of Order Management. For example, set the currency for all of Order Management to Euro.
Product	Higher than Site	Affects only the item. For example, set the currency for the AS54888 item to Renminbi.
User	Higher than Product	Affects only the current user. For example, set the currency for the user to US Dollar.

You can use each profile only for a single site, except you can use Currency Conversion Type and Retain Sales Order Number During Import for a single site and for each user.

Related Topics

- Integrate Order Management with Source Systems
- Roadmap for Setting Up Order-to-Cash
- Keep Your Order Number When You Import

Use Groups to Manage and Control Sales Orders

Set up the Order Type attribute on the order header to arrange sales orders into groups that you specify, then use these groups to help manage sales orders and control order processing.

- The Order Entry Specialist can search for sales orders on the Manage Orders page according to order type. This capability focuses the search and helps improve search productivity.

- The Manage Orders page doesn't come predefined to display the Order Type attribute in the search area. The Order Entry Specialist must click Add Fields, then add it to the search area.
- If you import a source order from a source system, then Order Management will examine the value of Order Type in the source order to make sure it matches one of the values that you set up for the lookup. If it doesn't match, then Order Management rejects the import and logs an error.

Use Order Type to Control Processing

Use Order Type in Business Rules

Use the order type in a business rule.

Type of Rule	Description
Product transformation	Use the order type as part of a pretransformation, transformation, or posttransformation rule. For example: • Write a rule that sets the default value for the order type according to the value of some other attribute on the order header. For example, assume your company writes a contract to fulfill sales orders from Customer y over all other customers. You can write a rule `if Customer is Y, then set Order Type to Top Priority`, then use a routing rule that routes the sales order to a fulfillment system that prioritizes shipment speed over other factors, such as cost. • Write a rule that uses the value of the order type to determine the default value to display for an attribute on the order header or order line. Order Management runs each pretransformation rule when the user creates or revises a sales order in the Order Management work area. For example, when the user sets or updates the Business Unit, Customer, or Order Type, or adds an order line. To avoid overwriting these selections, you can write a pretransformation rule so it sets a value only if the attribute is empty.
Process assignment	Write a rule that uses the order type to assign an orchestration process. For example, if Order Type is Government Department of the Interior, then set up an orchestration process that fulfills order lines according to Department of the Interior procurement requirements.
Routing	Write a rule that uses the order type to determine which fulfillment system will fulfill the sales order. For example, if Order Type is Emergency, then route the sales order to a fulfillment system that prioritizes shipment speed over other factors.

Use Order Types in Processing Constraints

Use the order type in the condition or result of a processing constraint. For example, prevent Order Management from updating an attribute according to the value of some other attribute. You can also use the order type to prevent Order Management from submitting a sales order. For example:

- If Order Type is x, then don't allow change to attribute y after submit.
- If Business Unit is x, then don't allow change to Order Type after submit.
- If a submit validation happens, and if Order Type is empty, then display an error.
- If a submit validation happens, and if Order Type is x, and if Purchase Order is empty, then display an error.

Set Up Order Types

In this example, you set up four order types so you can differentiate between sales orders in a deployment that uses drop ship.

This topic uses example values. You might need different values, depending on your business requirements.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups
2. On the Manage Order Lookups page, in the Search area, enter the value, then click **Search**.

Attribute	Value
Lookup Code	ORA_DOO_ORDER_TYPES

   Order Management uses the ORA_DOO_ORDER_TYPES lookup for attribute Order Type, but doesn't come predefined with values for this lookup. If you don't add any values, then Order Management still displays the Order Type attribute but it won't contain any values.
3. In the Lookup Codes area, click **Actions > New** to add each lookup code.

Lookup Code	Meaning
STD	Standard Sales Order
STD_DS	Drop-Ship Sales Order
MIX	Standard and Drop Ship Order Lines
RETN	Return Sales Order

   You can modify other attributes, as necessary.
4. Click **Save and Close**.

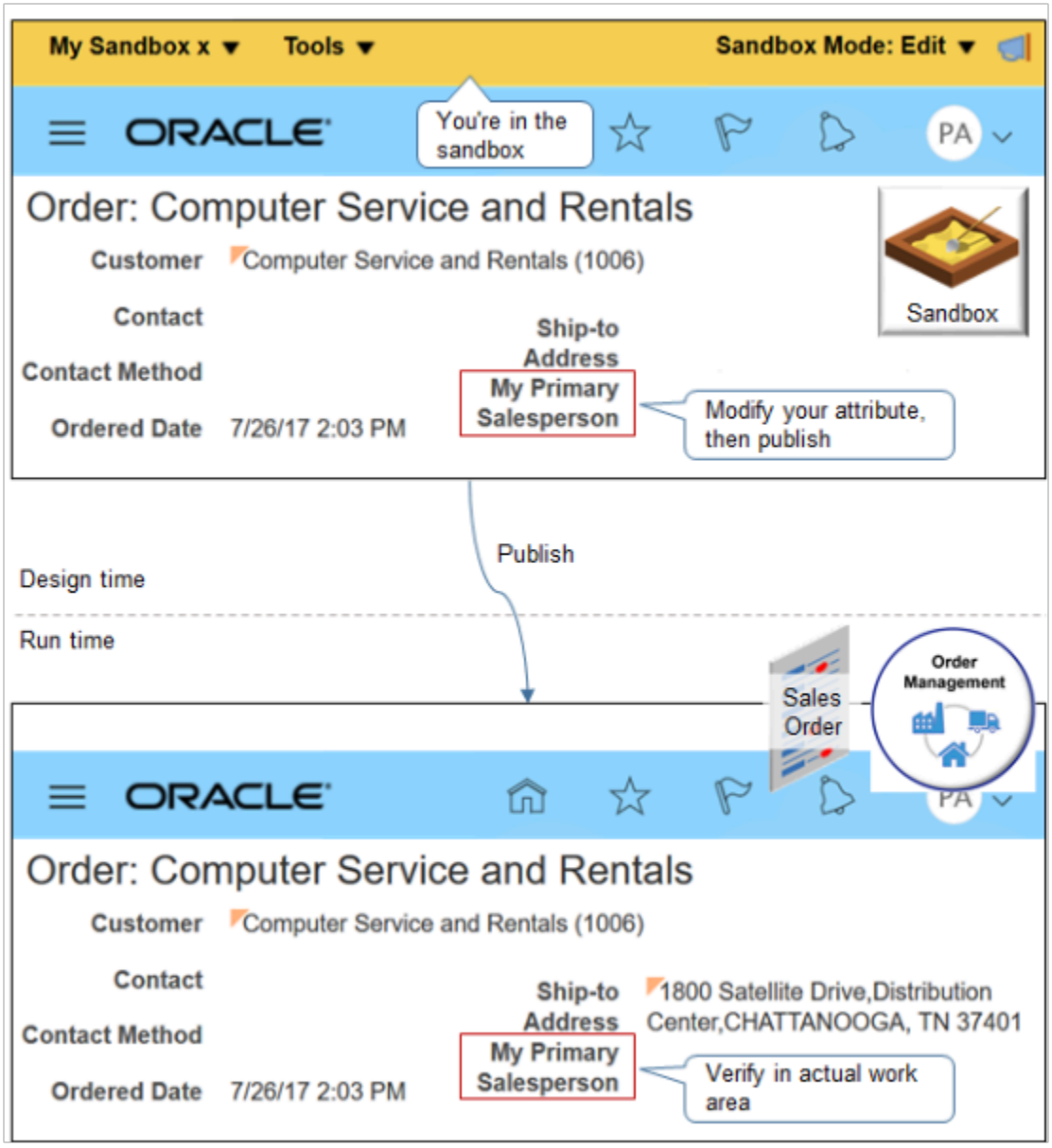
Related Topics

- Processing Constraints
- Use Visual Information Builder
- Guidelines for Assigning Orchestration Processes
- Route Requests from Order Management to Fulfillment Systems
- Manage Lookups in Order Management

Modify How Order Management Displays Attributes

Order Management comes predefined with hidden attributes that you can expose so the Order Entry Specialist can display them in the Order Management work area. You can also modify the display name of attributes to meet your needs.

You can expose and modify attributes in various locations, such as in the sales order header or on the order line. The Order Entry Specialist then uses the Order Management work area to display them. You use a sandbox to do the change, test it, publish it, then use the runtime version of the work area to verify.



In this example, you create a sandbox named My Sandbox, expose the Primary Salesperson attribute, then modify the display name so it says My Primary Salesperson. For more details, including screen prints, see *How To Display Primary Sales Person LOV on Order Entry Form Via Sandbox Page Customization. (Doc ID 2394038.1)*.

1. Create your sandbox.
    - On your home page, in the upper-right corner, click your **login name**.
    - In the Settings and Actions menu, click **Edit Pages**.
    - In the dialog that displays, click **Activate Sandbox**.
    - On the Sandbox page, click **Create Sandbox**. Create a new sandbox, activate it, then open it.
    - On the Create Sandbox page, on the Page Composer row.
        - In the Active column, add a check mark.
        - In the Support Context column, click the pencil, set the Category to Default, then click OK.
    - Enter a name, then click **Create**.

Attribute	Value
Name	My Sandbox

    - On the Sandbox page, in the My Sandbox row, click **Enter Sandbox**.
    - Make sure you're in the sandbox. Verify that a banner displays along the top of the page. The banner includes the name of your sandbox.
2. Open Page Composer.
    - Go to the Order Management work area, then click **Tasks > Manage Orders**.
    - On the Manage Orders page, search for, then open any sales order.
    - On the Order page, in the upper-right corner, click your **login name**.
    - In the Settings and Actions menu, click **Edit Pages**.

      The Order Management work area displays the Oracle Page Composer, which is a page editor that you can use to modify the visual layout that the work area uses. You can tell you're in Page Composer because it displays a section across the top of the page. The section includes tabs, such as Add Content, Select, and Structure.

      To modify an attribute in another work area, go to that work area, then go to the page in the work area that displays the attribute you must modify.

      Get details about how to use Page Composer. Go to *Configuring and Extending Applications*, then search for Guidelines for Using Page Composer.

3. Expose the attribute.
    - Click **Select**.
    - Click in the lower portion of the order header, for example, below the Customer attribute, then click **Edit Parent Component**.
    - In the Component Properties dialog, click **Children**, then notice the list of attributes that display.

      Each attribute that doesn't contain a check mark is hidden. You can add a check mark to unhide the attribute.
    - Add a check mark next to Primary Salesperson, then click **Apply > OK**.
4. Verify values in the sandbox.
    - Verify that the order header displays the Primary Salesperson attribute.
    - In the Order Lines area, click **View > Columns**, and notice you can display the attribute.
    - In the upper-right corner, click **Close**.
    - On the Order page, click **Done**.
    - Notice that you can now display the Primary Salesperson attribute on the Manage Orders page.
        - In the Search area, click **Add Fields**.
        - In the Search Results, click **View**, then click **Columns**, and notice you can display the Primary Salesperson attribute.
5. Modify the display name.
    - Navigate back to the order page.
    - In the order header, click **Primary Salesperson**, then click **Edit Component**.
    - In the Component Properties dialog, next to Label, click **Edit > Override**, then click OK.
    - In the order header, click **Primary Salesperson**, then click **Edit Component**.
    - In the Component Properties dialog, in the text window next to Label, enter `My` before the text Primary Salesperson, then click **OK**.
    - Verify that the order header now displays My Primary Salesperson.
6. Publish.
    - At the top-right corner, click **My Sandbox > Publish**.
    - On the Sandbox Detail page, click **Publish**.
    - In the dialog, click **Continue to Publish**.

      You're longer in the sandbox. You're now back in Order Management.
    - Repeat the verification you did earlier in the sandbox, but this time use the Order Management work area outside of the sand box.

Use Line Numbers to Filter Large Sales Orders

You can use query by example to filter a large sales order according to a range of attributes, but you can't use it to filter according to the order line. Instead, you can use Page Composer to add the Order Line attribute, and then filter by line.

You can add the Order Line attribute to any of these tabs:

- Order Lines

- Shipment Details
- Payment and Billing Details
- Sales Credits

Assume you want to add the Order Line attribute on the Order Lines tab. Do the procedure described above in this topic, but with these differences:

- When you use the Component Properties dialog, add a check mark to the Order Line attribute.
- Skip the step where you modify the display name.

Modify an Attribute's Width

Assume your items have a really long number. In fact they're so long that you can't see the entire number when you enter it in the Select Item field on the Create Order page in the Order Management work area. Here's your fix:

1. Open Page Composer, then open the Properties dialog of the Item Number field.
2. Use the Style tab to increase the width of the item to a higher value, such as 250px.

For details, see *How to Customize a Field in Oracle Applications Using Page Composer (Doc ID 2115127.1)*.

Related Topics

- Create and Activate Sandboxes
- Guidelines for Using Page Composer

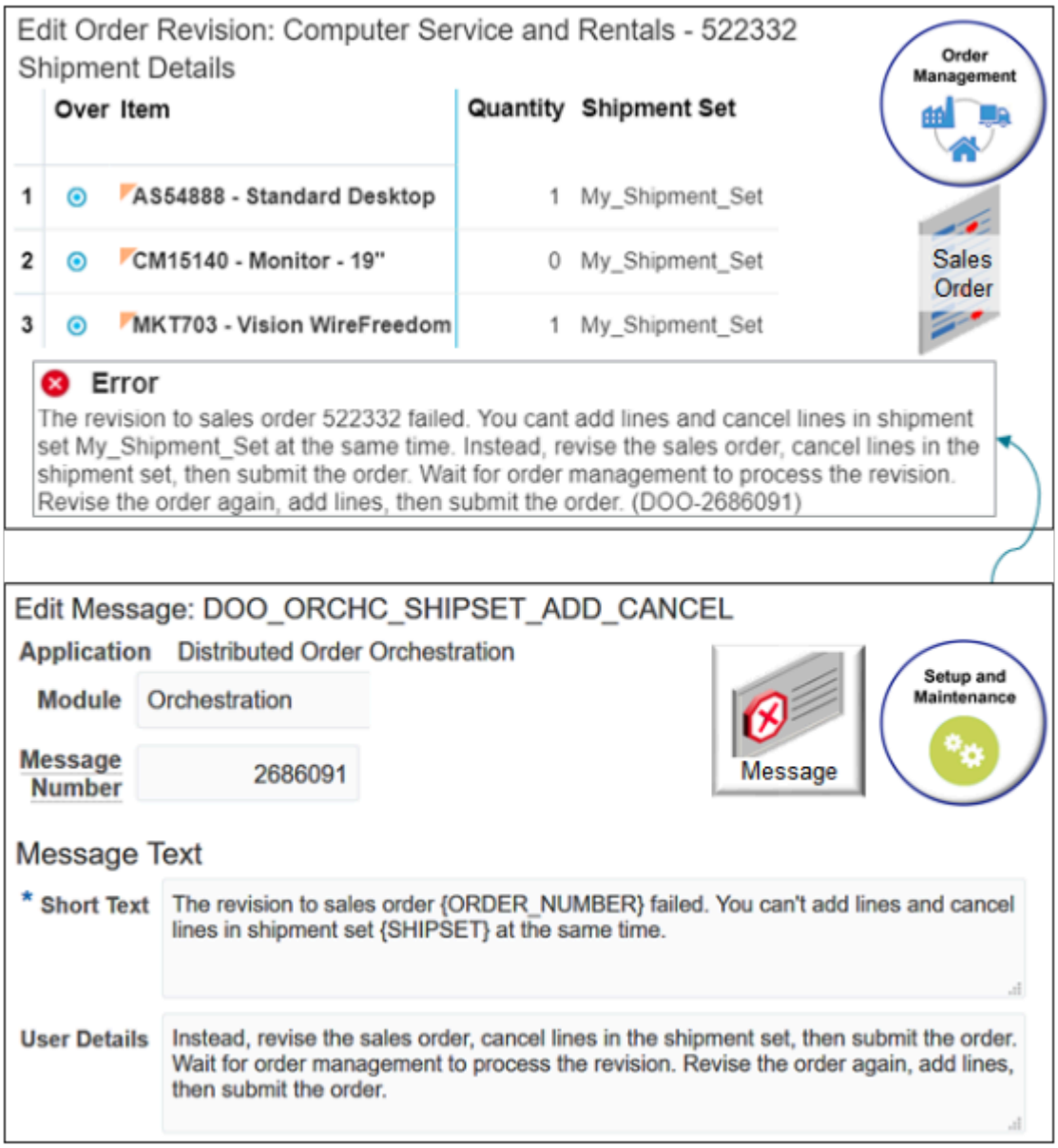
Set Up Messages in Order Management

Modify a message that displays in the Order Management work area, or create your own message.

Examine a Predefined Message

1. Go to the Order Management work area and create a sales order.
2. Add two order lines, assign each line to a shipment set named My_Shipment_Set, then click **Submit**.
3. Click **Actions > Create Revision**.
4. Add a new order line and assign it to My_Shipment_Set.
5. To cancel line 2, set the quantity on line 2 to zero.

6. Click **Submit**, then examine the message that displays.

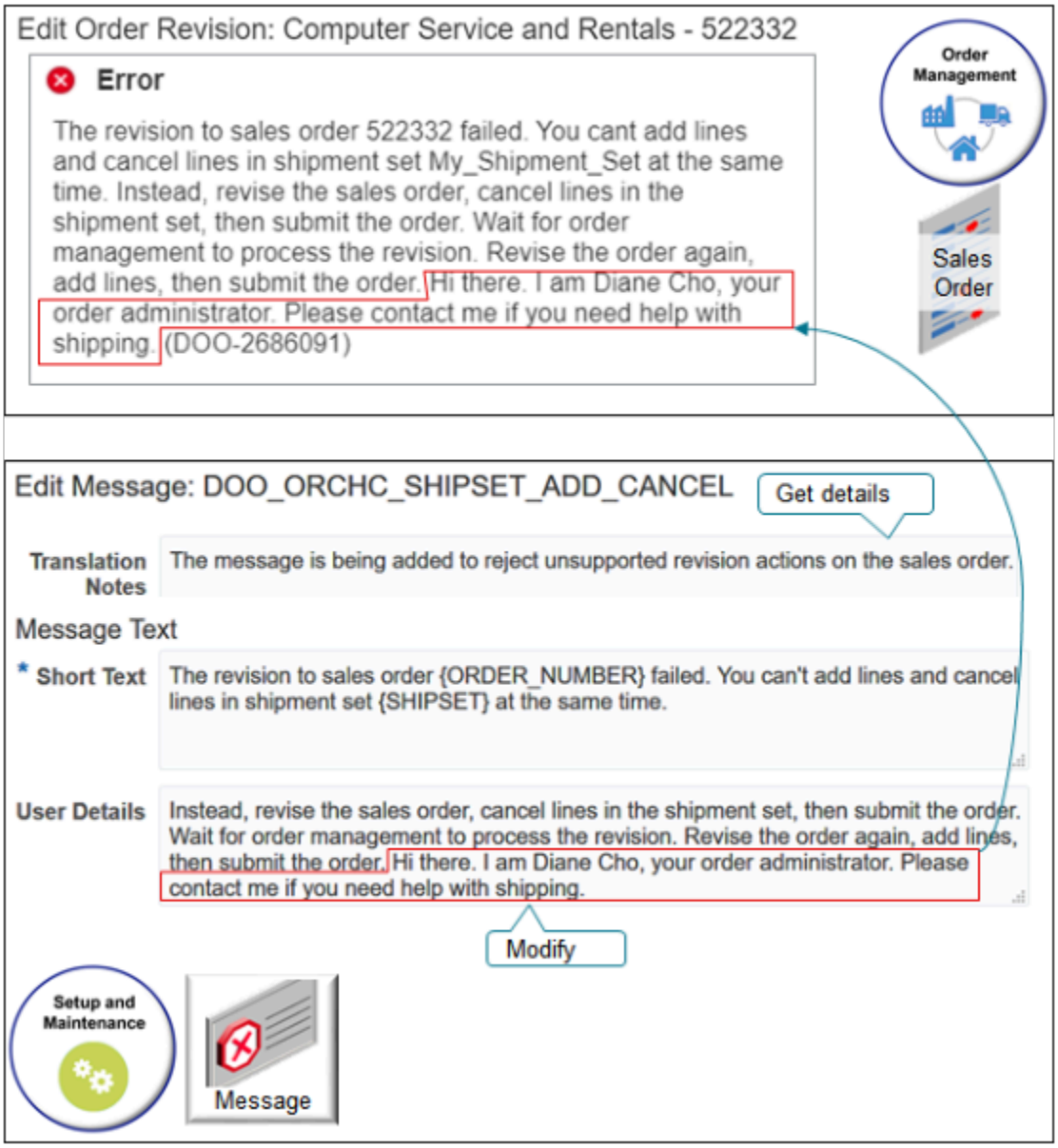


Note

- Order Management doesn't allow you to add a new line and cancel a line in the same shipment set during the same revision, so it displays the error message.
- The predefined DOO_ORCHC_SHIPSET_ADD_CANCEL message defines the message text.

Modify a Predefined Message

Assume you want to add some text to DOO_ORCHC_SHIPSET_ADD_CANCEL.


Edit Order Revision: Computer Service and Rentals - 522332
Error
The revision to sales order 522332 failed. You cant add lines and cancel lines in shipment set My_Shipment_Set at the same time. Instead, revise the sales order, cancel lines in the shipment set, then submit the order. Wait for order management to process the revision. Revise the order again, add lines, then submit the order. Hi there. I am Diane Cho, your order administrator. Please contact me if you need help with shipping. (DOO-2686091)
Order Management
Sales Order
Edit Message: DOO_ORCHC_SHIPSET_ADD_CANCEL
Get details
Translation Notes
The message is being added to reject unsupported revision actions on the sales order.
Message Text
* Short Text
The revision to sales order {ORDER_NUMBER} failed. You can't add lines and cancel lines in shipment set {SHIPSET} at the same time.
User Details
Instead, revise the sales order, cancel lines in the shipment set, then submit the order. Wait for order management to process the revision. Revise the order again, add lines, then submit the order. Hi there. I am Diane Cho, your order administrator. Please contact me if you need help with shipping.
Modify
Setup and Maintenance
Message

Try it.

1. Examine the predefined message.
    - Open another browser window.

      You open a second browser window so you can toggle between your design time set up in the Setup and Maintenance work area and run time rendering in the Order Management work area.
    - In the Setup and Maintenance work area, click **Tasks > Search**.
    - On the Search page, search for, then open Manage Messages.
    - On the Manage Messages page, set the value, then click Search.

Attribute	Value
Application	Order Management You can also search for Distributed Order Orchestration to examine messages that Order Management displays primarily during order fulfillment or during interactions with a web service.

Refine your search. Remove the value from the Application attribute and enter a value in Message Name, such as DOO_ORCHC_SHIPSET_ADD_CANCEL.

If you don't know the name, then remove the value from the Application attribute and enter a value in the Module attribute.

Module	Message Includes Errors That Happen When
Import Order	You import a sales order.
Decomposition	Preparing a source order for Order Management during order import.
Orchestration	An orchestration process runs.
Fulfillment Task	Processing a request through a fulfillment task.
Process Order	Order Management starts to process fulfillment. There's usually missing data, attribute values aren't correct, there's a problem in communicating with other applications during fulfillment, and so on.
Manage Orders	You are creating or revising a sales order.

Module	Message Includes Errors That Happen When
Common	Order Management encounters any one of a wide variety of problems. The problem isn't specific to Order Management. It might happen with other Oracle Applications. For example, communication with a web service fails, you attempt to use an application programming interface that Oracle Applications doesn't support, and so on.

Search is additive. For example, if you set Module to Common and Message Name to DOO_ORCHC_SHIPSET_ADD_CANCEL, then the search doesn't return anything because DOO_ORCHC_SHIPSET_ADD_CANCEL isn't in the Common module.

Use a wildcard. For example, enter DOO_% in Message Name to return all messages that start with DOO_. Here are some abbreviations you might find useful.

Abbreviation and Wildcard	Return Messages For
`FOM_%`	Order Management. FOM is an abbreviation for Oracle Order Management
`DOO_%`	Distributed order orchestration.
`QP_%`	Oracle Pricing.
`CTO_%`	Configured items. CTO is an abbreviation for configure-to-order.
`CZ_%`	Configuration model.
`DOS_%`	Supply Chain Orchestration

- Search Message Name for DOO_ORCHC_SHIPSET_ADD_CANCEL.

2. Edit the message.
   - In the search results, click the **line** that includes DOO_ORCHC_SHIPSET_ADD_CANCEL in the Message Name column, then click **Actions > Edit**.
   - On the Edit Message page, add your text at the end of the User Details attribute. For example:

```
Instead, revise the sales order, cancel lines in the shipment set, then submit the order. Wait
 for Order Management to process the revision. Revise the order again, add lines, then submit the
```

```
order. Hi there. I am Diane Cho, your order administrator. Please contact me if you need help
with shipping.
```

Note

- Don't modify the Message Number. Order Management uses it to uniquely identify the message.
- Examine the Translation Notes. They might contain details that describe when and why a message displays. Use them to help troubleshoot errors or warnings.
- In general, don't modify predefined text in the Short Text attribute. Each message describes an error or warning condition. Its important to keep the predefined Short Text to support these conditions.
- Short Text has a 160 character limit.
- In general, don't modify predefined text in the User Details attribute. If you want to add text, add it immediately after the predefined text.

  - Click **Save**.

3. Test your set up.

   - Go to the Order Management work area in your other browser.
   - If the work area still displays the error message, then close the message dialog.
   - Click Submit.
   - Verify that the message displays the text you added.

Create Your Own Message

Assume you must set up a warning message that displays when the Order Entry Specialist enters a quantity of more than 100 on an order line. The purpose of the message is to make sure the Order Entry Specialist agrees that the quantity is correct before submitting.

1. Go to the Manage Messages page, then search the predefined messages.

   Create a new message only if you can't find a predefined message that meets your needs.
2. Click **Actions > New**, then set the values.

Attribute	Value
Message Name	FOM_EXAMINE_QUANTITY You can enter any text. Use a format that helps you identify your message and distinguish it from predefined messages that Order Management and other Oracle Applications display. `application_text_description` where - **application**. Abbreviation that identifies the application. Use `FOM` for Oracle Order Management, or `DOO` for Distributed Order Orchestration. - **text_description**. Describe the purpose of the message. Use underscores to improve readability.

Attribute	Value
Application	Order Management
Module	Manage Orders Here are the modules you typically use when you set Application to Order Management. o Manage Orders o Import Orders Here are the modules you typically use when you set Application to Distributed Order Orchestration. o Fulfillment Task o Decomposition o Process Order o Orchestration
Message Number	Enter `10000001`. Enter any number from 10,000,000 to 10,999,999. The Order Entry Specialist can use this number when they contact the help desk. At runtime, the message displays the message name, number, and application abbreviation. For example: `fom_examine_quantity-fom-10000001`
Short Text	Enter this value. `Quantity {QUANTITY} that you entered on the order line exceeds 100. You can order this quantity, but make sure it is correct before you submit the sales order.` You use one set of curly brackets ({}) to enclose a token. A token is a placeholder for variable content, such as text or a number. Order Management replaces the token with a value at run time. In this example, it replaces {QUANTITY} with the number that the Order Entry Specialist enters in the Quantity attribute on the order line.
Message Type	Warning
Category	Product
Severity	Low
Logging Enabled	Contains a check mark.

Attribute	Value

3. In the Message Tokens area, click **Actions > New**, then set the values.

Attribute	Value
Token Name	QUANTITY
Data Type	Number
Description	Number that the Order Entry Specialist enters in the Quantity attribute on the order line.

4. Click **Save and Close**, and then click **Save and Close** again.

Set Up Item Substitution in Order Management

Set up substitution in Order Management so you can use a substitute item to fulfill an order line when the preferred item is out of stock.

You can use the Substitute Item action or the Check Availability action in a fulfillment view of the Order Management work area to substitute an item. You can also use Global Order Promising to substitute an item. This topic describes how to use Global Order Promising.

Note

- You use the Product Information Management work area to specify the item to substitute at design time.
- At run time, the Order Entry Specialist creates a sales order in the Order Management work area and clicks Submit.
- Fulfillment fulfills the item.
    - The scheduling fulfillment task in your orchestration process calls Global Order Promising.
    - Global Order Promising communicates with Inventory Management to determine whether the item is out of stock.
    - If its out of stock, and if the Allow Item Substitution attribute on the sales order is Yes, and if Global Order Promising determines that the substitute item is available, then Global Order Promising sends a recommendation to Order Management to use it, and Order Management uses the substitute to fulfill the order line.

Guidelines

- You can use Visual Information Builder to create a pretransformation rule that sets the default value for Allow Item Substitution to No.

  Here's an example rule you can create.

  ```
  If Order Type is equal to Mixed Orders, then Allow Item Substution is set to No
  ```

  For details, see *Use Visual Information Builder*.
- You can't specify substitution when you import a source order. Instead, cancel the existing line and add a new one.
- Don't use substitution to replace an obsolete item. Assume you add memory to the AS54888 and upgrade it to a new item, the AS55000. You have an open sales order that contains the AS54888. Revise the sales order, delete the line that has the AS54888, and add a new line that has the AS55000.
- Each pick-to-order item has a unique set of options that your user sets, making it impossible to stock these variations in inventory. So you can substitute a pick-to-order item for another pick-to-order item, but Order Promising won't consider the substitution when it calculates the promising dates.
- See what you can substitute in a shipment set. For details, see *Ship Order Lines in Shipment Sets*.

Unit of Measure

Make sure the original item and the item that you use to substitute the original have the same unit of measure. Order Management doesn't convert the unit of measure when it does the substitution.

Assume the unit of measure for the AS54888 is Each, but its Dozen for the AS55000. At run time, you submit the sales order, Order Management substitutes the AS55000 for the AS54888, but continues to display Each on the fulfillment line. Your warehouse won't be able to accurately do the substitution because the unit of measure isn't correct. Here's how to avoid this problem.

1. Go to the Product Information Management work area, then click **Tasks > Manage Items**.
2. Open the AS54888 for editing, then notice the value in the Primary Unit of Measure attribute.
3. Open the AS55000 for editing, then make sure the Primary Unit of Measure attribute contains the same value that the AS54888 has for this attribute.

Set Up Your Item

Summary of the Setup

1. Set up your item.
2. Collect data.
3. Test your set up.

Assume you must set up substitution for the AS54888 Computer.

- If the AS54888 Computer isn't available in inventory, then substitute it with the CN92777 Computer.

Try it.

1. Go to the Product Information Management work area.
2. On the Product Information Management page, click **Tasks > Manage Items**.
3. On the Manage Items page, search for your item.

Attribute	Value
Item	AS54888

4. In the Search results, in the Item column, click **AS54888**.
5. On the Edit Item page, click **Relationships**.
6. In the Relationships area, click **Actions > Create**, set the values, then click **OK**.

Attribute	Value
To Item	CN92777 Computer
Type	Substitute
Planning Enabled	Contains a check mark.

7. On the Edit Item page, click **Save**.

Collect Data

1. Go to the Plan Inputs work area, then click **Tasks > Collect Planning Data**.
2. In the Collect Planning Data dialog, set the values.

Attribute	Value
Source System	GPR
Collection Type	Targeted

3. Move the Item Substitution Relationships reference entity to the Selected Entities window, then click **Submit**.
4. In the Status dialog, notice the process number. For this example, assume its 50465.
5. Go to the Scheduled Processes work area.
6. Locate 50465 in the Process ID column. Monitor the process until the Status column displays Succeeded.
7. Refresh the Order Promising server.
    - Go to the Scheduled Processes work area, then click **Action > Schedule New Process**.
    - In the Schedule New Process dialog, set the value.

Attribute	Value
Name	*Refresh and Start the Order Promising Server*

- Add a check mark to the **Items** parameter, then click **Submit**.
- Click **Actions > Refresh**, then verify the status is Succeeded. Repeat, as necessary.

For details, see *Collect Planning Data for Order Management*.

Test Your Set Up

1. Go to the Order Management work area and create a new sales order.
2. Click **Shipment Details > Supply**, then set the value.

Attribute	Value
Allow Item Substitution	Yes

3. Search for the AS54888 on the catalog line, notice that the line displays Out of Stock, then click **Add**.
4. Notice that the order line contains the AS54888, then click **Submit**.
5. Click **Actions > Switch to Fulfillment View**.
6. Verify that Order Management replaced AS54888 with CN92777.

For more, see *How Does Item Substitution Work In Order Management? (Doc ID 2188378.1)*.

Related Topics

- Use Visual Information Builder
- Overview of Collecting Promising Data for Order Management
- Specify Supply Details for Sales Orders

Manage Order Status

Overview

Orchestration Process Status

An orchestration process status indicates the status of an orchestration process throughout Order Management. The value Started is an example status. It indicates that Order Management started the orchestration process.

Order Management finishes the fulfillment tasks of an orchestration process sequentially according to each orchestration process step when it processes your sales order.

Order Management comes predefined to use a default set of statuses for fulfillment tasks. You can also set up statuses and sequences. For example, you can set up an orchestration process that uses a set of statuses and rule logic for textbooks for a college, and set up another orchestration process that uses a different set of statuses and rule logic for textbooks for a primary school.

- You must specify the status that Order Management assigns to an orchestration process at each orchestration process step. For example, if the Schedule School Books task includes a status of Unsourced, then you must specify the status that Order Management assigns to the orchestration process for this step.

- You must specify a status that indicates when a task done. You can select only a status that you set up to indicate that a task is done.
- If you modify the name of the default status, then Order Management displays the modified name throughout the Order Management work area.
- If you don't set up statuses for an orchestration process, then Order Management uses the predefined statuses, by default.

Status Conditions for Orchestration Processes

You can create a status condition that determines the orchestration process status. For example:

- If the status of the Schedule task is Not Started, then set the orchestration process status to Unscheduled.

Order Management evaluates the status conditions that you create sequentially at runtime. The condition that evaluates to true, and that includes the highest sequence number, determines the orchestration process status.

How Order Management Assigns Statuses

1. A fulfillment system sends a status update to Order Management.
2. The interface translates the status into status values that Order Management uses.
3. Fulfillment tasks determine the status message to send.
4. The Status Service.
    - Uses source and target mapping to set the task status.
    - Sets the orchestration process status according to the statuses of the tasks that the process contains.
    - Sets the fulfillment line status according to the task statuses that the fulfillment line references.

Modify Status for Predefined Shipment Tasks or Invoice Tasks

You must not modify the status value for the orchestration process or the fulfillment line for a predefined shipment task or invoice task when that status is the final status that Oracle Shipping or Oracle Accounts Receivable sends to Oracle Order Management. If you do modify it, then you might encounter an error when you release the orchestration process.

Instead, you must use the predefined condition for the status value that the DOO_OrderFulfillmentGenericProcess orchestration process uses. To see that status, go to the Manage Orchestration Process Definitions task in the Setup and Maintenance work area, open the DOO_OrderFulfillmentGenericProcess orchestration process for editing, then examine the values in the Status Conditions area.

You must use these predefined values.

Status Value	Expression
Shipped	`SHIP = SHIPPED` where • SHIP is the task for the shipment task type. • SHIPPED is the fulfillment status that Shipping sends to Order Management.
Billed	`Invoice = BILLED`

Status Value	Expression
	where • Invoice is the task for the Invoice task type. • BILLED is the fulfillment status that Accounts Receivable sends to Order Management.

Related Topics

- Group Statuses for Orchestration Processes
- Set Up Orchestration Processes
- Add Status Conditions to Orchestration Processes
- Order Management Statuses

Manage Status Values

Use tabs on the Manage Status Values page to manage how Order Management displays status.

Tab	Description
Status Codes	Specify the display name that Order Management displays in the Order Management work area.
Fulfillment Lines	Create the status values that users can select for a fulfillment line in an orchestration process.
Task Types	Assign a status code to a task type. If a system outside of Oracle Order Management provides the status, then assign this status to the fulfillment task that references this system. For example, if a shipping system provides the status, then assign the code to the Shipping task type.
Orchestration Process Classes	Assign the status code to fulfillment lines or orchestration processes where Order Management must use the status. For example, you can use an orchestration process class to control the status codes that an administrator can choose when creating a status condition for an orchestration process. You must use the Manage Status Values page to set the status values and to make them available when you create a status condition.

Related Topics

- Group Statuses for Orchestration Processes
- Example of Integrating Order Management with Your Fulfillment System
- Order Management Statuses

Set Ups

Manage Task Status Conditions

Order Management fulfills each fulfillment task one step at a time when it processes a sales order, and it uses a predefined set of sequential statuses to track the progress of each task. If the predefined status conditions don't meet your needs, you can create your own.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Task Status Conditions
2. On the Manage Task Status Conditions page, in the task list, click the **task type** that you must manage.

    For example, to manage the status conditions when Order Management processes a return, click the **row** that contains Return in the Type column.
3. In the Status Conditions list, modify or add new conditions as necessary.

Attribute	Description
Internal Status Value	Displays a status that Order Management receives from a fulfillment system through a fulfillment task. For details, see *Create Your Own Task Type*.
Display Status Value	Specifies the value that Order Management displays. For example, assume your fulfillment system sends a status of Invoiced for a task, but your company uses Billed. You can set Display Status Value to Billed to display Billed throughout the Order Management work area.
Mark as Complete	If this option contains a check mark, then Order Management considers the task to be done when it reaches the condition. For example, assume you click the Return row in the task list on the Manage Task Status Conditions page, then, in the Status Conditions area, you add a check mark to Mark as Complete for the Canceled status condition and for the Delivered status condition. In this example, Order Management considers the task to be done when the task reaches the Canceled status or the Delivered status.

Related Topics

- Task Services

Group Statuses for Orchestration Processes

An orchestration process class is a set of statuses you can assign to an orchestration process. Use it to group statuses so they're meaningful.

For example, the Ship Order Class includes statuses that are meaningful when shipping an order, such as Scheduled, Awaiting Shipping, and Shipped.

This topic uses example values. You might need different values, depending on your business requirements.

Group statuses for orchestration processes.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Status Values
2. On the Manage Status Values page, click **Orchestration Process Classes**.
3. In the Orchestration Process Classes list, click **Actions > Create**, then set the values.

Attribute	Value
Code	Standard_Class
Name	Standard_Class
Description	Class for standard orders

4. Click **Save**.
5. In the Status Values area, click **Actions > Add Row**, then add the value.

Attribute	Value
Status Value	Scheduled

6. Repeat step 5 for each value.
    - Shipped
    - Reserved
    - Billed
    - Returned to Customer
7. Click **Save and Close**.

You can now set class Standard_Class in the Process Class attribute on the Edit Orchestration Process Definitions page. The class that you set determines the values you can select when you set the Status Value on the Status Conditions

tab. For example, if you set Process Class to Standard_Class, then you can use Status Value to select the statuses that Standard_Class references, such as Shipped.

Related Topics

- Roadmap for Setting Up Order-to-Cash
- Orchestration Process Status
- Set Up Orchestration Processes
- Order Management Statuses

Fulfillment Line

Fulfillment Line Status

Specify the status that your orchestration process assigns to a fulfillment line.

For example, you can specify that if the Schedule School Books task includes a status of Pending Scheduling, then set the fulfillment line status to Unscheduled.

- The orchestration process step runs fulfillment tasks when it processes the fulfillment line.
- The fulfillment line status displays throughout the Order Management work area.
- You can specify different sets of statuses and rules for different items that the fulfillment line references. For example, specify one set of status conditions for a hard cover book, and another set of status conditions for a paperback book.
- If you don't create conditions, then Order Management uses the status rule set that it assigns to the default category.

Use Status Catalogs and Status Categories to Group Fulfillment Lines

Use a status catalog to group items that are similar so they can achieve the same statuses at the same time.

You might need different fulfillment lines to use different sets of statuses. For example, a fulfillment line you ship, such as a computer, might need statuses that are different from a fulfillment line you don't ship, such as a warranty.

- Use a category to make sure Order Management applies the same set of status conditions to specific sets of fulfillment lines. It applies the same status conditions to all fulfillment lines that reference the item that resides in the category.
- Use catalogs and categories in more than one orchestration process.
- Select the status catalog when you set up your orchestration process.

  You can only select a catalog that meets these requirements.
  - The item exists in only one category in the catalog.
  - The category contains items **or** subcategories.
  - The category doesn't contain items **and** subcategories.
  - Order Management controls the catalog only for the master, and not for each organization.
- Use the Product Information Management work area to set up the status catalog.

Use Status Rule Sets with Fulfillment Lines

Use a status rule set so you can use a single set of rules with more than one fulfillment line instead of specifying a separate rule for each fulfillment line. For example:

- If the item is in status Unsourced, then set the fulfillment line status to Unscheduled.
- If the Schedule Text Books fulfillment task reaches a status of Completed, then set the fulfillment line status to Scheduled.

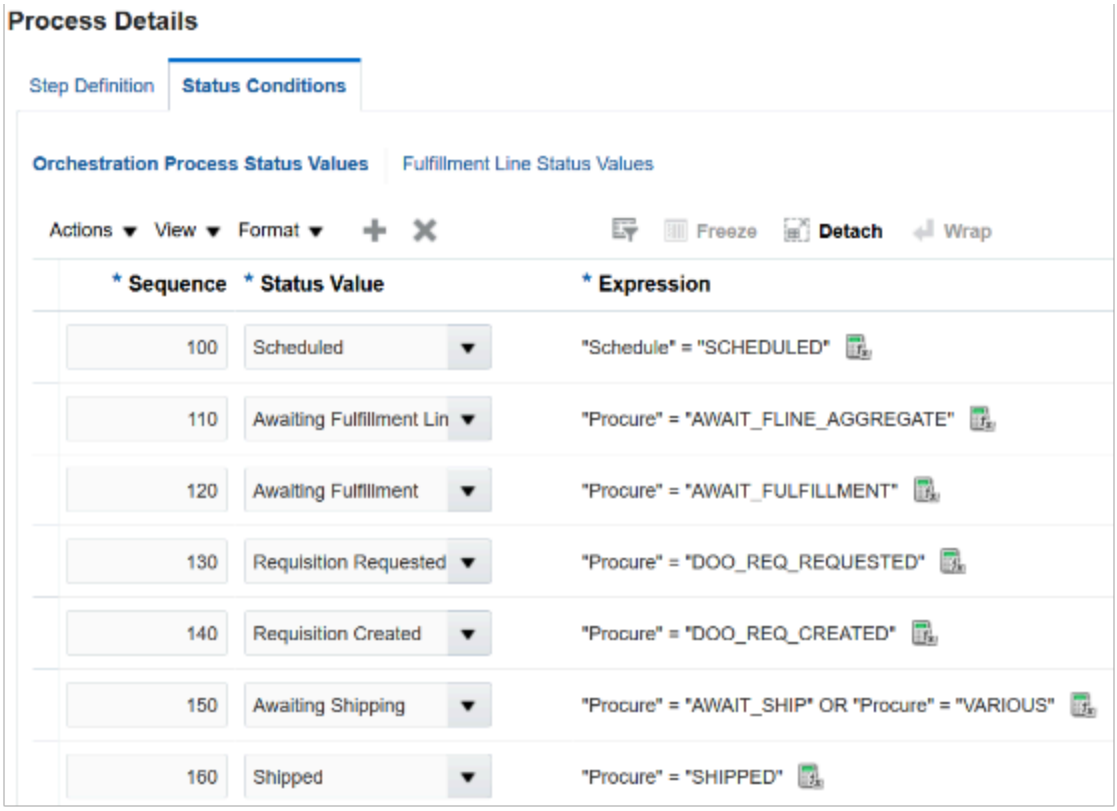
Note

- You can use a single status rule set with more than one category. If a parent category and a child category each reference a different status rule set, then Order Management uses the status rule set that the child references. Use this feature so you can create an All category to handle all items in one orchestration process. Use it to add a subcategory for a subset of items that must use a different status rule set.
- If you migrate an orchestration process between environments, such as from a development environment to a production environment, then don't modify the status rule set name in either environment. Modifying the name might prevent Order Management from updating references to other data in the orchestration process.

Set Up the Sequence for the Status Condition

Set up the sequence of status conditions so they match the logical progress of the status that the order line normally takes during the fulfillment line lifecycle. If more than one condition evaluates to true, then the orchestration process will set the fulfillment line status to the status value with the highest sequence number.

Here's an example.



Note

- You use the Sequence attribute in the Orchestration Process Status Values area on the Status Conditions tab of the Create Orchestration Process Definition page.
- You set the sequence of the status conditions so they match the logical progress of the status. For example, Sequence 150 for status Awaiting Shipping occurs before Sequence 160, status Shipped. If Awaiting Shipping and Shipped each evaluate to true, then the orchestration process sets the status to Shipped because 160 is higher than 150.

Related Topics

- Roadmap for Setting Up Order-to-Cash
- Add Status Conditions to Fulfillment Lines
- Order Management Statuses

Set Up Fulfillment Line Status

Set up the status values that Order Management displays on the fulfillment line.

Summary of the Set Up

1. Add the status codes.
2. Add status codes to fulfillment lines.

This topic uses example values. You might need different values, depending on your business requirements.

Add the Status Codes

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Status Values
2. On the Manage Status Values page, in the Status Codes list, click **Actions > Create**.
3. In the Create Status Code dialog, enter values, then click **Save and Create Another**.

Attribute	Value
Code	SCHED_GOODS
Name	Scheduled Goods The Order Management work area will display the value you enter.

4. In the Create Status Code dialog, enter values, then click **Save and Close**.

Attribute	Value
Code	SCHED_CLOTHING
Name	Scheduled Clothing

Add Status Codes to Fulfillment Lines

Add the status codes you just created to the fulfillment lines.

1. On the Manage Status Values page, click **Fulfillment Lines > Actions > Select and Add**.
2. In the Select and Add dialog, in the Status Code attribute, enter `SCHED_GOODS`, then click **Search**.
3. Click **SCHED_GOODS > OK**.
4. Click **Actions > Select and Add**.
5. In the Select and Add dialog, in the Status Code attribute, enter `SCHED_CLOTHING`, then click **Search**.
6. Click **SCHED_CLOTHING > OK**.
7. Click **Save and Close**.

You can now set the status value for the fulfillment line in the orchestration process to SCHED_GOODS or SCHED_CLOTHING.

Related Topics

- Fulfillment Line Status
- Roadmap for Setting Up Order-to-Cash
- Order Management Statuses

Status Condition

Add Status Conditions to Orchestration Processes

Add status conditions that specify when to set the status of an orchestration process.

Assume you need an orchestration process that fulfills sales orders for company t-shirts, and you must specify the statuses that the process uses throughout the order life cycle according to the status of the fulfillment task.

You will create this status condition.

- If the status of the Schedule task is Scheduled, then set the orchestration process status to Scheduled.

Summary of the Set Up

- Set the orchestration process class.
- Add the status condition.

You typically add more than one status condition to an orchestration process. For brevity, you will add only one in this topic.

This topic uses example values. You might need different values, depending on your business requirements.

Set the Orchestration Process Class

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.
   Learn how to create this process. For details, see *Add Branches to Orchestration Processes*.
3. On the Edit Orchestration Process Definitions page, set the value.

Attribute	Value
Process Class	Standard_Class You must set up Standard_Class before you can select it. For details, see *Set Up Orchestration Processes*.

4. Click **Save**.

Add the Status Condition

1. In the Process Details area, click **Status Conditions**.
2. In the Orchestration Process Status Values list, click **Actions > Add Row**.
3. In the new row, set the values.

Attribute	Value
Sequence	1
Status Value	Scheduled

4. Add the expression.
    - In the new row, in the Expression column, click the icon.
    - In the Expression Builder dialog, click **CallCustomerWhenLargeInvoice, Schedule**, then click **Insert Into Expression**. Make sure you don't expand CallCustomerWhenLargeInvoice, Schedule.
      Notice that the dialog added a value of `"Schedule"` in the Expression window of the Expression Builder.
    - In the Expression window, click anywhere after `"Schedule"`, then enter an equal sign (=).
    - Expand **CallCustomerWhenLargeInvoice, Schedule**, click **SCHEDULED**, then click **Insert Into Expression**.

- Notice that the Expression window contains `"Schedule"="SCHEDULED"`, then click **OK**.

5. Click **Save**.

Related Topics

- Add Branches to Orchestration Processes
- Group Statuses for Orchestration Processes

Add Status Conditions to Fulfillment Lines

Add status conditions to a fulfillment line that includes more than one item, and where each item needs a different status.

If different fulfillment lines must use different statuses, then you must determine how you will use catalogs and categories to group the lines. You do this when you create the orchestration process.

Assume you must set up orchestration processes that can handle sales orders for different types of t-shirts. You could use the same orchestration process for different types of merchandise, but you prefer to define statuses for each type of clothing separately because each clothing type requires a different status. To do this, you select the status catalog, then add the status conditions for a single category of items in the orchestration process.

Summary of the Set Up

- Set the status catalog.
- Add the status condition for each default.
- Add the status condition for the item.

You typically create more than one status condition for each fulfillment line. For brevity, in this topic you create only one status condition for the default category and another status condition for the item.

This topic uses example values. You might need different values, depending on your business requirements.

Set the Status Catalog

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.
   Learn how to create this process. For details, see *Add Branches to Orchestration Processes*.
3. On the Edit Orchestration Process Definitions page, set the value.

Attribute	Value
Status Catalog	Retail_Merchandising_Catalog Retail_Merchandising_Catalog is an example catalog. You must use the Product Information Management work area to define it and the values that it references before you can specify it for this orchestration process.

Attribute	Value

If a warning dialog displays, then click **OK**.

4. Click **Save**.

Add the Default Status Condition

Add the status condition that the orchestration process will use, by default.

1. In the Process Details area, click **Status Conditions > Fulfillment Line Status Values**.
2. In the Default row, in the Status Rule Set attribute, click the **arrow**, then click **Create**.

   You will select the statuses that this orchestration process uses, by default. Every row in the Fulfillment Line Status Values list must reference a status rule set, and each rule set must contain at least one status condition.
3. In the Create Status Rule Set dialog, set the values.

Attribute	Value
Code	LargeOrders
Name	Large Orders
Create New	Chosen

4. Click **Save and Close > Save**.
5. In the Default row, click **Edit Status Rule Set.**
6. On the Edit Status Rule Set page, click **Actions > Add Row**.
7. In the Sequence attribute, enter `1`.
8. Set the status value.
   - In the Status Value attribute, click the **arrow**, then click **Search**.
   - In the Search and Select dialog, in the Status Code attribute, enter `SCHED`, then click **Search**.
   - In the list, click **Scheduled**, then click **OK**.
9. Add the expression.
   - On the Edit Status Rule Set page, in the Expression column, click the icon.
   - In the Expression Builder dialog, click **CallCustomerWhenLargeInvoice, Schedule**, then click **Insert Into Expression**. Make sure you don't expand CallCustomerWhenLargeInvoice, Schedule.

     Notice that the dialog added a value of `"Schedule"` in the Expression window of the Expression Builder.
   - In the Expression window, click anywhere after `"Schedule"`, then enter an equal sign (=).
   - Expand **CallCustomerWhenLargeInvoice, Schedule**, click **SCHEDULED**, then click **Insert Into Expression**.
   - Notice that the Expression window contains `"Schedule"="SCHEDULED"`, then click **OK**.

10. On the Edit Status Rule Set page, add a check mark to Notify External Systems, then click **Save and Close**.

    Notify External Systems allows Order Management to communicate the status to a system that resides outside of Order Management. For details, see *Send Notifications from Order Management to Other Systems*.

Add the Status Condition for an Item

1. In the Process Details area, in the Fulfillment Line Status Values list, click **Actions > Select and Add: Category**.
2. In the Select and Add: Category dialog, select **Retailer**, then click **Save and Close**.
3. In the Fulfillment Line Status Values list, in the Retailer row, in the Status Rule Set column, click the **arrow**, then click **Create**.
4. In the Create Status Rule Set dialog, set the values.

Attribute	Value
Code	Shirts
Name	Shirts
Create New	Chosen

5. Click **Save and Close > Save**.
6. In the Fulfillment Line Status Values list, in the Retailer row, click **Edit Status Rule Set.**
7. Repeat steps 6 through 10 of the Add the Default Status Condition section, earlier in this topic.

Related Topics

- Add Branches to Orchestration Processes
- Send Notifications from Order Management to Other Systems
- Order Management Statuses

Manage Order Values

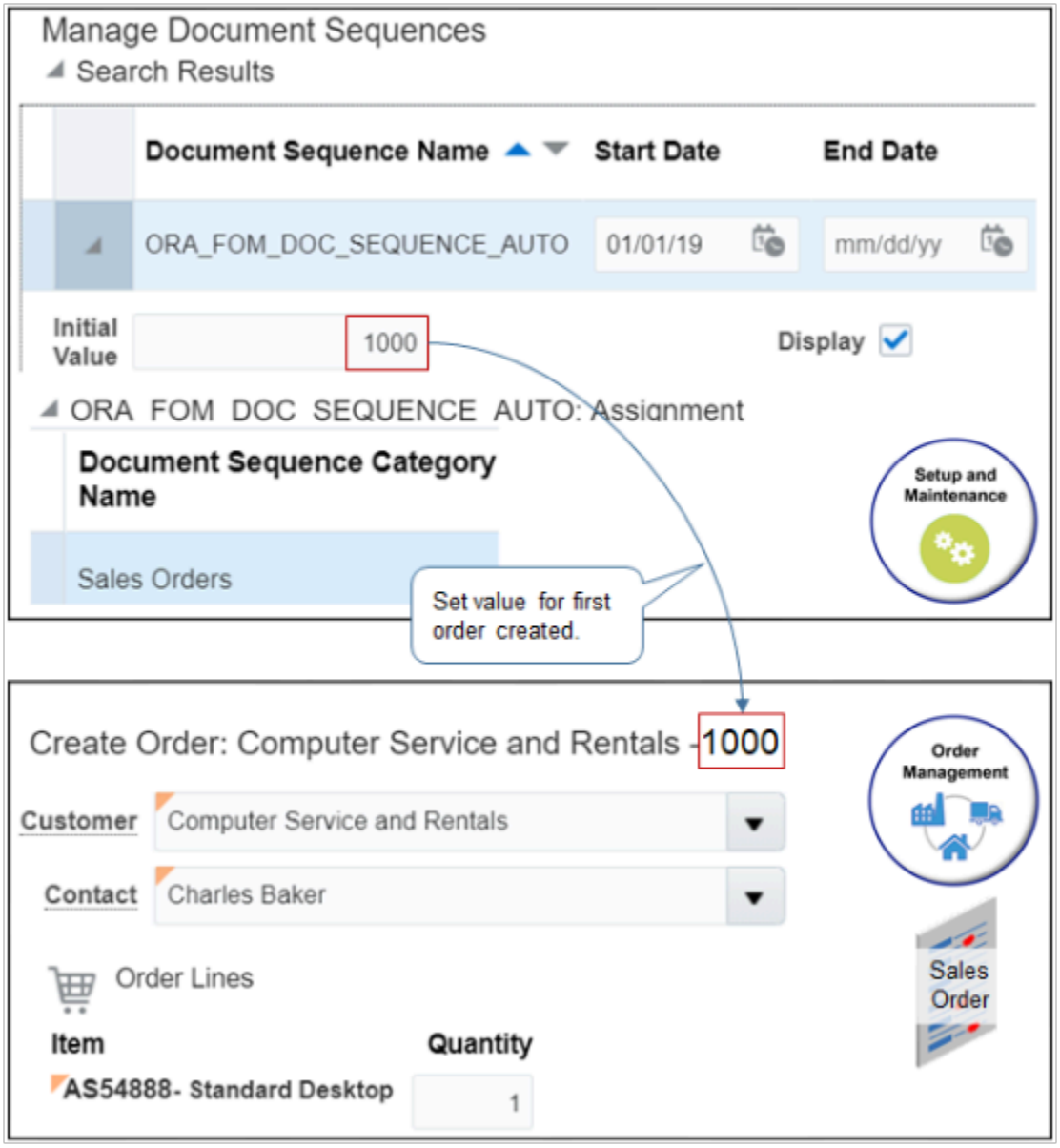
Order Numbers

Set Up Sequences for Sales Order Numbers

Set up the starting value for your sales order numbers.

In this example, you set up Order Management so it uses 1,000 as the order number for the first order that your users create on January 1, 2019.

Here's your setup.



This topic uses example values. You might need different values, depending on your business requirements.

1. In the Setup and Maintenance work area, click **Tasks > Search**, search for, then open Manage Document Sequences.
2. On the Manage Document Sequences page, search for the value.

Attribute	Value
Document Sequence Name	ORA_FOM_DOC_SEQUENCE_AUTO

3. In the search results, click **Expand**, set the values, then click **Save and Close**.

Attribute	Value
Initial Value	1000
Display	Ignore this attribute. Don't set it. It's for Oracle use only.
Start Date	01/01/19
End Date	Leave empty. Order Management will continue to increment the sales order number in perpetuity.

Note

- Order Management uses the predefined ORA_FOM_DOC_SEQUENCE_AUTO number sequence to create order numbers, starting with the value that you set in Initial Value.
- Each order number is unique.
- The order number is permanent. Order Management doesn't change the order number throughout fulfillment.
- If you import orders from a source system through a web service, REST API, a file, or any other way, and if you set the DOO_RT_USE_ORDER_NUMBER order profile to Y, then Order Management uses order numbers from your source system and ignores ORA_FOM_DOC_SEQUENCE_AUTO. For details about this profile, see *Keep Your Order Number When You Import*.

You can't:

- Use your own number sequence, except for order numbers that you import.
- Specify sequence according to determinant type.
- Specify gapless or manual sequencing.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click Sales Orders for Order Hub.

Related Topics

- Use Order Profiles to Control Order Management Behavior
- Document Sequences
- Guidelines for Managing Document Sequences

Keep Your Order Number When You Import

Control how Order Management sets the order number when you import a source order.

Use this profile option:

Profile Option Code	Value
DOO_RT_USE_ORDER_NUMBER	Select one:

Profile Option Code	Value
	• **Y**. Order Management will use the number that you import from your source system as the order number. Order Management will display the number that you import throughout the Order Management work area after transformation and during order fulfillment. • **N**. Order Management won't use the number from your system, but will instead create and assign a new order number for the order that you import. The default value is N. You can use this option only for a single site.

Use the Number From Your Source System

For example, you might want to use an entirely different set of identifying numbers for your sales orders. Assume you use your source system to log service requests from your customers, and then import each request into Order Management where you can manage it as a sales order. You don't want to use Order Management's order numbers, but instead want to use the service request number as the order number in Order Management. Assume you need to import service request 76546286. Here's how you can meet that requirement:

1. Set DOO_RT_USE_ORDER_NUMBER to Y.
2. Include 76546286 in the SourceOrderNumber attribute in your import payload.
3. Import your source orders.

During import, Order Management will set the sales order's Order Number attribute to 76546286, and will use 76546286 as the order number throughout fulfillment.

Don't Use the Number From Your Source System

If you set DOO_RT_USE_ORDER_NUMBER to N, then Order Management doesn't use your source order number as the order number during or after transformation.

- Order Management will assign a new order number for the source order and display the new number throughout the Order Management work area after transformation and during order fulfillment.
- Order Management creates a unique number for each sales order that you create in the Order Management work area, and each of these orders use Order Orchestration and Planning (OPS) as the source system.

  If you use OPS as your source system, and you if set DOO_RT_USE_ORDER_NUMBER to Y, then a conflict might happen between order numbers.

  For example, your source system might create a source order with order number 543865, and an order that you create in the Order Management work area might also use order number 543865. You must make sure these order numbers are unique. For example, use a value for your source order number that isn't numeric, or add a prefix or suffix to the number, such as src_543865.

If the source order number in the source order that you import from the source system isn't unique in the source system, then you must set this option to N. This is necessary to avoid an error when you import orders because Order Management requires a unique order number for each sales order in the same source system. For example, if source system A contains:

- Two source orders that each use order number 12345, then set DOO_RT_USE_ORDER_NUMBER to N.
- One source order that uses order number 12345, and if source system B contains one source order that uses order number 12345, then you can set DOO_RT_USE_ORDER_NUMBER to Y or N.

Integrate with Accounts Receivable

You might encounter a situation where the order number in Accounts Receivable isn't the same as the order number in Order Management. This might cause problems, such as when you do AutoInvoice during file-based data import in Accounts Receivable.

Order Management uses the RA_INTERFACE_LINES_ALL table to integrate with Accounts Receivable.

Order Management uses the ORDER_NUMBER column and SOURCE_ORDER_NUMBER column on the DOO_HEADERS_ALL table to populate columns in RA_INTERFACE_LINES_ALL.

Column in RA_INTERFACE_LINES_ALL	What Does it Store
INTERFACE_LINE_ATTRIBUTE1	SOURCE_ORDER_NUMBER
INTERFACE_LINE_ATTRIBUTE3	ORDER_NUMBER
SALES_ORDER	SOURCE_ORDER_NUMBER

Order Management sets the ORDER_NUMBER and SOURCE_ORDER_NUMBER in the DOO_HEADERS table to the same value that it uses in the Order Management work area.

However, if the values that you import are different than what Order Management uses, then you have two choices to make sure you use the same order number in Order Management and in Accounts Receivable:

- Set the DOO_RT_USE_ORDER_NUMBER profile to Y, and Order Management will set ORDER_NUMBER to the same value that SOURCE_ORDER_NUMBER contains.
- Use a document sequence to set the value in ORDER_NUMBER. For details, see *Set Up Sequences for Sales Order Numbers*.

Answers to Some of Your Questions About Sales Order Numbers

- You can't use different order number sequences for different order types or different source systems. For example, you can't use sales orders 1000 through 10000 for order type A, and sales orders 10001 through 20000 for type B.
- Order Management creates order numbers from a database sequence that makes sure order numbers are unique, but the sequence might contain gaps because of caching and other processing anomalies. You can't change this behavior.
- You can't assign an order number from one draft order to another draft order, even if you delete the original draft because deleting the draft will result in gaps in the order number sequence. Order Management assigns the order number for a draft order and the order details in the DOO_HEADERS_ALL table. If you delete the draft, then Order Management also deletes the entry for the draft from DOO_HEADERS_ALL.

You can use this SQL to identify the line numbers that Order Management assigned to an order line and the fulfillment line:

```
SELECT dha.header_id ,
 dha.order_number ,
 dha.source_order_number ,
 dha.submitted_flag,
 dla.line_number ,
 dla.SOURCE_LINE_NUMBER ,
```

```
 dla.SOURCE_LINE_ID,
 dla. DISPLAY_LINE_NUMBER ,
 dfla.SOURCE_LINE_NUMBER ,
 dfla.fulfill_line_number
FROM fusion.doo_headers_All dha,
 fusion.doo_lines_all dla ,
 Fusion.doo_fulfill_lines_all dfla
WHERE dha.source_order_number in ('&SOURCE_ORDER1','&SOURCE_ORDER2')
 AND dla.header_id = dha.header_id
 AND dfla.line_id = dla.line_id
--and submitted_flag = 'Y'
order by dha.SOURCE_ORDER_NUMBER , dla.LINE_NUMBER
```

Related Topics

- Integrate Order Management with Source Systems
- Roadmap for Setting Up Order-to-Cash
- Use Order Profiles to Control Order Management Behavior

Attribute Values

Control Decimal Precision

Use the Quantity Decimal Precision profile to control decimal precision for attributes that store numeric values.

For example, to control decimal precision on the order line quantity that the Order Entry Specialist sets on the Create Order page, or to control decimal precision during order import.

Decimal precision is the total number of digits that a number contains, where **scale** specifies the number of digits that exist to the right of the decimal point.

For example, the number 17.347 uses a precision of 5 and a scale of 3.

You can't modify precision but you can modify scale. For example, if you set the scale to 3 in the Quantity Decimal Precision profile, and if the Order Entry Specialist enters 17.3468 for the quantity, then the profile will round 17.3468 to 17.347.

Quantity for some items is typically a whole number. For example, most companies sell a desktop computer as a whole item. Don't use Quantity Decimal Precision to allow only a whole number in the quantity. Instead, you can set the Indivisible attribute to Yes in the Product Information Management work area when you set up the item.

Here's your scenario.

- You sell a variety of elements from the periodic table according to weight, such as Diamond.
- You sell these elements in grams and milligrams, which is 1/1000 of a gram.
- The value for milligram can range from 0.001 to 0.999.
- You already set up an item named Diamond in the Product Information Management work area, and you specified it to use Gram as the unit of measure.
- You will set the scale to 3.

Try it.

1. Go to the Setup and Maintenance work area, then click **Tasks > Search**.
2. On the Search page, search for, then open Manage Administrator Profile Values.

For details about this task, see *Implementing Common Features for Oracle SCM*.

3. On the Manage Administrator Profile Values page, in the Search area, enter the value, then click **Search**.

Attribute	Value
Profile Display Name	Quantity Decimal Precision

4. In the search results, in the Profile Values area, set the value, then click **Save and Close > Done**.

Attribute	Value
Profile Value	3

You can set decimal precision only at the Site level. You can't set it for each item or for each unit of measure.

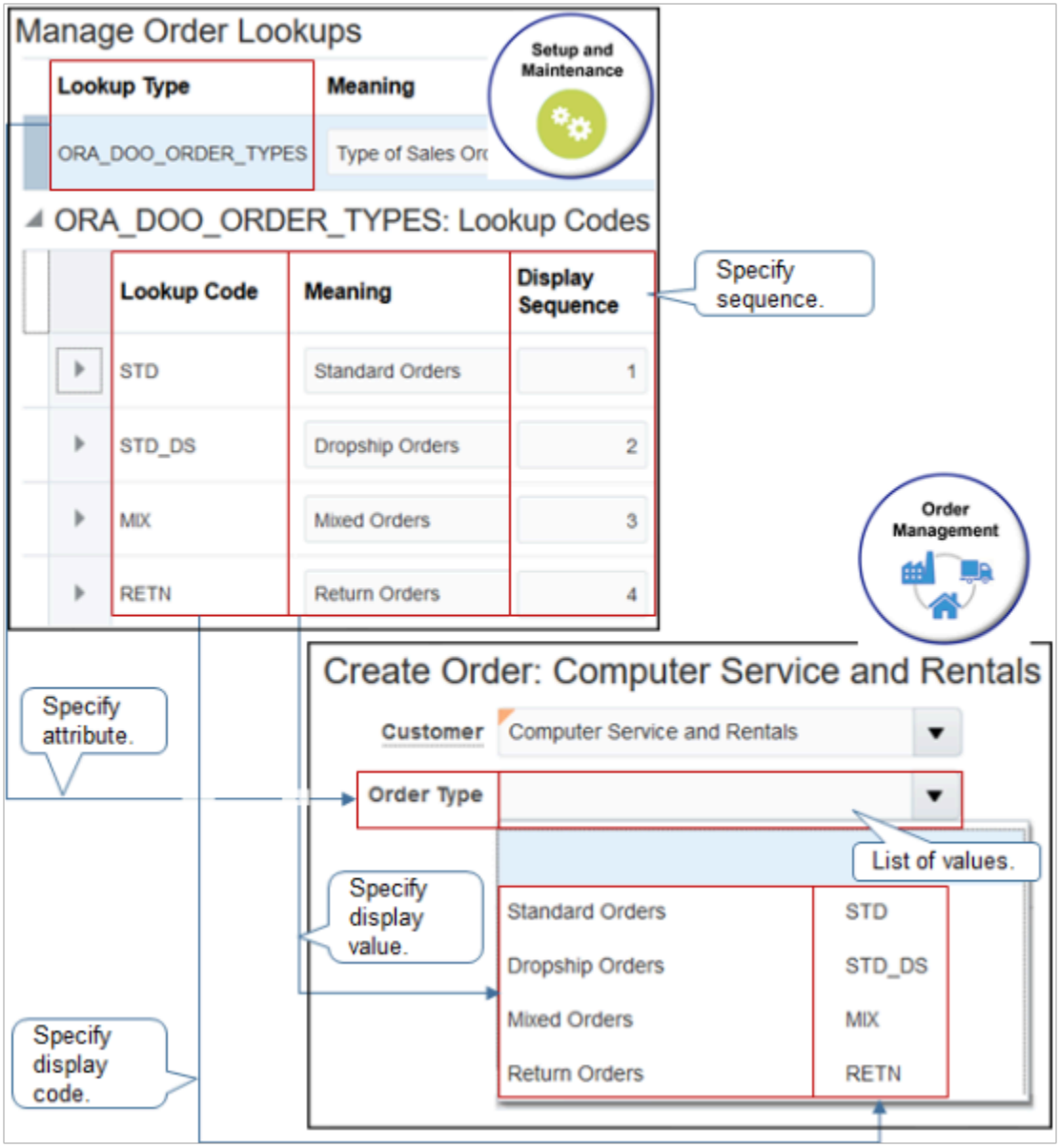
5. Go to the Order Management work area, then click **Create Order**.
6. On the Create Order page, in the Select Item window, enter `Diamond`, then click **Search**.
7. On the catalog search line, in the quantity window to the left of Gram, set the value to 17.3468, then click **Add**.
8. Verify that the Create Order page displays a message that it rounded your value to 17.347.

Related Topics

- Roadmap for Setting Up Order-to-Cash
- Manage Rounding Rules

Manage Lookups in Order Management

An order lookup specifies the values that Order Management displays in a list of values. You can set up these values.



A list of values is a user interface element that allows your users to select the value for an attribute. For example, the user can use a drop down list of values to set the value for the Order Type attribute on the order header. You can manage these values.

Assume you need to modify the sequence of values that display when the user clicks the Order Type attribute on the order header in the Order Management work area. You need to display Drop-Ship Orders as the first value in the list.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups
2. On the Manage Order Lookups page, don't enter any values in the search section, then click **Search**.
3. Examine the search results. It contains all the order lookups that come predefined with Order Management and any custom ones that you have added.

4. In the search section, enter the value, then click Search.

Attribute	Value
Lookup Type	ORA_DOO_ORDER_TYPES This attribute specifies a group of values. For example, ORA_DOO_ORDER_TYPES allows the user to select from a list of values in the Order Type attribute on the order header.

5. Examine these attributes.

Attribute	Description
Lookup Code	Abbreviation that represents the meaning. Each code uses one value that the user can select in the list of values.
Meaning	User-friendly value that displays in the list of values. For example, a table in the Oracle database or a web service payload might contain a value of STD_DS for a standard drop ship sales order. You can set Meaning to the more user-friendly value Drop-ship Order. See how a web service payload can contain coded values. For details, see *Attributes That You Can Use with Web Services*.
Display Sequence	Sequence of the lookup codes in the list of values. For example, if Sequence is 1 for Standard Orders, 2 for Drop-ship Orders, and 3 for Mixed Orders, then the list of value displays values in this sequence. o Standard Orders o Drop-ship Orders o Mixed Orders

6. Modify the values.

Display Sequence	Meaning
1	Drop-ship Orders
2	Standard Orders
3	Mixed Orders

7. Click **Save and Close**.

Use REST API to Specify Lookup Values

You can specify lookup values in your REST API payload. For example, you might need to set the value to Insufficient Supply for the DOO_SUBSTITUTION_REASON lookup.

The lookups that you can use in your payload depend on the value in the REST Access Secured attribute on the Manage Order Lookups page, such as Anonymous, Authenticated, or Secure. For details, see *Overview of Lookups*.

Related Topics

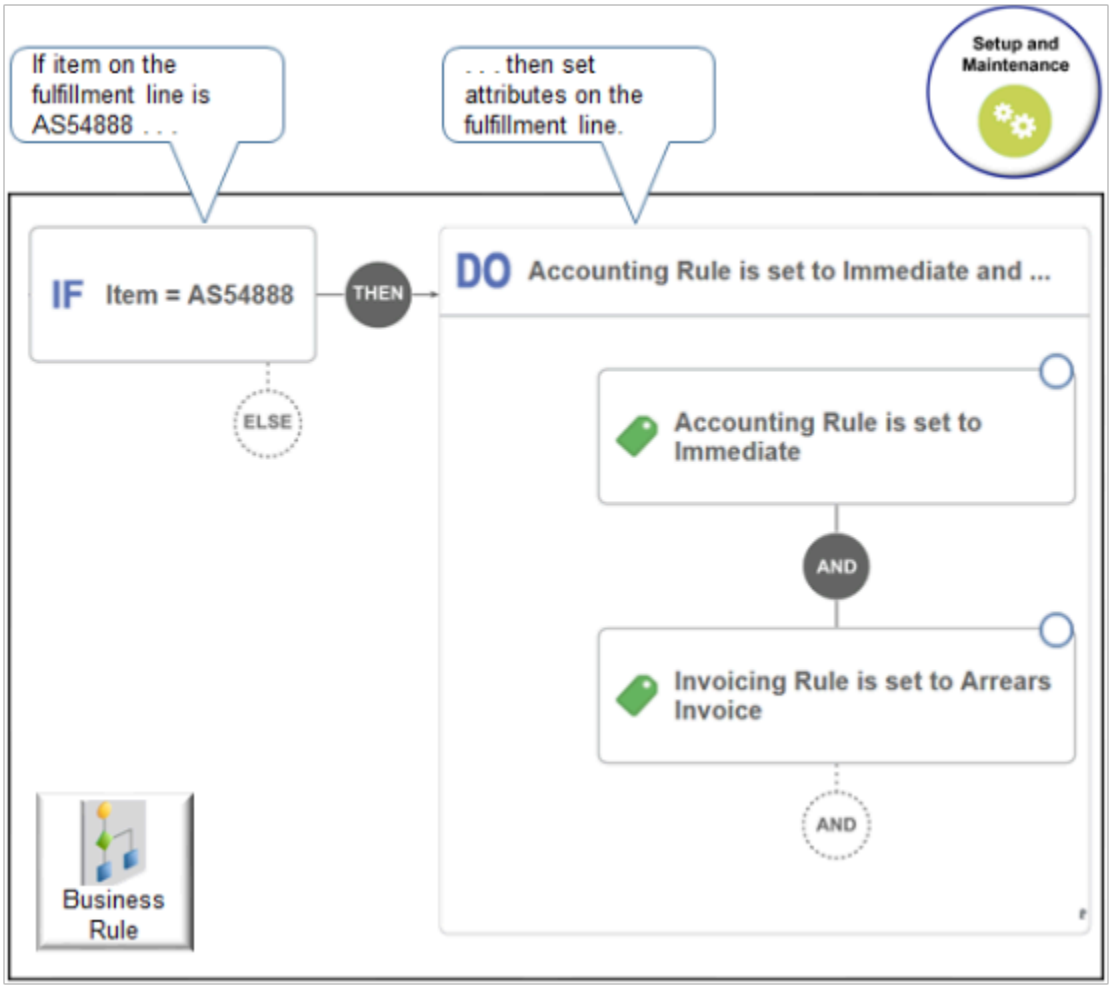
- Attributes That You Can Use with Web Services

Set Attribute Values Before You Transform Source Orders

Create a business rule that sets the default value for an attribute on a source order before you transform the order.

You can an attribute value directly in a business rule. For example, assume you need a rule.

```
If the Item attribute contains AS54888, then set the Accounting Rule attribute to Immediate, and the
 Invoicing Rule to Arrears Invoice.
```



Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Pretransformation Rules for Sales Orders
2. Create a new rule.

Attribute	Value
Name	Set Default Values for Accounting and Invoicing Rules

3. Create the If statement.
    - In the If area, click **New Condition**.
    - In the Create Condition dialog, enter `item`, wait a moment, then click **Item (Order Fulfill Line)**.

      Item (Order Fulfill Line) indicates that you're referencing the Item attribute on the fulfillment line of the sales order.
    - Click **Search**.
    - In the Search dialog, search for AS54888, click **AS54888** in the search results, then click **OK**.
    - In the Create Condition dialog, click **OK**.

```
IF item = AS54888
```

4. Click **Then > Do**.
5. Create the action for the Accounting Rule attribute.
    - In the DO area, click **New Action > Set a Value**.
    - In the Create Action Dialog, enter `Accounting Rule`, then click **Accounting Rule (Order Fulfill Line)**.
    - Click **Search**.
    - In the Search dialog, enter the value, then click **Search**.

Attribute	Value
Name	% % is a `wild card` that means to return all possible values.

Attribute	Value

The search result displays all the values that you can use for the attribute you entered, Accounting Rule.

- In the search results, click the row that contains 12 Months Fixed, then click **OK**.
- In the Create Action dialog, notice the value.

```
Accounting Rule Order Fulfill Line is set to 12 Months Fixed
```

- Click **OK**.

  At run time, if the Item attribute on the order line contains AS54888, then Order Management will set the Accounting Rule attribute to 12 Months Fixed.

6. Repeat step 5 but this time create an action for the Invoicing Rule attribute.

```
Invoicing Rule Order Fulfill Line is set to Arrears Invoice
```

Test Your Setup

1. Go to the Order Management work area and create a sales order.
2. Add the AS54888 to an order line.
3. Click **Billing and Payment Details**.
4. On the order line, in the Item column, click the **down arrow**, then click **Edit Accounting Details**.
5. In the Edit Accounting Details dialog, verify the values.

Attribute	Value
Accounting Rule	12 Months Fixed
Invoicing Rule	Arrears Invoice

Related Topics

- Guidelines for Assigning Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules

Set Attribute Values After You Transform Source Orders

Set the default value for an attribute after you transform the source order.

You can use Visual Information Builder to set the default value before you transform, but you must use Oracle Business Rules to set the value after you transform the source order.

Assume that if the order type is Standard orders, then you must set the default value for the legal entity, and you must do it after you transform the source order.

Summary of the Setup

1. Get values for your attributes.
2. Create a posttransformation rule.
3. Test your setup.

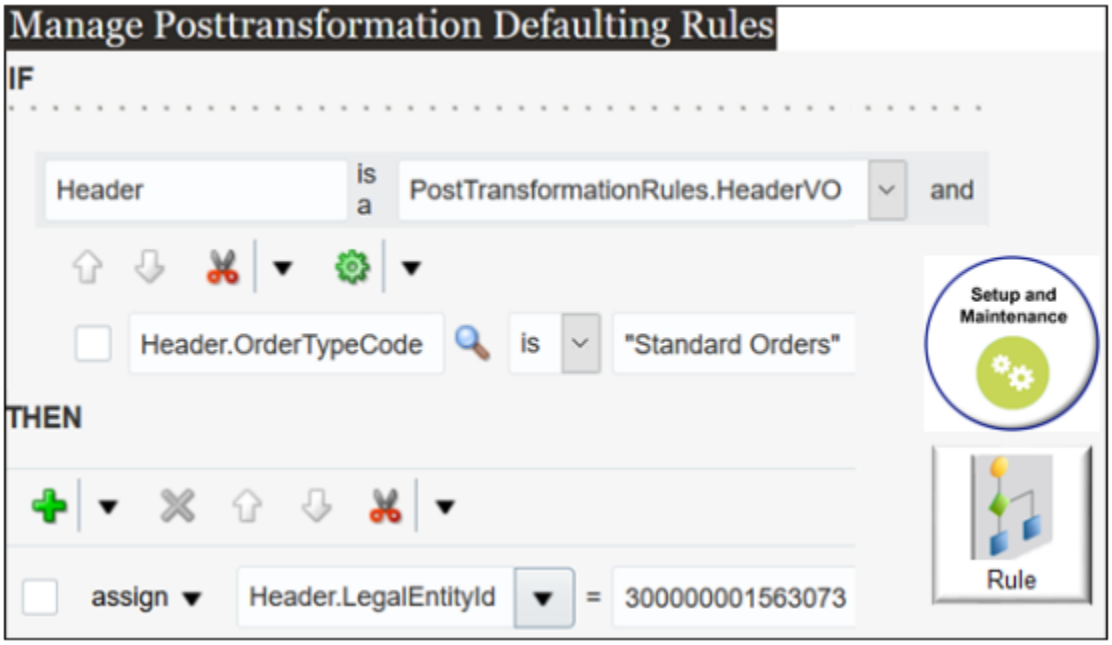
Get Values for Your Attributes

1. Get the order type and the ID for the legal entity.
    - Do an SQL.

```
SELECT distinct LOOKUP_TYPE, LOOKUP_CODE
FROM fusion.FND_LOOKUP_VALUES_TL
WHERE LOOKUP_TYPE = 'ORA_DOO_ORDER_TYPES'
order by 1, 2

SELECT DISTINCT NAME, LEGAL_ENTITY_ID
FROM fusion.HR_LEGAL_ENTITIES
WHERE upper(name) LIKE '&LEGAL_ENTITY_NAME%'
ORDER BY name
```

    For details, see *Use SQL to Query Order Management Data*.

Create a Posttransformation Rule



Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules
2. On the Manage Posttransformation Defaulting Rules page, create a new rule.

3. Add a check mark to the Advanced Mode option.
4. In the If area, set the conditions.

Code	Description
`Header is a PosttransformationRules.Header`	Declare the Header variable into the PosttransformationRules dictionary. Get values for attributes of the order header that the orchestration process is currently processing from the header virtual object (VO), then store them in the Header variable.
`Header.OrderTypeCode is "Standard Orders"`	Proceed to the Then statement only if the order type on the order header is Standard Orders. Make sure you enclose the order type in double quotation marks (" ").

5. In the Then area, add an Assign action.

Code	Description
Assign Header.LegalEntityId = 300000001563073	Order Management uses the LegalEntityId attribute to identify the legal entity.

6. Click **Save > Release**.

   Learn how to create a business rule. For details, see *Overview of Using Business Rules With Order Management*.

Test Your Setup

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Order Type	Standard Orders

2. Notice the value in the legal entity attribute.
3. Click **Submit.**
4. Verify that the legal entity now contains 300000001563073.

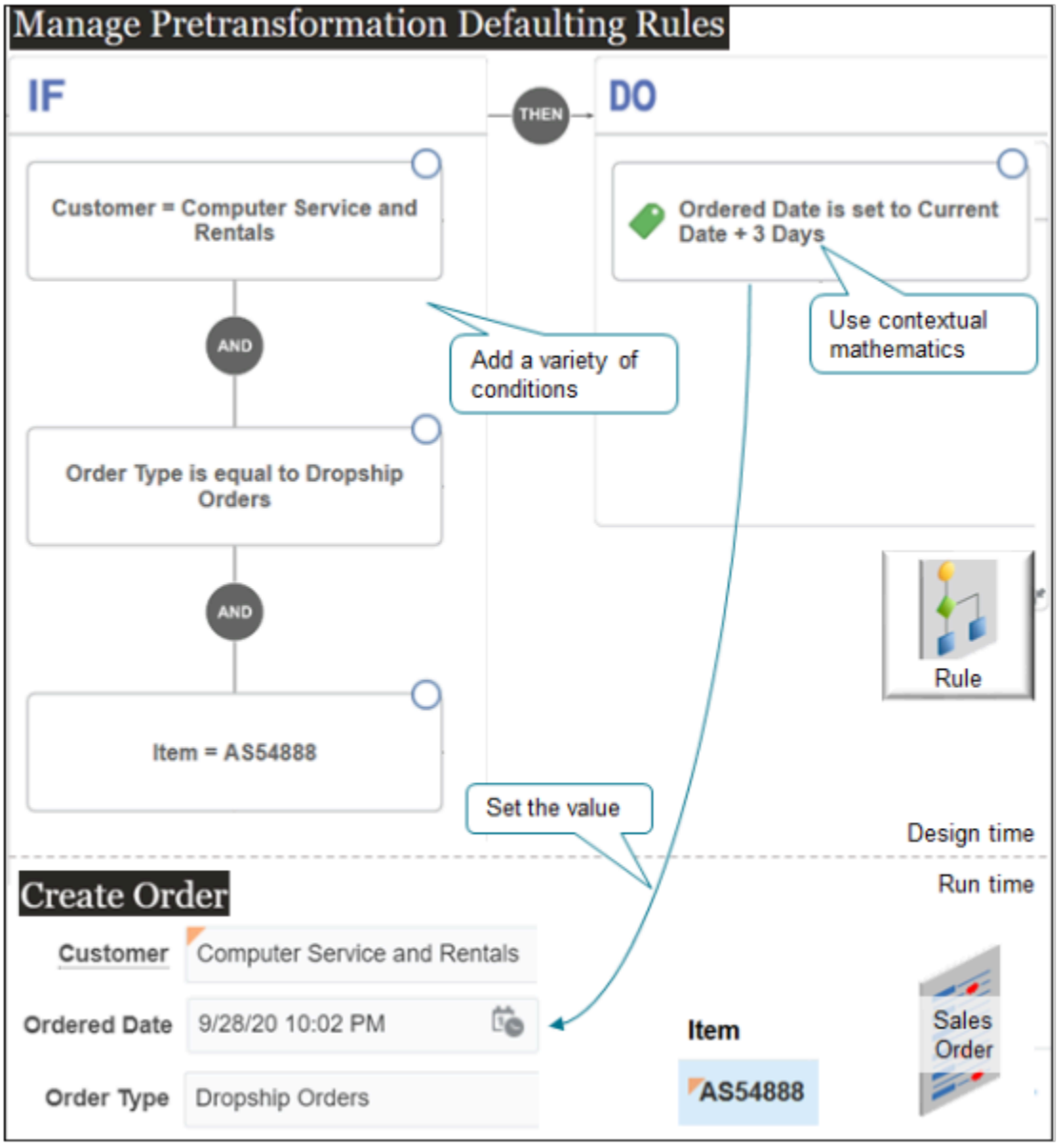
Related Topics

- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules
- Import Shipping Method
- Use SQL to Query Order Management Data

Use Different Attributes in the Condition and Do Simple Math

Create a pretransformation rule on a variety of attributes in the condition or action, and include a simple mathematical equation.

Assume you have an agreement with a drop ship supplier where the supplier needs 3 days lead-time to before they begin processing sales orders that include the AS54888 item for your Computer Service and Rentals customer. To reflect this requirement, you create a pretransformation rule that moves the Ordered Date out by 3 days.



Here's pseudocode for the rule.

```
If the Customer attribute on the order header contains Computer Service and Rentals, and if the Order
 Type attribute on the order header contains Dropship Orders, and if the Item attribute on the order
 line contains AS54888, then set the Ordered Date on the order header to the value that the Current Date
 attribute contains plus 3 days.
```

Summary of the Setup

1. Create the If statement.
2. Create the action.
3. Test your setup.

Create the If Statement

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Pretransformation Rules for Sales Orders
2. Create a new rule.

Attribute	Value
Name	Set the Requested Ship Date for Drop Ship Orders

3. Create the If statement.
    - In the If area, click **New Condition**.
    - In the Create Condition dialog, enter `customer`, wait a moment, then click **Item (Order Fulfill Line)**.

      Customer (Order Header) indicates that you're referencing the Customer attribute on the header of the sales order.
    - Click **Search**.
    - In the Search dialog, search for Computer Service and Rentals, click **Computer Service and Rentals** in the search results, then click **OK**.
    - In the Create Condition dialog, click **OK**.
4. Add the order type to the If statement.
    - In the If area, click **And**.
    - In the Create Condition dialog, enter **order type**, wait a moment, then click **Order Type (Order Header)**.
    - Set the equation to **Is Equal To**.
    - Set the order type to `Dropship Orders`, then click **OK**.
5. Add the item to If statement.
    - In the If area, click **And** that's immediately below the condition you just added.
    - In the Create Condition dialog, enter **item**, wait a moment, then click **Item (Order Fulfill Line)**.
    - Click **Search**.
    - In the Search dialog, search for AS54888, click **AS54888** in the search results, then click **OK**.
    - In the Create Condition dialog, click **OK**.

Create the Action

1. In the DO area, click **New Action > Set a Value**.
2. In the Create Action Dialog, enter `Ordered Date`, then click **Ordered Date (Order Header)**.
3. In the window below Is Set To, click **Attribute**, then enter `Current Date`.
4. Click `Add Arithmetics`. Its the f(x) icon.

   The rules editor adds a row that you can use to do math according to the context of the attribute. For example, you're modify a date attribute that's measured in days, so the editor lets you add or subtract days.
5. In the next line that displays, set the sign to + (plus) and enter the value `3` in the window.
6. Click **OK**.
7. Publish and release your rule.

Test Your Setup

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals
Order Type	`Dropship Orders`

2. Notice that the Ordered Date on the order head defaults to the current date. Assume the current date is 9/01/20 10:02 PM.
3. Add the AS54888 item to an order line, then click **Submit**.
4. Notice that the Ordered Date on the order header is now the current date plus 3 days. In this example, verify that Ordered Date contains 9/04/20 10:02 PM.

Related Topics

- Guidelines for Assigning Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules

Set Values for Attributes That Depend On Each Other

Some attributes depend on each other. For example, if you need to use a shipment set, you must first enable the Shipset attribute.

Assume you have a product line that includes a variety of desktop computers. For example, it includes the CN92777 Desktop Computer, which is a configured item. The CN92777 includes a CPU, monitor, keyboard, and mouse. You want to ship all these components together.

Here are your requirements.

- Use the Standard Order Models order type to indicate that the sales order includes only a configured item.

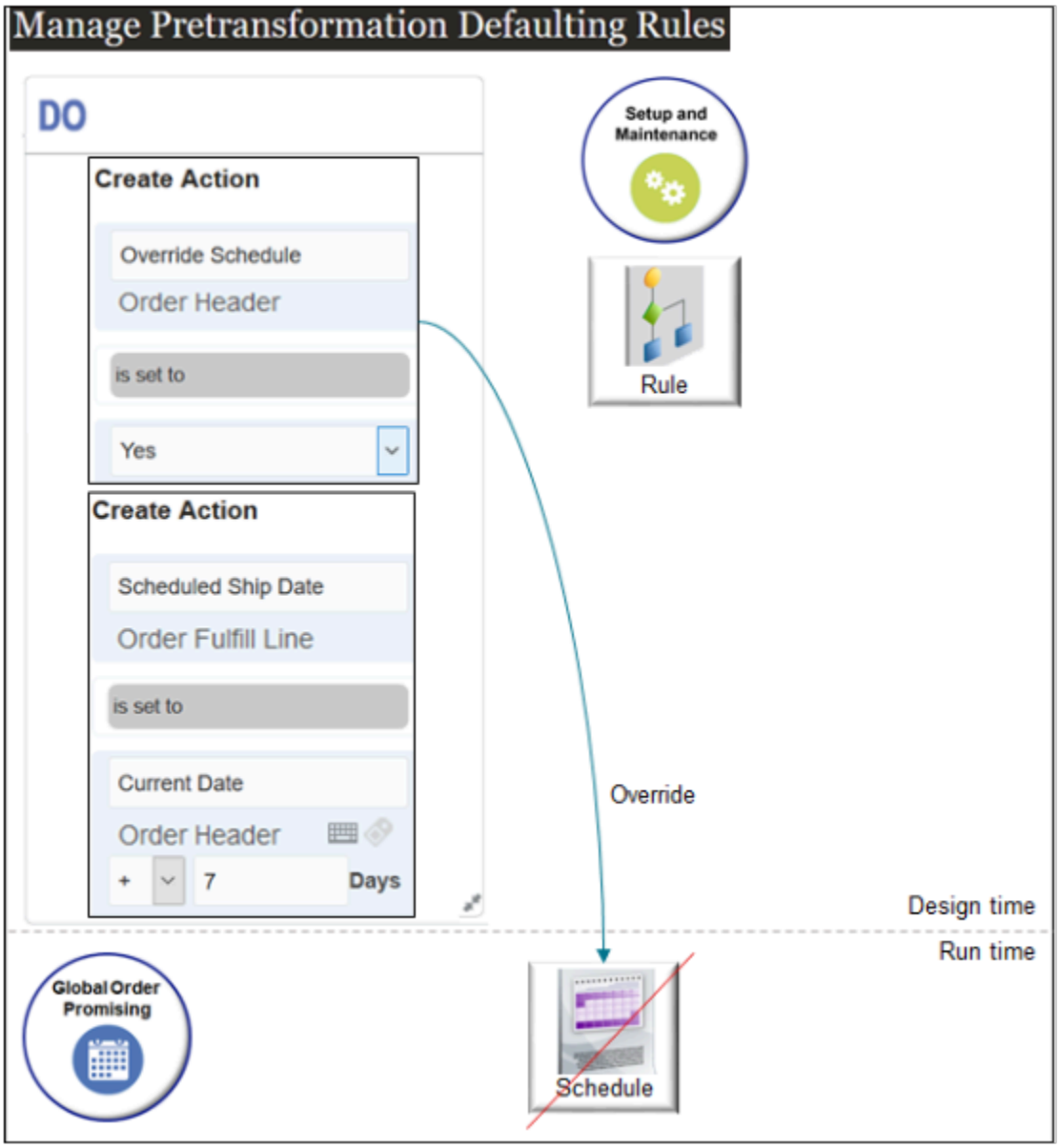
- Ship all order lines together.
- Set the name of the shipment set on each fulfillment line to the order number.

What the Numbers Mean

1. If the order type is Standard Order Models, then do the actions in the rule.
2. You typically set the Ship Lines Together attribute on the order header to Yes as the way to make sure Order Management ships lines together. The name of this attribute in the business rule is `Shipset`, so in your action you set `Shipset` to Yes on the order header.
3. The Source Order attribute in the business rule contains the order number on the order header, so this rule sets the name of the shipment set to the order number. In this example, the order number is 521417.

Override Global Order Promising

Here's another example where one attribute depends on the other. Here, you set the Override Schedule attribute to Yes to prevent Global Order Promising from overriding the Scheduled Ship Date.



Here's the pseudocode for the Do statement.

```
Set the Override Schedule attribute on the order header to Yes, then set the Scheduled Ship Date on the
 fulfillment line to the value of the Current Date on the order header plus 7 days.
```

Related Topics

- Guidelines for Assigning Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules

Use Flexfields to Set Attribute Values

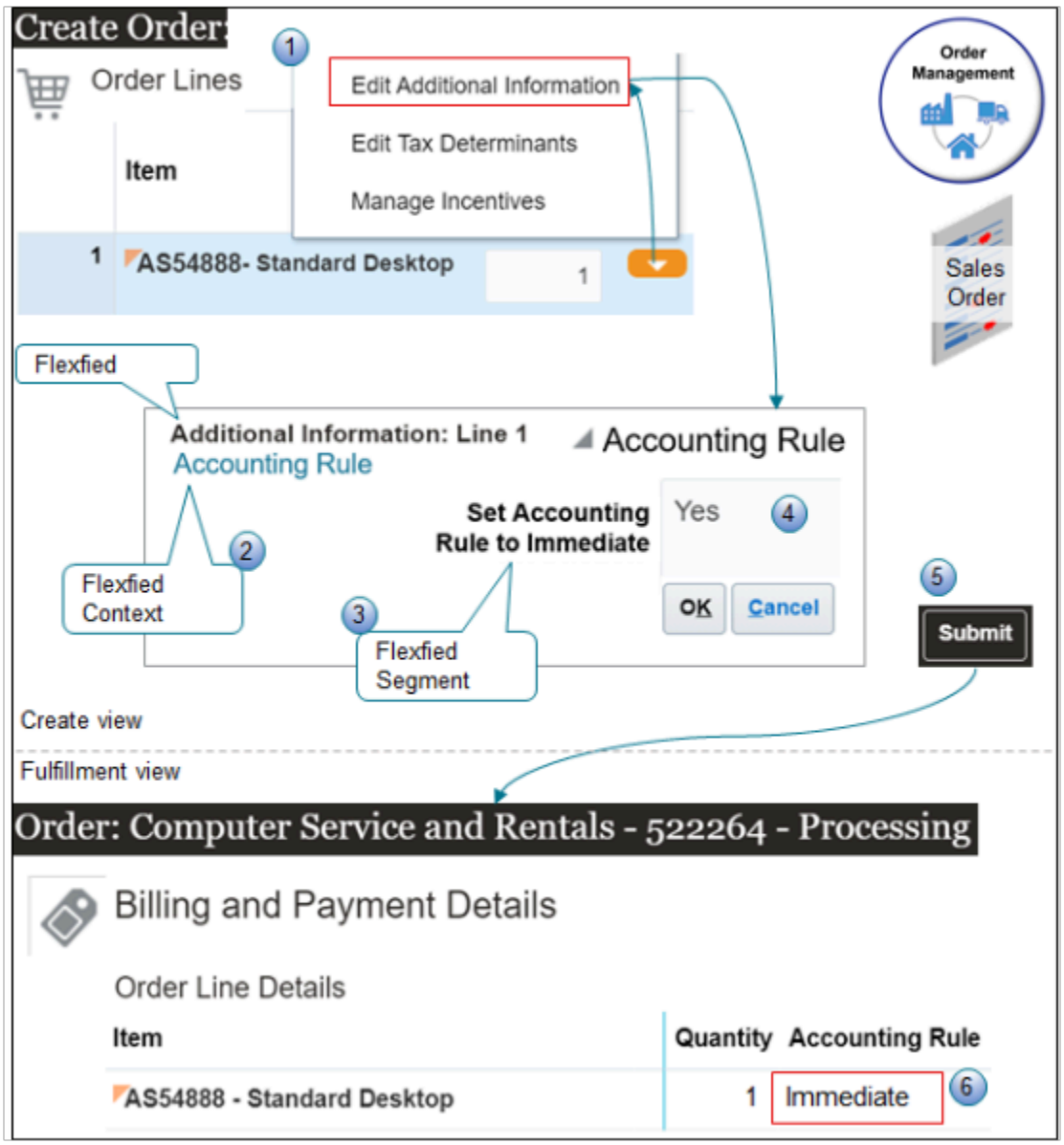
Assume you must allow your Order Entry Specialists to enter a value in a flexfield to indicate they need to change the value of an attribute.

In discussions with your financials team and order entry managers, you determine that you will add a flexfield on the order line with a display name of Set Accounting Rule to Immediate. The Order Entry Specialist will enter the text Yes to indicate whether to change the accounting rule. The rule doesn't affect order processing. Financials uses it for billing purposes.

Here's the pseudocode for your rule.

```
If the Set Accounting Rule to Immediate flexfield contains Yes, then change the value of the Accounting Rule
 attribute to Immediate.
```

Here's how the flow works.



What the Numbers Mean

1. You edit the predefined Fulfillment Line Information extensible flexfield. Order Management displays it when you click **Actions > Edit Additional Information** on the order line.
2. You create a flexfield context with a display name of Account Rule. The Additional Information dialog displays the context.
3. You create a flexfield segment with a display name of Set Accounting Rule to Immediate. The Additional Information dialog displays the segment. The user can enter the text Yes in the segment.
4. You create a business rule that determines whether the segment contains the text Yes.
5. Order Management runs the rule when the user clicks **Submit**.
6. If the segment contains Yes, then the rule sets the Accounting Rule attribute to Immediate.

Summary of the Setup

1. Edit the flexfield.
2. Create your business rule.
3. Test your set up.

1. Edit the Flexfield

You will display your attribute on the fulfillment line, the sales order displays the Fulfillment Line Information extensible flexfield on the fulfillment line, so you edit the Fulfillment Line Information flexfield.

Learn how to do this. For details, see *Overview of Setting Up Extensible Flexfields in Order Management*.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields
2. Open the Fulfillment Line Information flexfield for editing.
3. Click Manage Contexts, then add a new one.

Attribute	Value
Display Name	Accounting Rule
Code	Accounting_Rule
API Name	SetAccountingRuleToImmediate

4. Create a context usage.

Attribute	Value
Name	Additional Fulfillment Line Information

5. Create a context sensitive segment.

Attribute	Value
Display Name	Set Accounting Rule to Immediate
Code	Accounting_Rule_Segment
API Name	setAccountingRuleToImmSegment
Value Data Type	Character
Description	Segment of the extensible flexfield that allows the user to specify whether to set the Accounting Rule attribute to Immediate.
Data Type	Character
Value Set	10 Characters
Prompt	Set Accounting Rule to Immediate
Display Type	Text Box
Instruction Help Text	Enter the text Yes, and the sales order will set the Accounting Rule attribute to Immediate.

6. Add an associated category.

Attribute	Value
Category	Additional Fulfillment Line Information
Code	DOO_FULFILL_LINES_ADD_INFO

7. Add an associated page.

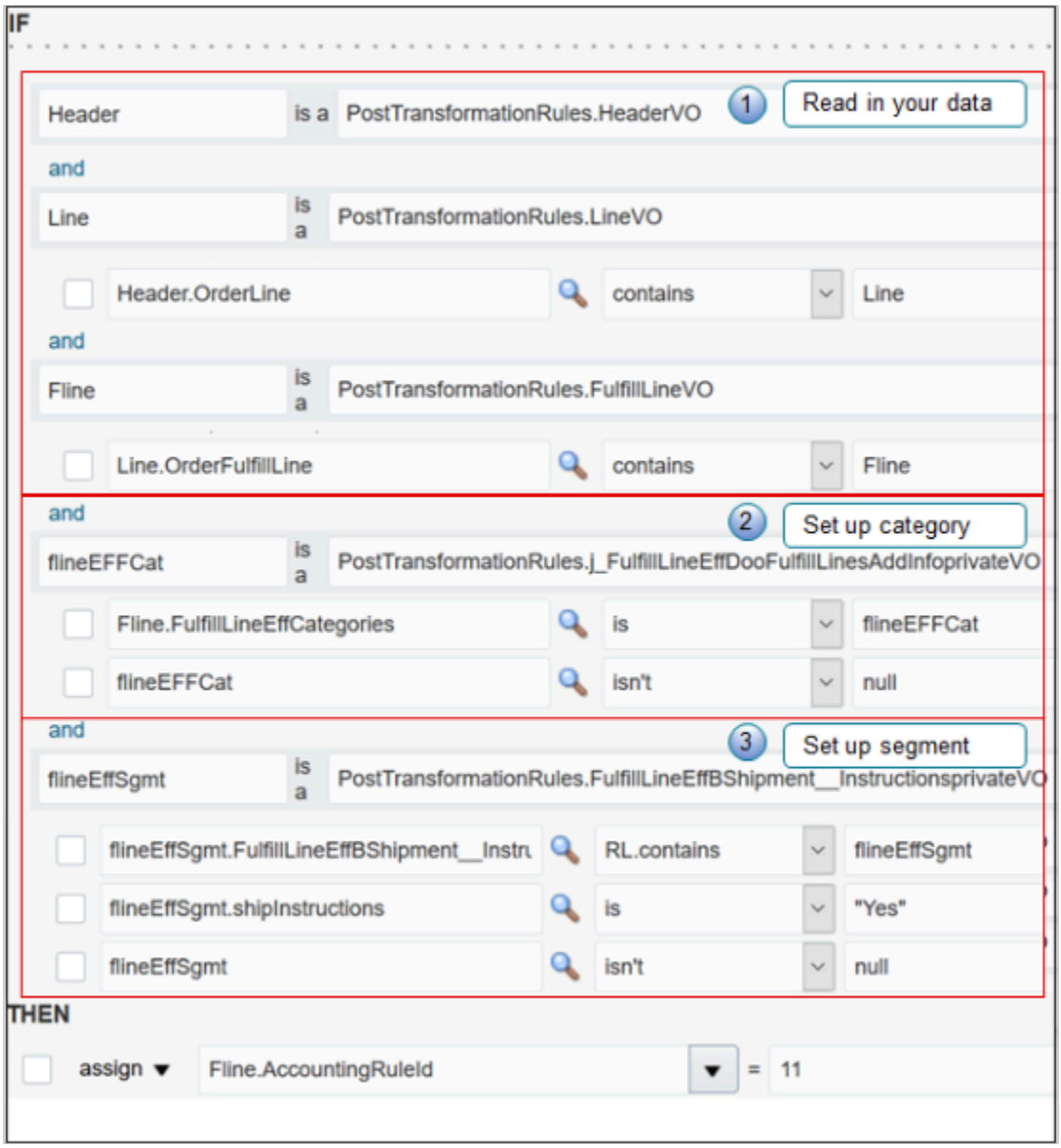
Attribute	Value
Display Name	FulfillLineEFFInfo

Attribute	Value
Code	FulfillLineEFFInfo
Category	Additional Fulfillment Line Information
Usage	Additional Fulfillment Line Information

8. Deploy the Fulfillment Line Information flexfield. For details, see *Set Up Extensible Flexfields in Order Management*.

2. Create Your Business Rule

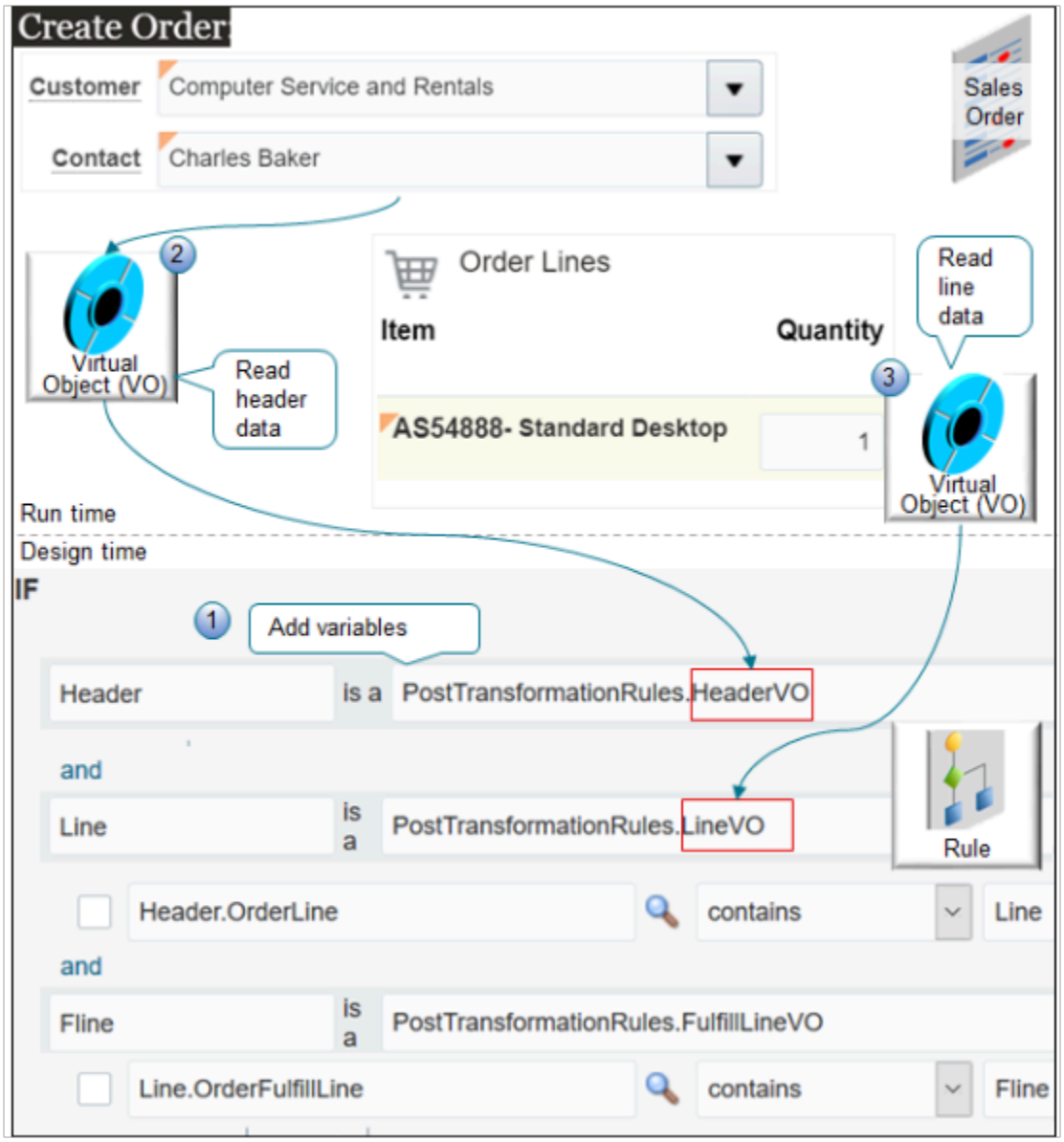
Here's the rule that you will create.



What the Numbers Mean

1. Read your data.
2. Set up the category for the flexfield.
3. Set up the segment for the flexfield.

Read Your Data



What the Numbers Mean

1. Create an If statement in your business rule, then add variables to the rule. The orchestration process uses virtual objects to read data from the sales order into these variables at run time.
2. You use the HeaderVO virtual object (VO) to read header data into the rule, such as values for the Customer attribute and the Contact attribute.
3. You use the LineVO virtual object to read order line data into the rule, such as the values for the Item attribute and the Quantity attribute.

This set up can get complex. Learn about the code and what it means, including the hierarchy and virtual objects. For details, see *Patterns That You Can Use with Extensible Flexfields in Business Rules*.

Examine an example that explains how to create the rule, including stepwise details for establishing a hierarchy and all this other fancy stuff you need to do. For details, see *Use Extensible Flexfields in Transformation Rules*.

Here's an explanation of the code that you use in this example.

Code	Description
`Header is a PostTransformationRules.HeaderVO`	Declare the Header variable into the PosttransformationRules dictionary. Get values for attributes of the order header that the orchestration process is currently processing from the Header virtual object (VO), then store them in the Header variable.
`Line is a PostTransformationRules.LineVO`	Declare the Line variable into the PosttransformationRules dictionary. Get values for attributes of the order line that the orchestration process is currently processing from the Line virtual object (VO), then store them in the Line variable.
`Header.OrderLine contains Line`	Declare the OrderLine variable, then place it under the Header variable. Copy the contents of the Line variable into the OrderLine variable. This code establishes the hierarchy for your flexfield.
`Fline is a PostTransformationRules.FulfillLi`	Declare the Fline variable into the PosttransformationRules dictionary. Get values for attributes of the order line that the orchestration process is currently processing from the FulfillLine virtual object (VO), then store them in the Fline variable.

`Line.OrderFulfillLine contains Fline`	Declare the OrderFulfillLine variable, then place it under the Header Line.
	
	Copy the contents of the Fline variable into the OrderFulfillLine variable.
	
	This code finishes the hierarchy for your flexfield, which is.
	
	`Header`
	`Line`
	`Fline`

`flineEFFCat is a PostTransformationRules.j_FulfillLineEffDooFulfillLinesAddI`	Declare the flineEFFCat variable into the PosttransformationRules dictionary. Get values for attributes of the fulfillment line that the orchestration process is currently processing from the j_FulfillLineEffDooFulfillLinesAddInfoprivateVO virtual object (VO), then store them in the flineEFFCat variable. j_FulfillLineEffDooFulfillLinesAddInfoprivateVO is a predefined object that Order Management uses to process details for an extensible flexfield that sits on an order line.
`Fline.FulfillLineEffCategories is flineEFFCat`	Use the FulfillLineEffCategories property to identify the flineEFFCat variable as a flexfield category. To establish a hierarchy, this code places the flineEFFCat variable under the Fline variable.
`flineEFFCat isn't null`	Avoid a null pointer exception. Make sure the flineEFFCat variable contains a value.
`flineEffSgmt is a PostTransformationRules.FulfillLi Accounting_RuleprivateVO`	Declare the flineEffSgmt variable into the PostTransformationRules dictionary. Get values from the FulfillLineEffB Accounting_RuleprivateVO virtual object, then store them in flineEffSgmt. The business rule gets this virtual object from the order line that the orchestration process is currently processing.

Code	Description
	The business rule uses the pattern FulfillLineEffBxxxprivateVO to store details for your context code, where xxx identifies the context code that you created earlier in this procedure in the Setup and Maintenance work area. Namely, Accounting_Rule.

`flineEFFCat.FulfillLineEffBAccour RuleprivateVO RL.contains flineEffSgmt`	Place the flineEffSgmt variable under the flineEFFCat variable.
	
	This code establishes a hierarchy between the context code and the context segment.
	
	`flineEFFCat`
	`flineEffSgmt`

`flineEffSgmt.Accounting_Rule_ Segment is "Yes"`	Examine the value of the Accounting_Rule_Segment attribute in the flineEffSgmt variable. See if it contains the text Yes. Recall that Accounting_Rule_Segment is the value that you entered for the Code attribute when you created the context segment earlier in this procedure in the Setup and Maintenance work area.
`flineEffSgmt.Accounting_Rule_ Segment isn't null`	Avoid a null pointer exception. Make sure the flineEffSgmt variable contains a value.

Set Up the Category

Code	Description
`flineEFFCat is a PostTransformationRules.j_ FulfillLineEffDooFulfillLinesAddI`	Declare the flineEFFCat variable into the PosttransformationRules dictionary. Get values for attributes on the fulfillment line that the orchestration process is currently processing from the j_FulfillLineEffDooFulfillLinesAddInfoprivateVO virtual object (VO), then store them in the flineEFFCat variable. j_FulfillLineEffDooFulfillLinesAddInfoprivateVO is a predefined object that Order Management uses to process details for an extensible flexfield that sits on an order line.
`Fline.FulfillLineEffCategories is flineEFFCat`	Use the FulfillLineEffCategories property to identify the flineEFFCat variable as a flexfield category. To establish a hierarchy, place the flineEFFCat variable under the Fline variable.
`flineEFFCat isn't null`	Avoid a null pointer exception. Make sure the flineEFFCat variable contains a value.

Set Up the Segment

Code	Description
`flineEffSgmt is a PostTransformationRules.FulfillLi Accounting_RuleprivateVO`	Declare the flineEffSgmt variable into the PostTransformationRules dictionary. Get values from the FulfillLineEffB Accounting_RuleprivateVO virtual object, then store them in flineEffSgmt. The business rule gets this virtual object from the order line that the orchestration process is currently processing.

Code	Description
	The business rule uses the pattern FulfillLineEffBxxxprivateVO to store details for your context code, where xxx identifies the context code that you created earlier in this procedure in the Setup and Maintenance work area. Namely, Accounting_Rule.

`flineEFFCat.FulfillLineEffBAccour RuleprivateVO RL.contains flineEffSgmt`	Place the flineEffSgmt variable under the flineEFFCat variable.
	
	This code establishes a hierarchy between the context code and the context segment.
	
	`flineEFFCat`
	`flineEffSgmt`

`flineEffSgmt.Accounting_Rule_ Segment is "Yes"`	Examine the value of the Accounting_Rule_Segment attribute in the flineEffSgmt variable. See if it contains the text Yes. Recall that Accounting_Rule_Segment is the value that you entered for the Code attribute when you created the context segment earlier in this procedure in the Setup and Maintenance work area.
`flineEffSgmt.Accounting_Rule_ Segment isn't null`	Avoid a null pointer exception. Make sure the flineEffSgmt variable contains a value.

Create the Then statement.

Code	Description
`Assign Fline.AccountingRuleId = 11`	The AccountingRuleId has a set of numeric values. Each value represents an accounting rule. The value 11 represents Immediate.

Make sure you release your rule.

3. Test Your Setup

1. Go to the Order Management work area, create a sales order, and add an order line.
2. On the order line, click the **Actions** down arrow, then click **Edit Additional Information**.
3. Click **Accounting Rule**.
4. In the text box that's labeled Set Accounting Rule to Immediate, enter the text `Yes`.
5. Click **Submit**.
6. On the Order page, click **Billing and Payment Details**.
7. In the Order Line Details area, click **View > Columns > Accounting Rule**.
8. Examine the order line and verify that the Accounting Rule attribute contains Immediate.

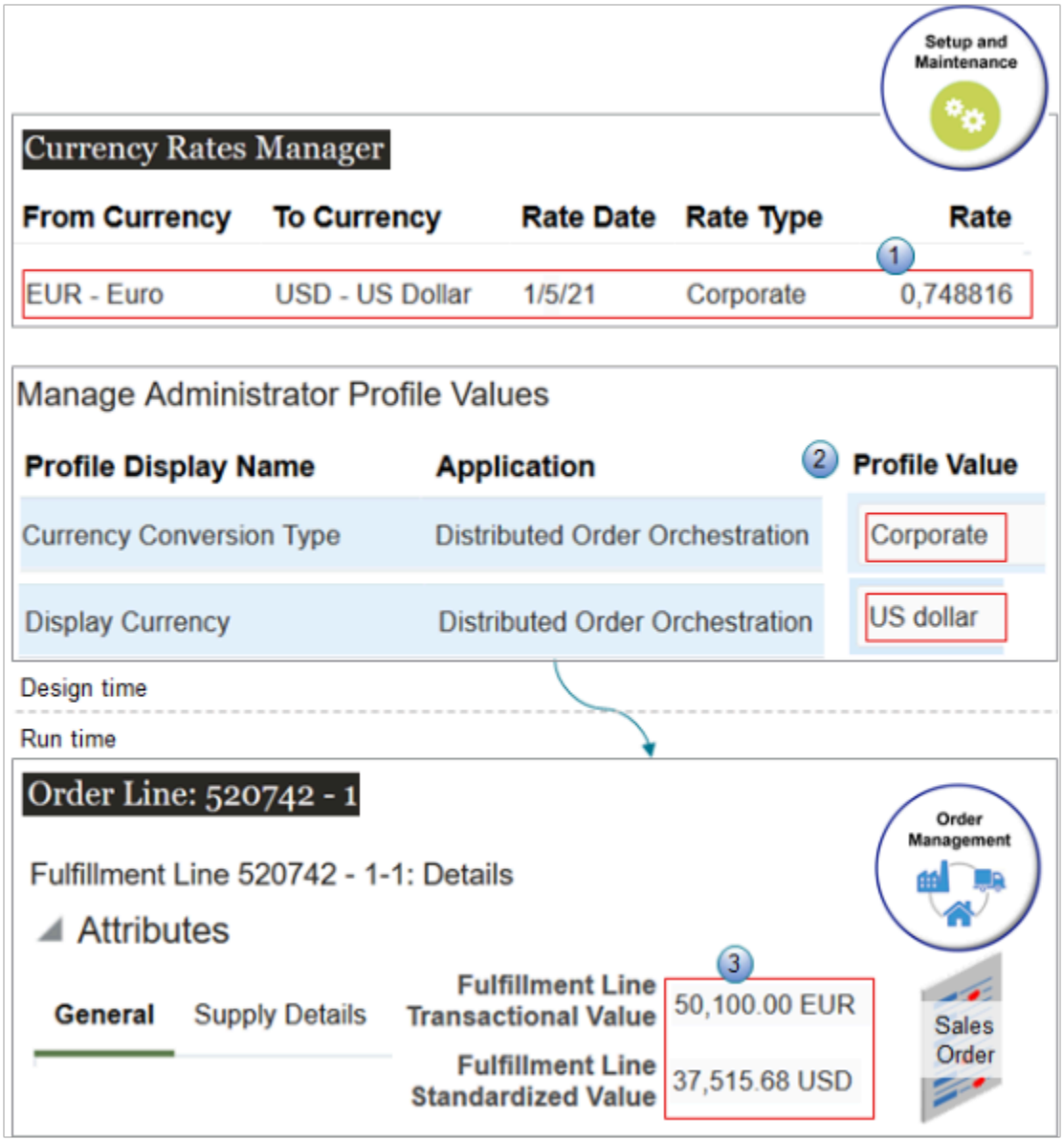
Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Set Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Use Extensible Flexfields in Transformation Rules

Convert Your Currency

Set up Order Management so it converts currency when you sell into markets that use different currencies.

Assume you sell into markets that use the USD currency and the EUR currency.



What the Numbers Mean

1. You use the Currency Rates Manager page to set the conversion rate to use between the EUR and USD currencies on 1/5/21.
2. You use the Manage Administrator Profile Values page to set the Currency Conversion Type, such as Corporate, and the Display Currency to display on the sales order, such as US Dollar.
3. At run time, Order Management uses your conversion rate to calculate the transactional value in EUR and the standardized value in USD of the fulfillment line.

Summary of the Set Up

1. Modify the predefined conversions.
2. Examine the conversion type.
3. Examine the display currency.
4. See how currency shows up on sales orders.

Modify the Predefined Conversions

1. Make sure you have the privileges that you need to manage conversion rates. For example, the predefined Inventory Manager job role has these privileges, but you must set up your own job role and add privileges to them.
   For details, see *Privileges That You Need to Implement Order Management*.
2. In the Setup and Maintenance work area, go to the task.
   - Offering: Manufacturing and Supply Chain Materials Management
   - Functional Area: Financial Reporting Structures
   - Manage Conversion Rate Types
3. On the Currency Rates Manager page, click **Daily Rates**, then search for the values.

Attribute	Value
From Currency	EUR
To Currency	USD
Rate Date	1/1/20 to 1/6/21
Rate Type	Corporate

   Oracle Applications come with a number of predefined conversion rates. Search for the one you need. Create a new one only if you can't find a predefined rate that meets your needs.
4. Examine the search results.

From Currency	To Currency	Rate Date	Rate Type	Rate
EUR	USD	1/6/21	Corporate	0.815302
EUR	USD	1/5/21	Corporate	0.748816
EUR	USD	1/4/21	Corporate	0.748816
EUR	USD	1/3/21	Corporate	0.748816

From Currency	To Currency	Rate Date	Rate Type	Rate
EUR	USD	1/2/21	Corporate	0.748816
EUR	USD	1/1/21	Corporate	0.748816

This example illustrates how Oracle Applications automatically adjusts the conversion to match the daily rate, which might change each day in reply to market conditions. Notice that the range changed from 0.748816 to 0.815302 on 1/6/21.
You can also change the daily rate to meet your needs.
For details, see *Manage Currency Conversion Lists*.

5. Collect data.
If your deployment depends on the daily rate, and if you don't use a currency conversion list, then you must collect the Currencies entity at least one time each day so Order Management can use the current rate. Set up collections so it collects the Currencies entity automatically each day. For details, see *Collect Planning Data for Order Management*.

Examine the Conversion Type

Examine the predefined conversion type that Order Management uses for sales orders.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Sales
    - Functional Area: Sales Foundation
    - Task: Manage Administrator Profile Values
2. On the Manage Administrator Profile Values page, search for the value.

Attribute	Value
Profile Display Name	Currency Conversion Type
Application	Distributed Order Orchestration This value is an earlier name for Oracle Order Management. Oracle Applications still use it for Order Management.

3. In the Profile Values area, notice the value.

Attribute	Value
Profile Value	Corporate

Attribute	Value
	Notice that Corporate is the same value that you examined in the Rate Type attribute on the Currency Rates Manager page.

Examine the Display Currency

Examine the predefined display currency that Order Management uses for sales orders.

1. Search for the value.

Attribute	Value
Profile Display Name	Display Currency

2. In the Profile Values area, notice the value.

Attribute	Value
Profile Value	US Dollar Notice that US Dollar is the same value that you examined in the To Currency attribute on the Currency Rates Manager page. If you don't set a value, then Order Management uses US Dollar, by default.

See How Currency Shows Up On Sales Orders

1. Go to the Order Management work area, create a sales order, and set the values.

Attribute	Value
Business Unit	Vision Germany
Item	AS54888
Quantity	10
Price	5010.00

Attribute	Value

Assume the Ordered Date is 1/5/21.

2. Click **Submit**, then click **Actions > Switch to Fulfillment View**.
3. On the Order page, on the Order Lines tab, click the value in the Order Line column.
4. On the Order Line page, in the Attributes area, on the General tab, notice the values.

Attribute	Value
Fulfillment Line Transactional Value	50,100.00 EUR 50,100.00 equals the Quantity of 10 multiplied by the Selling Price of 5010.00 each.
Fulfillment Line Standardized Value	37,515.68 USD 37,515.68 equals the Quantity of 10, multiplied by the Selling Price of 5010.00 each, multiplied by the currency conversion rate of 0.748816.

Related Topics

- Privileges That You Need to Implement Order Management
- Collect Planning Data for Order Management
- Manage Currency Conversion Lists

Item Values

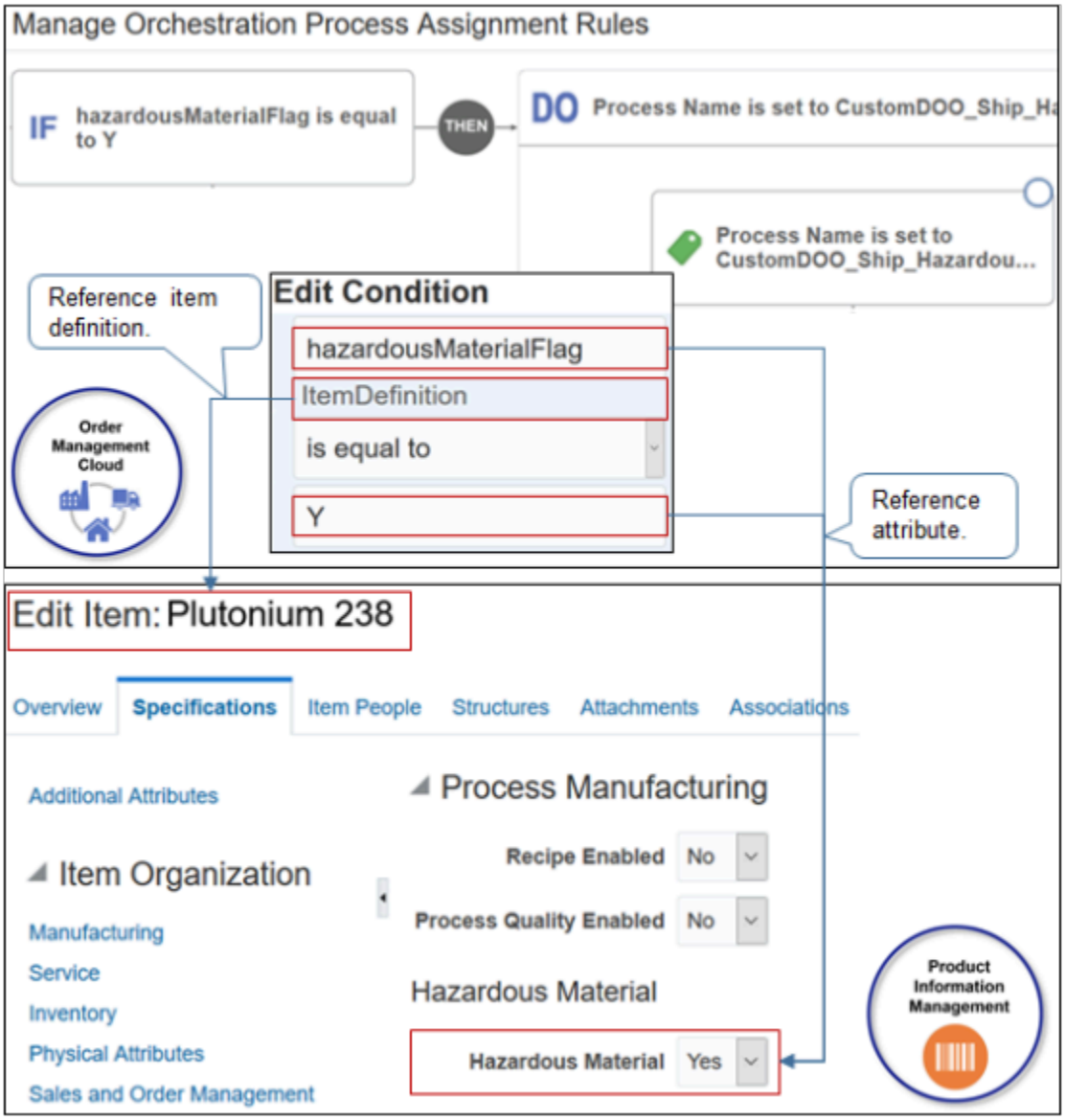
Get Data from Product Information Management

Create a business rule to get data from the Product Information Management work area and bring it into Order Management.

In this example, you assign an orchestration process that includes extra steps so order fulfillment safely handles hazardous material, such as packing in a controlled container and shipping through a company that specializes in shipping hazardous material. The fulfillment line doesn't come predefined with a hazardous attribute, so you reference an attribute in Product Information Management instead. Here's the assignment rule you will create.

- If shipping a hazardous material, then use the Ship_Hazardous_Material orchestration process.

You create a rule that references the hazardousMaterialFlag attribute in Product Information Management.



Learn how to create a business rule. For details, see *Overview of Using Business Rules With Order Management*.

Do it.

1. Create the rule.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Process Assignment Rules for Sales Orders
    - On the Manage Orchestration Process Assignment Rules page, click **Create New Rule**, then set the values.

Attribute	Value
Name	Assign According to Hazardous Material
Description	Assign orchestration process depending on hazardous material.

2. Create the If statement.
    - Click **New Condition**.
    - In the Create Condition dialog, enter `haz`, wait a moment, then click **harzardousMaterialFlag (Item Definition)**.

      The phrase Item Definition indicates that the attribute resides on an item definition in Product Information Management.
    - Set the operator to `Is Equal To`.
    - Enter `Y`, then click **OK**.
3. Create the Do statement.
    - Click **Then > Do > New Action**.
    - In the Create Action dialog, enter `process`, wait a moment, then click **Process Name (Order Fulfill Line)**.

      The phrase Order Fulfill Line indicates that the orchestration process you set will process order fulfillment lines.
4. Search for your orchestration process.
    - Click **Search > Advanced**.
    - Set Process Name to Contains.
    - Enter `Ship_Hazardous_Material`.

      The search is case sensitive.
    - Click **Search**.

      For this example, assume you already created and deployed this orchestration process. If you didn't deploy, then Search won't find it.
    - Click the **row** in the search results.
    - Click **OK**.

- In the Create Action dialog, click **OK**.

5. Activate and publish your rule.

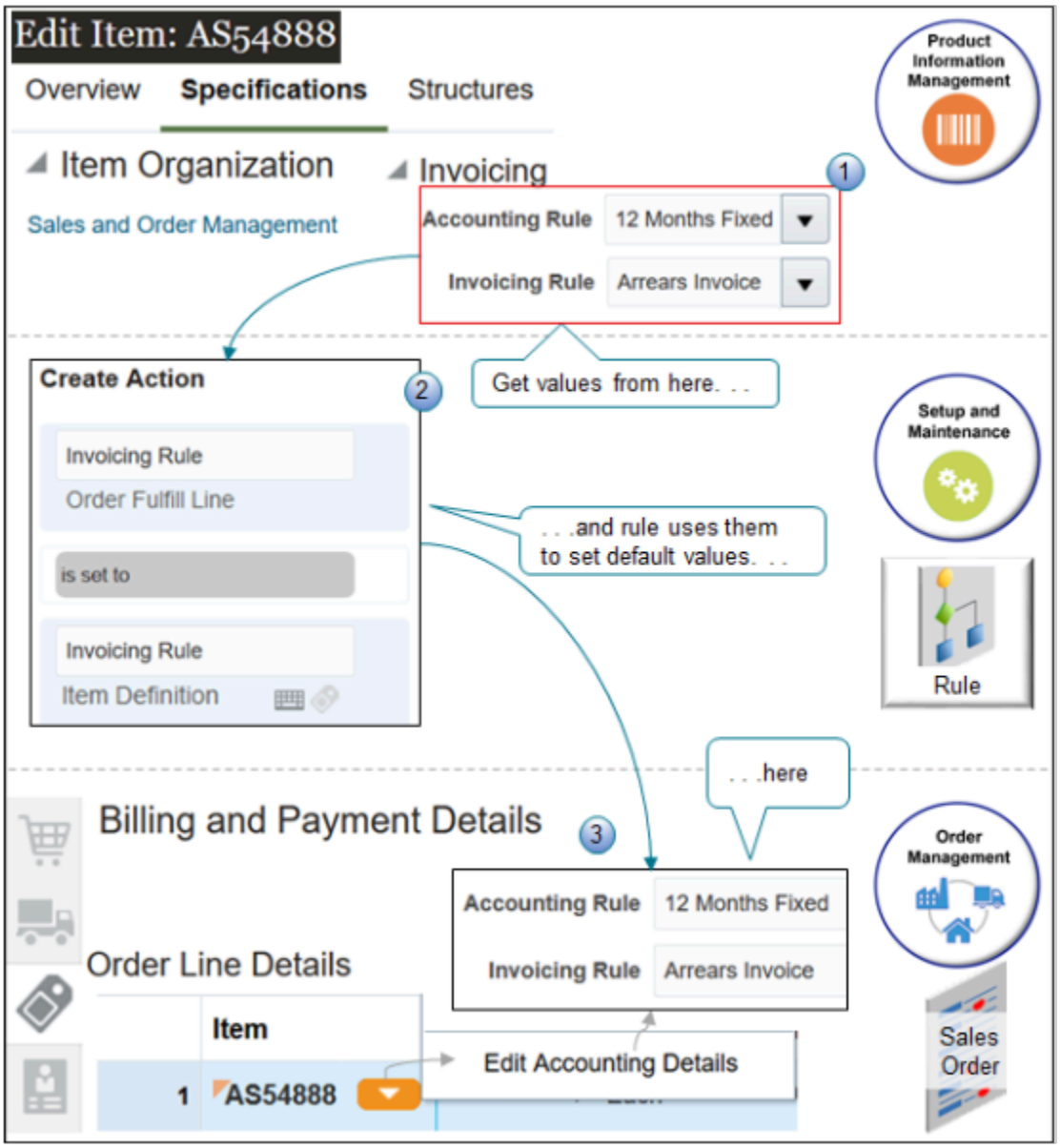
Related Topics

- Guidelines for Assigning Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Overview of Using Business Rules With Order Management

Get Values from Product Information Management and Use Them to Set Values on Sales Orders

Create a business rule that gets a value for your item from the Product Information Management work area, then use it to set the default value for an attribute on the sales order.

Here's an example of how it works.



What the Numbers Mean

1. Use the Product Information Management work area to set the values for the attributes you need to reference.
2. Use the Setup and Maintenance work area to create a business rule that gets the values from Product Information Management and uses them to set default values for attributes on the sales order in the Order Management work area.
3. At run time, use the Order Management work area to create the sales order and Order Management automatically sets the attribute values.

Here's the example pretransformation rule you will create.

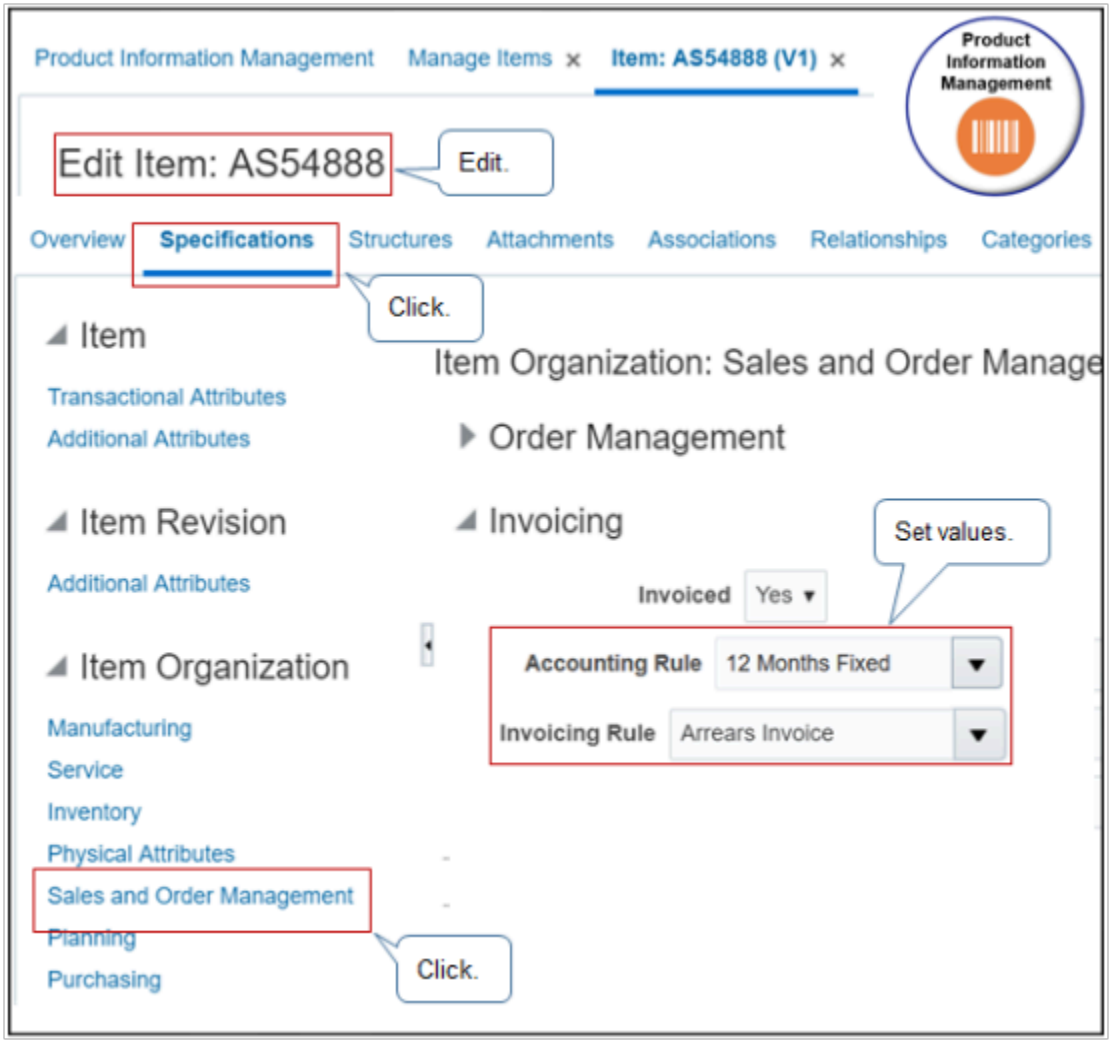
```
If the Order Entry Specialist adds the AS54888 item to an order line, then set the default value for the Accounting Rule attribute on the order line to the same value that the Accounting Rule attribute contains for the item in Product Information Management, and set the value for the Invoicing Rule on the order line to the same value that the Invoicing Rule attribute contains for the item in Product Information Management.
```

This topic uses example values. You might need different values, depending on your business requirements.

Summary of the Steps

1. Modify the item.
2. Create the pretransformation rule.
3. Test your setup.

Modify the Item



Do it.

1. Make sure you have the privileges that you need to administer Product Information Management.

   If you don't sign in with these privileges, then the Product Information Management work area won't display your product details and you can't do this procedure.
2. Go to the Product Information Management work area.

3. Click **Tasks > Manage Items**, then search for the value.

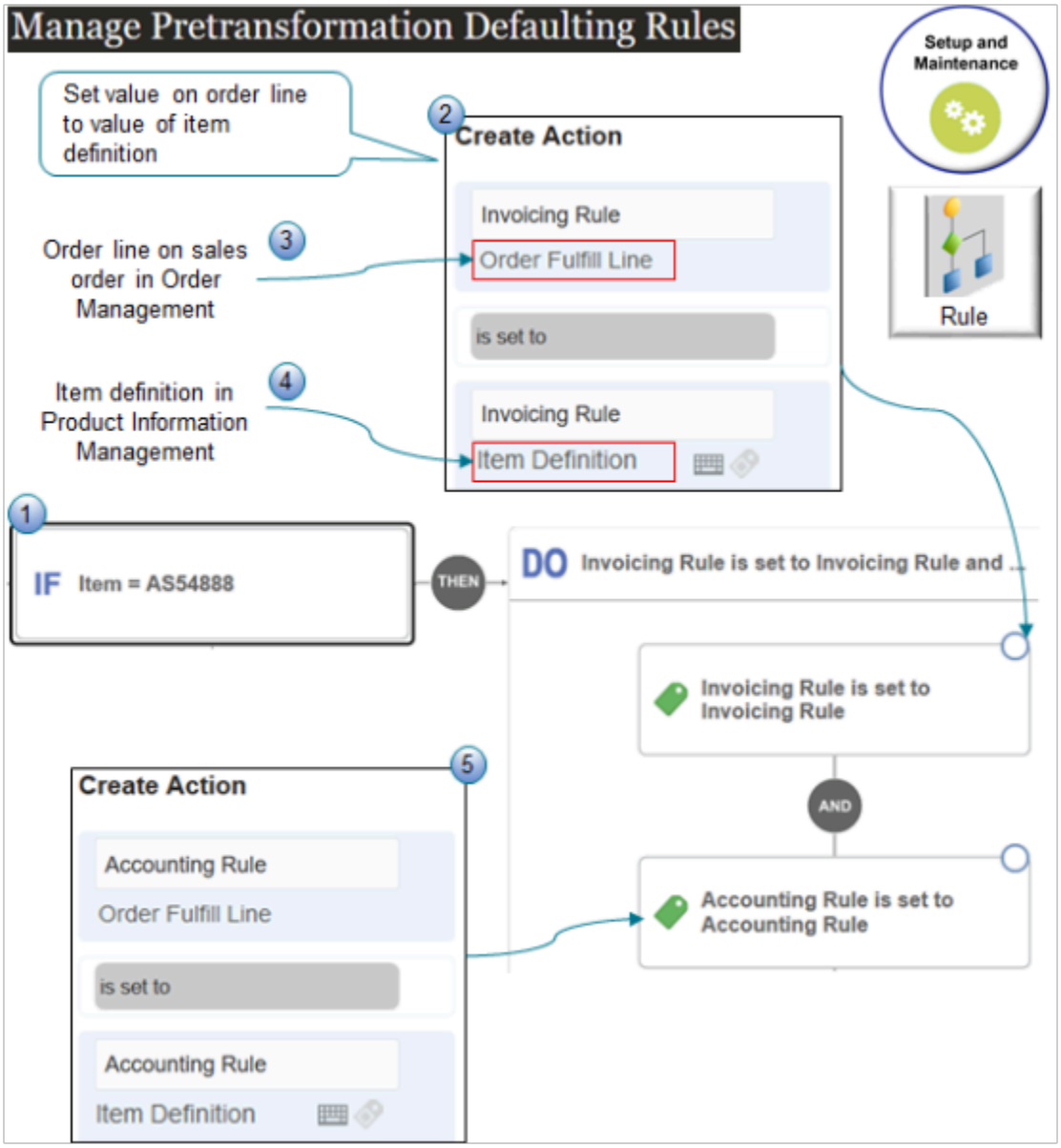
Attribute	Value
Item	AS54888

4. In the search results, click **AS54888**.
5. On the Edit Item page, click **Specification > Sales and Order Management**.
6. In the Invoicing area, set the values, then click **Save**

Attribute	Value
Accounting Rule	12 Months Fixed
Invoicing Rule	Arrears Invoice

7. Collect data. For details, see *Collect Planning Data for Order Management*.

Create the Pretransformation Rule



What the Numbers Mean

1. Create the condition that asks whether the item on the order line is the AS54888.
2. Create an action for the Invoicing Rule.
3. Set the Invoicing Rule attribute on the order line on the sales order in Order Management to:
4. The value that the Invoicing Rule attribute contains on the item definition in Product Information Management.
5. Create an action for the Accounting Rule.

Do it.

1. Sign out, then sign into Oracle Applications with the privileges that you need to administer Order Management.
2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders

- Task: Manage Pretransformation Rules for Sales Orders

3. Create a new rule.

Attribute	Value
Name	Set Default Values for Accounting and Invoicing Rules

4. Create the If statement.
   - In the If area, click **New Condition**.
   - In the Create Condition dialog, enter `item`, wait a moment, then click **Item (Order Fulfill Line)**.

     Item (Order Fulfill Line) indicates that you're referencing the Item attribute on the fulfillment line of the sales order.
   - Click **Search**.
   - In the Search dialog, search for AS54888, click **AS54888** in the search results, then click **OK**.
   - In the Create Condition dialog, click **OK**.

```
IF item = AS54888
```

5. Create the Then statement.
   - Click **Then > Do**.
   - In the DO area, click **New Action > Set a Value**.
   - In the Create Action Dialog, enter `Accounting Rule`, then click **Accounting Rule (Order Fulfill Line)**.
   - Click Attribute.
   - In the window that contains the text Attribute, enter Accounting Rule, then click **Accounting Rule (Item Definition)**.

     Accounting Rule (Item Definition) indicates that you're referencing the Accounting Rule attribute of the item in the Product Information Management work area.

     Here's some pseudocode for the statement you just created.

     ```
     Set the Accounting Rule attribute on the fulfillment line of the sales order to the value that the Accounting Rule attribute contains for the item in the Product Information Management work area
     ```
   - Click **OK**.

```
Invoicing Rule is set to Arrears Invoice
```

6. Add another action to the Then statement.
    - In the DO area, click **And > Set a Value**.
    - In the Create Action Dialog, enter `Invoicing Rule`, then click **Invoicing Rule (Order Fulfill Line)**.
    - Click Attribute.
    - In the window that contains the text Attribute, enter Invoicing Rule, then click **Invoicing Rule (Item Definition)**.

      Invoicing Rule (Item Definition) indicates that you're referencing the Invoicing Rule attribute of the item in the Product Information Management work area.

      Here's some pseudocode for the statement you just created.

      ```
      Set the Invoicing Rule attribute on the fulfillment line of the sales order to the value that the Invoicing Rule attribute contains for the item in the Product Information Management work area
      ```
    - Click **OK**.
7. Activate and publish your rule. For details, see *Manage Pretransformation Rules*.

Test Your Setup

1. Go to the Order Management work area and create a sales order.
2. Add the AS54888 to an order line.
3. Click **Billing and Payment Details**.
4. On the order line, in the Item column, click the **down arrow**, then click **Edit Accounting Details**.
5. In the Edit Accounting Details dialog, verify the values.

Attribute	Value
Accounting Rule	12 Months Fixed
Invoicing Rule	Arrears Invoice

6. Go to the Product Information Management work area and change the value of the Accounting Rule to Immediate.
7. Collect data.
8. Create another sales order and verify that Accounting Rule defaults to Immediate.

Related Topics

- Guidelines for Assigning Orchestration Processes
- Roadmap for Setting Up Order-to-Cash
- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules

Constrain Actions

Processing Constraints

Create a processing constraint to control the modifications that you can make on a sales order.

A processing constraint is a rule that controls who can change a sales order, what can change in the sales order, and when the change can occur. For example:

- If an Order Entry Specialist attempts to submit or modify a sales order, order line, or fulfillment line, and if a processing constraint doesn't allow the submit or change, then Order Management rejects it and displays a message.
- If a source system attempts to submit or modify a sales order, then Order Management rejects it and sends a return message.
- Order Management also uses predefined processing constraints to make sure each fulfillment request includes the attributes that it needs to process the request.

Examples

Example Constraint	Description
Reject a change when the sales order is shipping.	Assume an orchestration process achieves the shipping stage for a sales order, and a user submits a change. • The shipping stage occurs late in the orchestration process. • Its expensive and not practical to change the sales order. Create a constraint that rejects the change after the orchestration process has achieved the shipment step.
Reject each sales order that doesn't include required attributes.	Assume your company doesn't deliver items to an address that doesn't include a ship-to contact. Create a constraint that rejects the sale order when it doesn't include a ship-to contact.
Reject a change that requires approval.	Assume your company doesn't allow the Order Entry Specialist to submit a change if the transaction value exceeds $10,000, and if a manager hasn't approved the change. Create a processing constraint that rejects a change when all of these conditions are true. • The user signed in with the Order Entry Specialist role. • The transaction exceeds $10,000. • A manager hasn't approved the change.

Parts of a Processing Constraint

Part	Description
Role	The job role that the constraint references doesn't allow you to make the change. For example, you can constrain an Order Entry Specialist from changing a sales order when the orchestration process proceeds beyond a step that you specify.
Action	The action that the constraint doesn't allow. You can constrain the user from doing these actions. • Create • Validate • Update • Split • Submit • Cancel • Delete
Condition	The condition that the constraint evaluates to determine whether to apply the constraint. For example, you can create a condition that applies a constraint when Order Management submits a sales order.

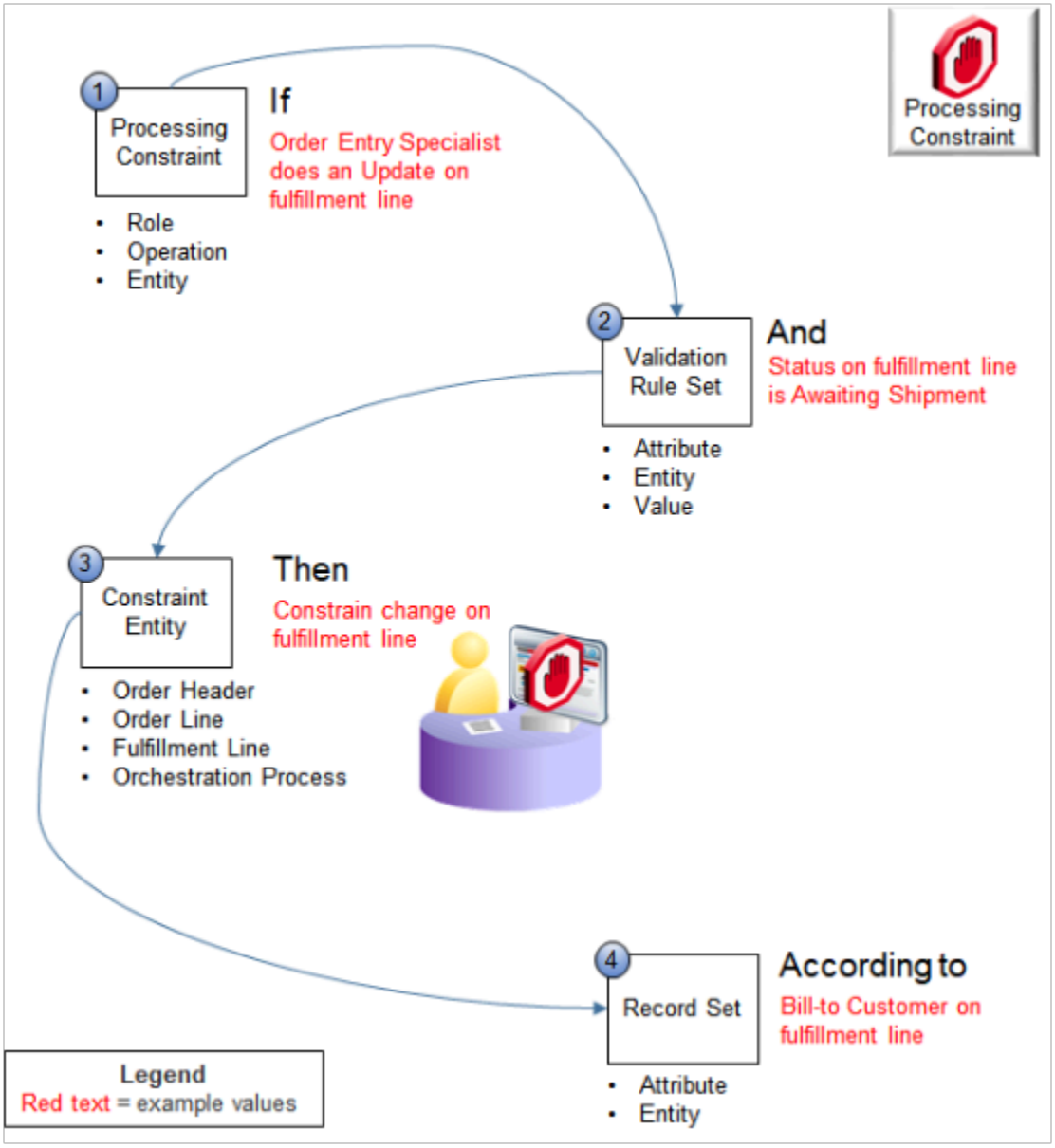
This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

How It Works

A processing constraint applies this logic.

- If the validation rule set is true, and if the user attempts an action on the record set that the processing constraint prevents, then constrain the operation and display a message.

Consider an example.



Here is the logic for this example.

Step	Object	Description
1	Processing constraint	If the Order Entry Specialist role does an Update operation on the fulfillment line entity:
2	Validation rule set	And if the Status attribute on the fulfillment line entity contains a value of Awaiting Shipment:
3	Constraint entity	Then constrain the change that the Order Entry Specialist is attempting to make:

Step	Object	Description
4	Record set	According to the Bill-to Customer attribute on the fulfillment line entity.

Another Example

Consider the predefined Shipping Method on Order Header Isn't Valid processing constraint.

If This Applicable Role	Attempts to Do This Constrained Operation	On This Constraint Entity	And If This Validation Rule Set is True	Reject the Change and Display This Message
All	Submit	Order Header	Shipping Method on Order Header Isn't Valid	You can't submit the sales order because the combination of the mode of transport, service level, and carrier that determines the shipping method on the order header isn't valid.

The Shipping Method on Order Header Isn't Valid rule set looks at the mode of transport, service level, and carrier on the order header. If one of these attributes contains a value, then all of them must contain a value. If one of them contains a value but not all of them do, then the rule set evaluates to true and the processing constraint prevents the submit.

Objects That You Set Up for a Processing Constraint

Object	Description
Validation rule set	A group of one or more If statements. For example: • If the order is closed, then reject the change. In this example, a predefined validation rule set named Order is Closed examines the Open attribute on the order header to determine whether the value is N. Note • Use the validation rule set to restrict the validation that the constraint does to lines that meet the condition that you specify, such as lines that are billed. • The validation rule set prevents the constraint from examining all lines, which might degrade performance. • Create the validation rule set before you create the constraint. • You can't modify or delete a predefined validation rule set, but you can create a new one. • You can apply a processing constraint when the condition is true or isn't true. • You can't enter a value that contains a ~ (tilde) when you create a validation rule set because Order Management uses the tilde as a delimiter when it evaluates the constraint rule at runtime. You might also encounter this problem at runtime with some order attributes. For example, a shipping instruction might contain a tilde.

Object	Description
Record set	A set of records that Order Management groups according to common attribute values so it can evaluate a constraint. For example, to evaluate all sales orders for a customer, specify to evaluate one of these entities when you create the record set. • Order header • order line • Order fulfillment line You can then select an attribute to refine the record set. For example, to evaluate all sales orders for a customer, select the Order Header entity, then select the Sold-to Customer attribute. A validation rule set and a record set work together to create the conditions where the constraint constrains the action. You must create your validation rule set and record set first, then create the constraint.
Constraint entity	The business object or orchestration process that the constraint constrains. For example, an order header, or an attribute of an order header, such Latest Acceptable Ship Date.
Constraint package	A set of triggers that Order Management applies to a table in an Oracle database. A background process sets triggers when you create a constraint package. A constraint package can activate a new or modified. • Validation rule set that's of a table type • Record set for a processing constraint Here are the ways that you create a constraint package. • Use the Manage Processing Constraints page or run the Generate Constraint Packages scheduled process. • Create or modify a record set or a validation rule set that's of a table type. You don't need to create a constraint package for a validation that isn't a table type.

Constraint Name

Here are the text strings that you can use in the constraint name.

Text String	Constraint That Occurs During Validation With
GTM	Oracle Global Trade Management.
PAYMENT	Payment.
PRICING	Pricing.
Constraint name doesn't include GTM, PAYMENT, or PRICING	Screen for all types during order validation.

Text String	Constraint That Occurs During Validation With

For example, to indicate payment, use DOO_PAYMENT_EXCEPTION.

The text style in the text string isn't case-sensitive. For example, use `GTM`, `Gtm`, `gtm`, `gtM`, and so on.

You can place the text string anywhere in the constraint name.

How Order Management Displays Your Constraint's Message

Order Management typically displays the value that you enter in your constraint's Message attribute in the Order Management work area. However, it doesn't display that message for some actions in a fulfillment view.

Assume you create a constraint that prevents you from editing each fulfillment line that's in the shipping stage, and you add this text in the constraint's Message attribute.

```
You can't update the fulfillment line because its in the shipping stage.
```

At runtime, you:

1. Create a sales order with 100 order lines and click **Submit**.
2. Wait several days. While you wait, Order Management ships 70 of the 100 order lines.
3. Open the order and click **Switch to Fulfillment View**.

On the Order page, you click **Fulfillment Lines**, select all 100 lines, then click **Actions > Edit**.

Order Management displays a warning that you can't do the action on all the lines, but you expected it to display your constraint's message. Order Management doesn't display the value from your constraint's Message attribute because it would display 70 separate instances of that message. Instead, you can click OK in the warning dialog, then Order Management will display a dialog that allows you to edit the 30 lines that aren't constrained.

More

Note

- If you set up a constraint that doesn't include a condition, then the constraint is always true. For example, the predefined processing constraint that prevents the Order Entry Specialist from deleting a sales order prevents deletion in all situations.
- You can use a constraint with an extensible flexfield.
- You can write a constraint for a sales credit. For example, write a constraint that prevents the user from updating the sales credit if Order Management already shipped the order line. You can't write a validation rule set that constrains a sales credit. For example, you can't write a constraint that prevents the user from updating the warehouse if the sales credit empty.
- Learn how to use a constraint to manage change, including how to set up AND and OR conditions. For details, see *Guidelines for Managing Change That Occurs During Order Fulfillment*.

Related Topics

- Manage Processing Constraints
- Constrain Changes That Users Make in Extensible Flexfields
- Constraint Entities
- Guidelines for Managing Change That Occurs During Order Fulfillment

Constraint Entities

A constraint entity is the view or orchestration process that a processing constraint constrains.

Type of Constraint Entity	Description
View	Constrain change according to the value of an attribute that a user can view in the Order Management work area. Here are the types of view constraint entities you can select: • Order Fulfillment Line • Order Line • Order Header For example, if you set Order Header as the constraint entity, then the constraint constrains changes according to the order header. Use a predefined view constraint entity, but you can't create a new one. The Attribute Details list on the Manage Constraint Entities page displays when you select a view constraint entity. Use it to select the attributes to constrain.
Process	Constrain an action from occurring at some point in an orchestration process, such as updating an attribute or deleting a table entity, according to a combination of orchestration process, task, and service. For example, the Update Shipping fulfillment task references the OrderOnlyProcess constraint entity. Here's what OrderOnlyProcess specifies: • Orchestration Process is OrderOnlyProcess • Task is Shipping • Service is UpdateShipment OrderOnlyProcess constrains a change if the OrderOnlyProcess orchestration process is running, and if this process is currently on the Shipping task, and if this process calls the UpdateShipment service. A process constraint entity: • Considers the current position of the transaction in the orchestration process flow. • Can validate required attributes for a fulfillment request, such as a Create Shipment request, Update Shipment request, or Create Reservation request. You can use a predefined process constraint entity or create a new one.

Related Topics

- Constrain Changes to Attributes
- Processing Constraints

Manage Processing Constraints

Set up a processing constraint.

Summary of the Steps

1. Create the record set.
2. Create the validation rule set.
3. Create the processing constraint.
4. Test your set up.

In this example, you set up a processing constraint that prevents your users from changing an order fulfillment line that's part of an orchestration process that's in the shipping stage.

This topic uses example values. You might need different values, depending on your business requirements.

Create the Record Set

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, click **Record Sets > Actions > Add Row**, then set the values.

Attribute	Value
Name	Fulfillment Lines That Belong to Same Customer
Description	A record set created on fulfillment lines that belong to the same customer.
Short Name	FCST
Entity	Order Fulfillment Line

3. In the Fulfillment Lines That Belong to Same Customer area, click **Actions > Add Row**, then set the value.

Attribute	Value
Attribute Name	Bill-to Customer

4. Click **Save > Generate Packages**.

    Generate Packages automatically runs the Generate Constraint Packages scheduled process.

5. Notice that the Confirmation dialog displays your request ID, click **OK**, then wait for the request to finish.

   For example:

   ```
   The concurrent request to generate constraints validation packages was submitted. Request ID: 10650.
   ```

   To monitor the status, go to the Scheduled Processes work area, click refresh, then look for the process ID, such as 10650.

Create the Validation Rule Set

1. Click **Validation Rule Sets**.
2. Click **Actions > Add Row**, then set the values.

Attribute	Value
Name	Shipment Validation Rule Set
Description	The validation rule set for lines with status Awaiting Shipment.
Short Name	SHIP
Validation Type	Table
Entity	Order Fulfillment Line

3. In the Shipment Validation Rule Set area, click **Actions > Add Row**, then set the values.

Attribute	Value
Attribute Name	Status
Validation Operation	Equal to
Value String	Awaiting Shipment If you enter your own attribute, then you must enclose the string with double quotation marks (" ").

4. Click **Save > Generate Packages**.

5. In the Confirmation dialog, click **OK**.
    - Generate Packages deploys all enabled constraints into your implementation.
    - If you create your own processing constraint or modify a predefined one, and if you update to a new release, then you must generate packages again immediately after you do the update.

Create the Processing Constraint

1. Click **Constraints**.
2. Click **Actions > Add Row**, then set the values.

Attribute	Value
Constraint Name	Shipping Constraint
Display Name	Shipping Constraint
Constraint Entity	Order Fulfillment Line
Constrained Operation	Update
Enabled	Contains a check mark.

3. In the Shipping Constraint area, in the Conditions list, click **Actions > Add Row**, then set the values.

Attribute	Value
Group Number	1
Validation Entity	Order Fulfillment Line
Validation Rule Set	Shipment Validation Rule Set
Record Set	Fulfillment Lines That Belong to Same Customer
Message	You can't update the fulfillment line because its in the shipping stage.

4. In the Shipping Constraint area, click **Applicable Roles**, make sure All Roles is enabled, then click **Save**.

Test Your Set Up

1. In the Order Management work area, access a fulfillment line that's in the Awaiting Shipment status.
2. Attempt to change an attribute.

3. Verify that Order Management doesn't allow you to make the change, and displays the message you entered in the Message attribute when you set up the constraint.

Set Other Attributes

Reverse Your Logic

You can use the Invert Validation Rule Set option to reverse your constraint's logic. Assume you create a constraint that has this validation rule set:

```
Status attribute on the fulfillment line entity contains a value of Awaiting Shipment
```

Here's how it works.

Invert Validation Rule Set	Description
Doesn't contain a check mark	Apply the constraint when the Status attribute contains Awaiting Shipment.
Contains a check mark	Apply the constraint when the Status attribute doesn't contain Awaiting Shipment.

Here's another example:

```
Order Type attribute on the order header entity doesn't contain a value
```

Here's how it works.

Invert Validation Rule Set	Description
Doesn't contain a check mark	Apply the constraint when the Order Type attribute doesn't contain a value.
Contains a check mark	Apply the constraint when the Order Type attribute contains a value.

Add AND or OR Logic

Use the Group Number attribute to add AND or OR logic to your condition. For example, set Group Number to 10 for condition x and for condition y. If x and y both evaluate to True, then apply the constraint. For another example, see the Constrain Change subtopic in *Guidelines for Managing Change That Occurs During Order Fulfillment*.

Related Topics

- Processing Constraints
- Roadmap for Setting Up Order-to-Cash

Constrain Changes to Attributes

Constrain the changes your users can make to an attribute.

Order Management comes predefined to apply a processing constraint on some attributes but not on others. In this example, you create a processing constraint that constrains changes on the Latest Acceptable Ship Date attribute on the order header.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Constraint Entities
2. On the Manage Constraint Entities page, in the Entity Type attribute, click **View Entity**, then click **Search**.

   The Manage Constraint Entities page displays view constraint entities and process constraint entities. Order Management can constrain changes to any process constraint entity, so it isn't necessary to enable them for constraint.
3. In the Search Results, click the constraint entity that displays the attribute you must constrain.

   For example, to constrain changes to Latest Acceptable Ship Date on the order header, in the Display Name column, click **Order Header**.
4. In the Attribute Details list, in the row that contains Latest Acceptable Ship Date in the Attribute column, add a check mark to the **Constraint Enabled** option.

Related Topics

- Constraint Entities

Constrain Changes That Users Make in Extensible Flexfields

Constrain changes that your users make in an extensible flexfield.

For example, use a processing constraint to prevent the Order Entry Specialist from updating an extensible flexfield when Order Management already closed the fulfillment line, or to require the user to enter a value in an extensible flexfield at a step of an orchestration process.

- Use the Manage Constraint Entities page to enable an extensible flexfield so you can use it on the Manage Processing Constraints page.
- Extensible flexfields aren't available on the Attributes menu of the Record Sets tab.
- You must enable an extensible flexfield before you can use it.
- A processing constraint prevents changes to the order line after Order Management closes the line. If you set up an extensible flexfield on the order line, and if you don't constrain the extensible flexfield, then the constraint doesn't prevent the user from changing the value in the extensible flexfield.

Summary of the Set Up

1. Enable the extensible flexfield.
2. Create the validation rule set.
3. Create the processing constraint.
4. Test your set up.

Here's the constraint you will create.

- If Order Management already closed the fulfillment line, then don't allow the user to change the value of the Subcontractor ID extensible flexfield.

This topic uses example values. You might need different values, depending on your business requirements.

Enable the Extensible Flexfield

Enable the extensible flexfield so the constraint can reference it.

1. Publish and deploy the extensible flexfield.

   For details, see *Set Up Extensible Flexfields in Order Management*.
2. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Constraint Entities
3. On the Manage Constraint Entities page, set Entity Type to Equals View Entity, then click **Search**.
4. In the search results, select Order Fulfillment Line.
5. In the Attribute Details area, locate the extensible flexfield you must enable, make sure Constraint-Enabled contains a check mark, then click **Save and Close > Done**.

   For this example, enable Subcontractor ID.

Create the Validation Rule Set

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Processing Constraints
2. On the Manage Processing Constraints page, click **Validation Rule Set**.
3. Click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Name	Fulfillment Line Is Closed
Description	Fulfillment line is closed.
Short Name	FLCLOSE
Validation Type	Table
Entity	Order Fulfillment Line

4. In the Details area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Attribute Name	Open
Validation Operation	Equal To
Value String	N

5. Click **Generate Packages**.

   Order Management activates the validation rule set so you can use it in your constraint.

If you add more than one line in the Details area of a validation rule set, then Order Management evaluates them together.

If you select two context and segment attributes, then the attributes must use the same context value. The context is mutually exclusive. Consider this example.

Context	Segment
Dealer Information	ID
Dealer Information	Location
Warranty Information	ID

You can't simultaneously select segment ID for the Dealer Information context and for the Warranty Information context.

Create the Processing Constraint

1. Click **Constraints > Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Name	Constrain Subcontractor ID
Display Name	Constrain Subcontractor ID
Constraint Entity	Order Fulfillment Line

Attribute	Value
Constrained Operation	Update
Attribute Name	Subcontractor:ID

2. In the Details area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Group Number	100
Validation Entity	Order Fulfillment Line
Validation Rule Set	Fulfillment Line Is Closed
Scope	Any
Record Set	Fulfillment Line Default Record Set
Message	The fulfillment line is closed. You can't modify it.

Test Your Set Up

1. In the Order Management work area, open a sales order you already submitted.
2. Verify that Order Management doesn't allow you to modify Subcontractor ID on a fulfillment line that's closed.

Related Topics

- Overview of Using Extensible Flexfields in Order Management
- Set Up Extensible Flexfields in Order Management

Import and Export Processing Constraints

We recommend that you test your processing constraints in a test environment. In this topic, learn how you can import and export processing constraints between environments, such as between your test environment and your production environment.

In this example, assume you must constrain the ORDER_ENTRY_SPECIALIST role from updating the Carrier attribute on a sales order but exclude the ORDER_MANAGER role from this constraint. You need this because updating the carrier might affect your operating costs, so you must limit these changes to the ORDER_MANAGER role.

Assume you created the CARRIER_UPDATE_NOT_ALLOWED constraint at some earlier point, and now must modify it. Here are its current values.

Attribute	Value
Constraint Name	CARRIER_UPDATE_NOT_ALLOWED
Constraint Entity	Order Fulfillment Line
Constraint Operation	Update
Attribute Name	Carrier

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.

Export Your Processing Constraints

1. Go to the Setup and Maintenance work area, then select the Order Management offering.
2. In the Functional Areas list, click the **row** that says Orders.
3. In the Orders area, click **View > Columns**, then add a check mark to the Actions option.
4. In the Search Tasks window, search for Manage Processing Constraints.
5. In the Task list, in the row that has Manage Processing Constraints in the Task column, click **Actions > Export to CSV File > Create New**.
6. On the Export Setup Data to CSV File page, click **Submit**, then, in the Confirmation dialog, note the value of the process name, such as Manage Processing Constraints_20210212_062347_662.
7. On the Setup page, in the row that has Manage Processing Constraints in the Task column, click **Actions > Export to CSV File**, then notice the status that displays.

   If the status says Processing, then wait for it to finish. If the status says Ready for Download, then click **Ready for Download**.
8. On the Export Setup Data to CSV File Results page, click **Actions > Download > CSV File Package**.
9. In the dialog that displays, select the option to save as a file, then click **OK**.
10. Use your browser to access the download location. It depends on your browser. For example, in Firefox, click the **down arrow** on the top banner.
11. Notice that the download creates a single zip file that contains several csv files. Extract these files to a folder of your choice.

Modify Your Processing Constraints

1. Edit the ORA_DOO_CONSTRAINT_ASSIGN.csv file to assign roles to the constraint.

Attribute	Value
ORA_DOO_CONSTRAINT_SETUP.ConstraintName	CARRIER_UPDATE_NOT_ALLOWED
RoleCommonName	ORDER_ENTRY_SPECIALIST

2. Edit the ORA_DOO_CONSTRAINT_EXCLUSION.csv file to exclude roles from the constraint.

Attribute	Value
ORA_DOO_CONSTRAINT_SETUP.ConstraintName	sru_shippingmeth
RoleCommonName	ORDER_MANAGER

3. Save your changes to the zip file.

Import Your Processing Constraints

1. On the Setup Order Management page, In the Search Tasks window, search for Manage Processing Constraints.
2. In the Task list, in the row that has Manage Processing Constraints in the Task column, click **Actions > Import from CSV File > Create New**.
3. On the Import Setup Data to CSV File page, browse to the zip file you modified, then click **Submit**.
4. On the Setup page, in the row that has Manage Processing Constraints in the Task column, click **Actions > Import from CSV File**, then notice the status that displays.
5. Wait for the status to change from Importing Setup Data to Ready for Data Validation.
6. Click the Ready for Data Validation status.
7. On the Import Setup Data from CSV File Results page, verify the value.

Attribute	Value
Status	Completed Successfully

8. Expand the **Business Objects** area, then click the **arrow** in the Go to Task column.
9. On the Manage Processing Constraints page, verify that you can see the role you added.

Note

- The export doesn't export predefined processing constraints. It only exports constraints that you create. So, if the Predefined option on the Manage Processing Constraints page contains a check mark, then you can't modify any part of the constraint.
- The Manage Processing Constraints task in the Setup and Maintenance work area manages the relationships for you, such as between the constraint, conditions, rule set, record set, and so on. If you import or export, then you must manage these relationships manually in the csv files.
- Import and export the entire package of csv files as a single zip file. This will help maintain relationships between files. Don't import and export individual csv files or a subset of them.
- Only import data that exists in Oracle Applications. For example, if you import a role named MY_ROLE, then make sure MY_ROLE exists in Oracle Applications and that its active. If necessary, use the Security Console to add the role and make it active, or don't include the role in your import file. If you include the role, then make sure it matches the value of the RoleCommonName attribute from the Security Console.

Descriptions of the Files

Here are the files that the package contains.

File	Description
ASM_SETUP_CSV_METADATA.xml	Contains the metadata for the virtual object that the download creates. You must not modify this file.
DOO_CHANGE_CONSTRAINT_DEF.csv	Contains all the record sets that you create or reference from your constraints.
ORA_DOO_CONSTRAINT_ASSIGN.csv	Creates a relationship between each constraint on the Constraints tab and the roles in the Constrained Roles area of the Constraints tab.
ORA_DOO_CONSTRAINT_CONDITION.csv	Creates a relationship between each constraint on the Constraints tab of the Manage Processing Constraints page, such as Constraint Name, and data on the Conditions tab, such as Group Number, Message, Enabled, and so on.
ORA_DOO_CONSTRAINT_EXCLUSION.csv	Creates a relationship between each constraint on the Constraints tab and the roles in the Excluded Roles area of the Constraints tab.
ORA_DOO_CONSTRAINT_RECORD_SET_COLUMN.csv	Contains a record set that you use for a conditional constraint.
ORA_DOO_CONSTRAINT_SETUP.csv	Contains details for each constraint from the Constraints tab of the Manage Processing Constraints page, such as the Constraint Name, Constrained Operation, Enabled, On Operation Action, Applicable Roles, and so on. It also contains other attributes that you can't set on the Manage Processing Constraints page, such as the effective dates. The csv file contains these columns but the Manage Processing Constraints page doesn't. Remove these columns from the .csv file.
ORA_DOO_CONSTRAINT_VALIDATION_TEMPLATE.csv	Creates a relationship between the constraints and the validation rule sets.

File	Description
	Contains details from the Validation Rule Sets tab of the Manage Processing Constraints page, such as the Short Name, Validation Type, and so on.
ORA_DOO_CONSTRAINT_VALIDATION_TEMPLATE_COLUMN.csv	Creates a relationship between constraints on the Constraints tab and data in the Details area of the Validation Rule Sets tab, such as Attribute Name, Validation Operation, and Value String.

Related Topics

- Overview of Using Extensible Flexfields in Order Management

Ship

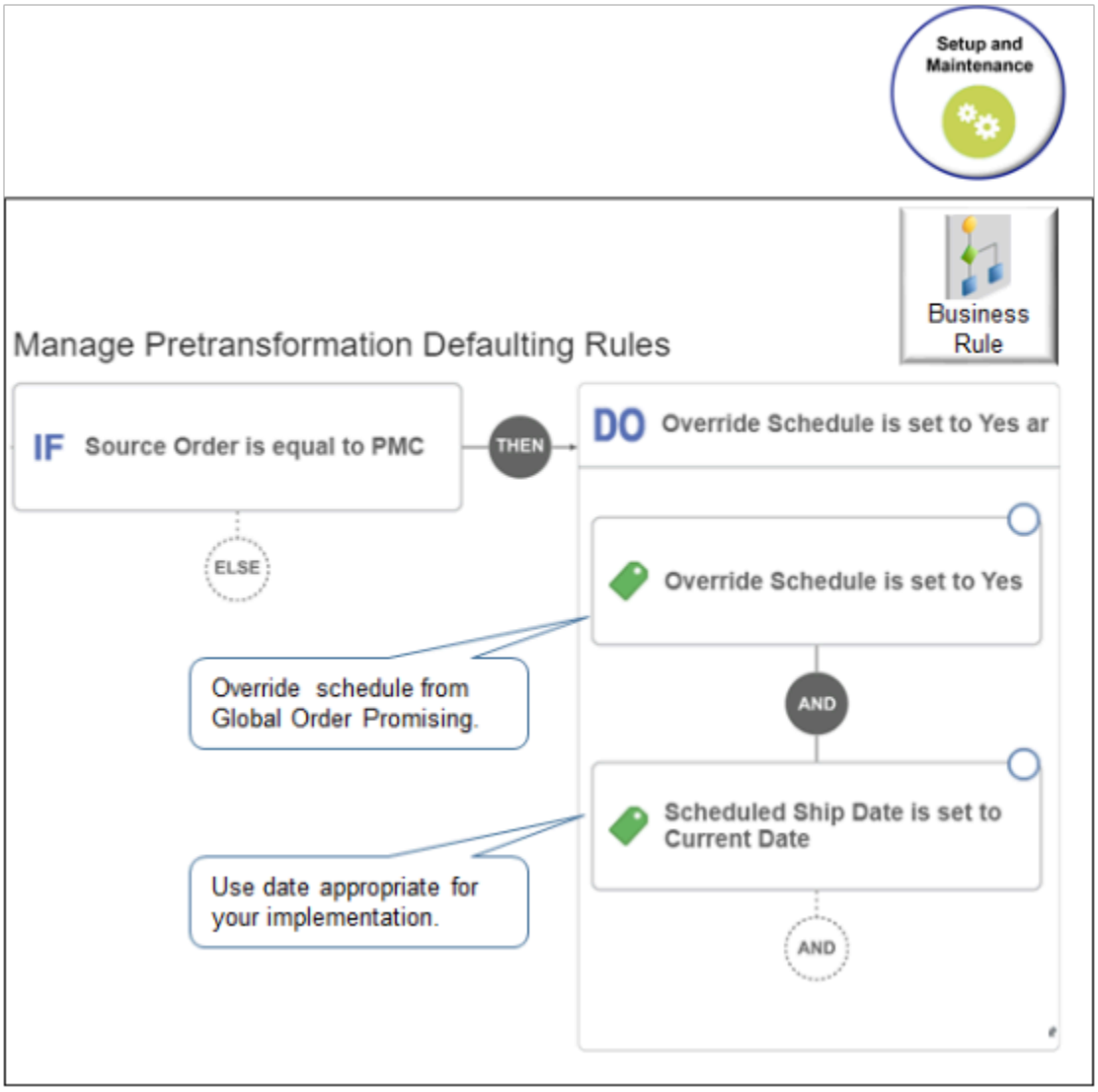
Various Setups

Set Default Values for Scheduled Ship Date

Set the default value for the Schedule Ship Date attribute on the fulfillment line.

You will create an example rule.

- If the source order header includes PMC, then set the default value for Scheduled Ship Date to the current date.



Note

- Use Visual Information Builder.
- Enter Source Order (Order Header) when you create the If clause.
- Enter values when you create the Then clause.
    - Override Schedule (Order Fulfill Line)
    - Scheduled Ship Date (Order Fulfill Line)

Verify You Collected Shipping Attributes

As an option, you can verify you collected shipping attributes that affect the Schedule Ship Date.

The orchestration process uses shipping attributes when it orchestrates shipment during fulfillment.

- Carrier
- Method of Transport

- Service Level

Here's an example payload that includes them.

```
<coresalesorder:ProcessSalesOrderFulfillment>
<coresalesorder:ModeOfTransportCode>AIR</coresalesorder:ModeOfTransportCode>
 <coresalesorder:ServiceLevelCode>1ST</coresalesorder:ServiceLevelCode>
<corecom:PaymentTerm>
 <corecom:Code>24</corecom:Code>
</corecom:PaymentTerm>
<corecom:CarrierPartyReference>
 <corecom:PartyIdentification>
 <corecom:ApplicationObjectKey>
 <corecom:ID>DHL</corecom:ID>
 </corecom:ApplicationObjectKey>
 </corecom:PartyIdentification>
</corecom:CarrierPartyReference>
```

Use SQL to confirm you collected this data.

```
select
mai.instance_code
, mxm.entity_name
, mxm.attribute_name
, mxm.Source_value
, mxm.target_value
from
 MSC_XREF_MAPPING MXM
, MSC_APPS_INSTANCES MAI
where upper(entity_name) in ('WSH_SERVICE_LEVELS','WSH_MODE_OF_TRANSPORT','CARRIERS') AND
 (source_value = 'DHL' or Source_value = 'AIR' or Source_value = '1ST') AND
 MAI.instance_id = MXM.SR_instance_id AND
 MAI.instance_code = 'LEG'
order by
 ENTITY_NAME
, SOURCE_VALUE;
```

Assume your query returns results.

INSTANCE CODE	ENTITY NAME	ATTRIBUTE NAME	SOURCE VALUE	TARGET VALUE
LEG	CARRIERS	CARRIER_ID	DHL	32512
LEG	WSH_MODE_OF_ TRANSPORT	LOOKUP_CODE	AIR	39
LEG	WSH_SERVICE_LEVELS	LOOKUP_CODE	1ST	15

Next, get details about the service level and mode of transport. Query according to the lookup codes from the result of your first query.

```
SELECT
 lookup_code
, lookup_type
, meaning
, enabled_flag
, start_date_Active
, end_date_Active
FROM
 MSC_SR_LOOKUP_VALUES_vl
```

```
WHERE LOOKUP_CODE IN (15,39)
order by
 lookup_code;
```

Here's the result.

LOOKUP_CODE	LOOKUP_TYPE	MEANING	ENABLED_FLAG	START_DATE_ACTIVE	END_DATE_ACTIVE
15	WSH_SERVICE_ LEVELS	1st Class	Y	01-JAN-59	Not applicable
39	WSH_MODE_OF_ TRANSPORT	Air	Y	01-JAN-59	Not applicable

Next, get details about the carrier. Query according to the lookup code for the carrier from the result of your first query.

```
SELECT
 tp_id
 , partner_name
 , party_id
FROM FUSION.MSC_GLOBAL_TRADING_PARTNERS
 where tp_id = 32512;
```

Here's the result.

TP_ID	PARTNER_NAME	PARTY_ID
32512	DHL	DHL

Related Topics

- Overview of Using Business Rules With Order Management
- Use Visual Information Builder
- Use SQL to Query Order Management Data

Set Default Values for Shipping Methods

You can specify the default value that Order Management uses to set the Shipping Method attribute when the Order Entry Specialist creates a sales order.

Assume you must set the default value for the Shipping Method attribute according to the Sold-to Party attribute.

Here's pseudocode for the rule.

```
If the Sold-to Party is Computer Service and Rentals, then set the Shipping Method to USPS Air Express.
```

For details, see *Import Shipping Method*.

Summary of the Setup

1. Get values for your attributes.
2. Create a posttransformation rule.
3. Test your setup.

Get Values for Your Attributes

1. Go to the Order Management work area and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

2. On the catalog line, search for, then add the AS54888 item.
3. Click **Shipment Details > General**, then set the values.

Attribute	Value
Shipping Method	USPS Air Express

4. Click **Save**, then notice the order number. In this example, assume its 521495.
5. Do an SQL.

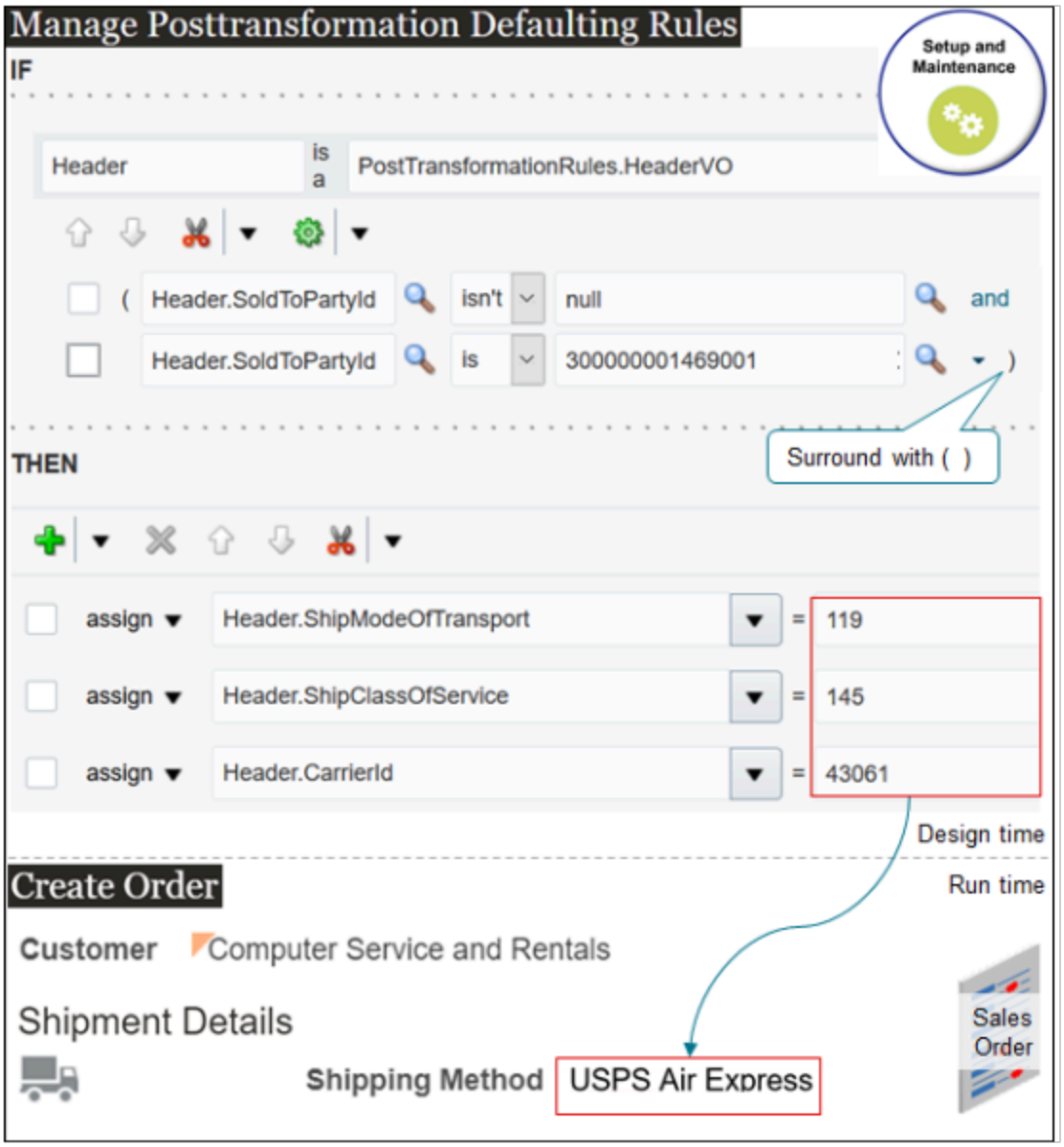
```
SELECT SOLD_TO_PARTY_ID ,
source_order_number ,
order_number ,
CARRIER_ID ,
SHIP_MODE_OF_TRANSPORT,
SHIP_CLASS_OF_SERVICE
FROM doo_headers_all
& ORDER_NUMBER='521495'
ORDER BY creation_Date DESC;
```

For details, see *Use SQL to Query Order Management Data*.

Assume the query returns these values.

Attribute	Value
SHIP_MODE_OF_TRANSPORT	119
SHIP_CLASS_OF_SERVICE	145
CARRIER_ID	43061
SOLD_TO_PARTY_ID	300000001469001

Create a Posttransformation Rule



At run time, if you create a sales order and set the Customer attribute to Computer Service and Rentals, then the posttransformation rule uses the Assign actions that you specify to set the Shipping Method attribute to USPS Air Express.

Try it.

1. Open another browser.

   Use two browsers. One for end-user tasks and the other for administrative tasks. This way, you can toggle back and forth during setup and testing.

2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules
3. On the Manage Posttransformation Defaulting Rules page, create a new rule.
4. In the If area, set the conditions.

Code	Description
`Header is a PosttransformationRules.Header`	Declare the Header variable into the PosttransformationRules dictionary. Get values for attributes of the order header that the orchestration process is currently processing from the header virtual object (VO), then store them in the Header variable.
`Header.SoldToPartyId isn't null`	Make sure the value of the SoldToPartyId attribute on the order header contains a value. You do this to avoid the null pointer exception.
`Header.SoldToPartyId is 300000001469001`	Make sure the value of the SoldToPartyId attribute on the order header contains the value 300000001469001. This value identifies the Computer Service and Rentals customer that you identified when you ran the SQL.

Note

- Add a check mark to the Advanced Mode option when you create the rule.
- Add the two conditions.
- Add a check mark to the square box on each condition, then, above Header.SoldToPartyId, click Surround, and verify that the rule adds a pair of parentheses ( ) that surrounds the conditions.

  This set up makes sure the rule evaluates the conditions with AND logic. If both conditions evaluate to true, then the rule will assign the values that you specify in the Then section. If you don't add the parentheses, then the rule evaluates the conditions with OR logic, which you don't want.

5. In the Then area, add three Assign actions.

Code	Description
`Header.ShipModeOfTransport = 119`	Set the mode of transport to the value you identified from the SQL query. In this example, 119 means Air.
`Header.ShipClassOfService = 145`	Set the mode of transport to the value you identified from the SQL query. In this example, 145 Express.

Code	Description
`Header.CarrierId = 43061`	Set the mode of transport to the value you identified from the SQL query. In this example, 43061 means USPS.

6. Click **Save > Release**.

   For details, see *Overview of Using Business Rules With Order Management*.

Test Your Setup

1. Go back to your other browser and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

2. Click **Shipment Details > General**, then verify that Order Management automatically set the value.

Attribute	Value
Shipping Method	USPS Air Express

Consider the Request Type

Promising attempts to fulfill the sales order within the requested date or with minimal delay when it promises the order according to the attributes that you set.

If You Set the Request Type Attribute on the Order Line To	Then Promising Tries to Match The
Ship On, or if you set the PromisingType attribute to Ship in a web service payload	Scheduled ship date with the requested ship date. Promising uses the default shipping method because the method doesn't affect whether Promising can promise the order on time or with minimal delay.
Arrive On, or if you set the PromisingType attribute to Arrival in a web service payload	Scheduled arrival date with the requested arrival date. If Promising can't meet the requested arrival date or can't promise with minimal delay, then it uses the fastest shipping method.

Note

If you specify the shipping method on the order line, then Promising uses that method.

You must manage the transit time for each shipping method that you create between a source and a destination.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Manufacturing and Supply Chain Materials Management
    - Functional Area: Carriers and Transit Times
    - Task: Manage Transit Times
2. Collect your changes. For details, see *Collect Planning Data for Order Management*.

For details, see *Set Default Values for Shipping Methods*. For details about the Global Order Promising web service, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Global Order Promising**.

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules
- Import Shipping Method
- Use SQL to Query Order Management Data

Set Warehouse According to the Value of Some Other Attribute

Set up Order Management so it sets the default value for the Warehouse attribute on the fulfillment line according to the value of the Business Unit attribute on the order header.

Assume the Vision Operations business unit sells desktop computer systems, and it uses the same warehouse to build them. It uses the M5 Denver Manufacturing warehouse to fulfill all of its sales orders.

Summary of the Setup

1. Get values for your attributes.
2. Create a posttransformation rule.
3. Test your setup.

Get Values for Your Attributes

1. Get the ID for the business unit.
    - Do an SQL.

```
SELECT haotl.NAME BU_NAME,
 hao.organization_id
FROM
 fusion.HR_ALL_ORGANIZATION_UNITS_F hao,
 fusion.HR_ORGANIZATION_UNITS_F_TL haotl,
 fusion.HR_ORG_UNIT_CLASSIFICATIONS_F houc,
 fusion.hr_organization_information_f hoi
WHERE
 hao.ORGANIZATION_ID = haotl.ORGANIZATION_ID
 AND houc.ORGANIZATION_ID = haotl.ORGANIZATION_ID
 AND houc.CLASSIFICATION_CODE = 'FUN_BUSINESS_UNIT'
 AND hoi.ORGANIZATION_ID = haotl.ORGANIZATION_ID
 AND hoi.ORG_INFORMATION_CONTEXT = houc.CLASSIFICATION_CODE
 AND TRUNC(SYSDATE) BETWEEN haotl.EFFECTIVE_START_DATE AND haotl.EFFECTIVE_END_DATE
 AND haotl.LANGUAGE = USERENV('LANG')
 AND haotl.EFFECTIVE_START_DATE = hao.EFFECTIVE_START_DATE
 AND haotl.EFFECTIVE_END_DATE = hao.EFFECTIVE_END_DATE
 AND upper(Haotl.name)LIKE '&BUSINESS_UNIT_NAME%'
```

```
ORDER BY BU_NAME;
```

For details, see *Use SQL to Query Order Management Data*.

- In the query result, locate the row that contains Vision Operations in the BU_NAME column. Assume the query returns these values.

BU_NAME	ORGANIZATION_ID
Vision Operations	300000001616323
Customer	Computer Service and Rentals

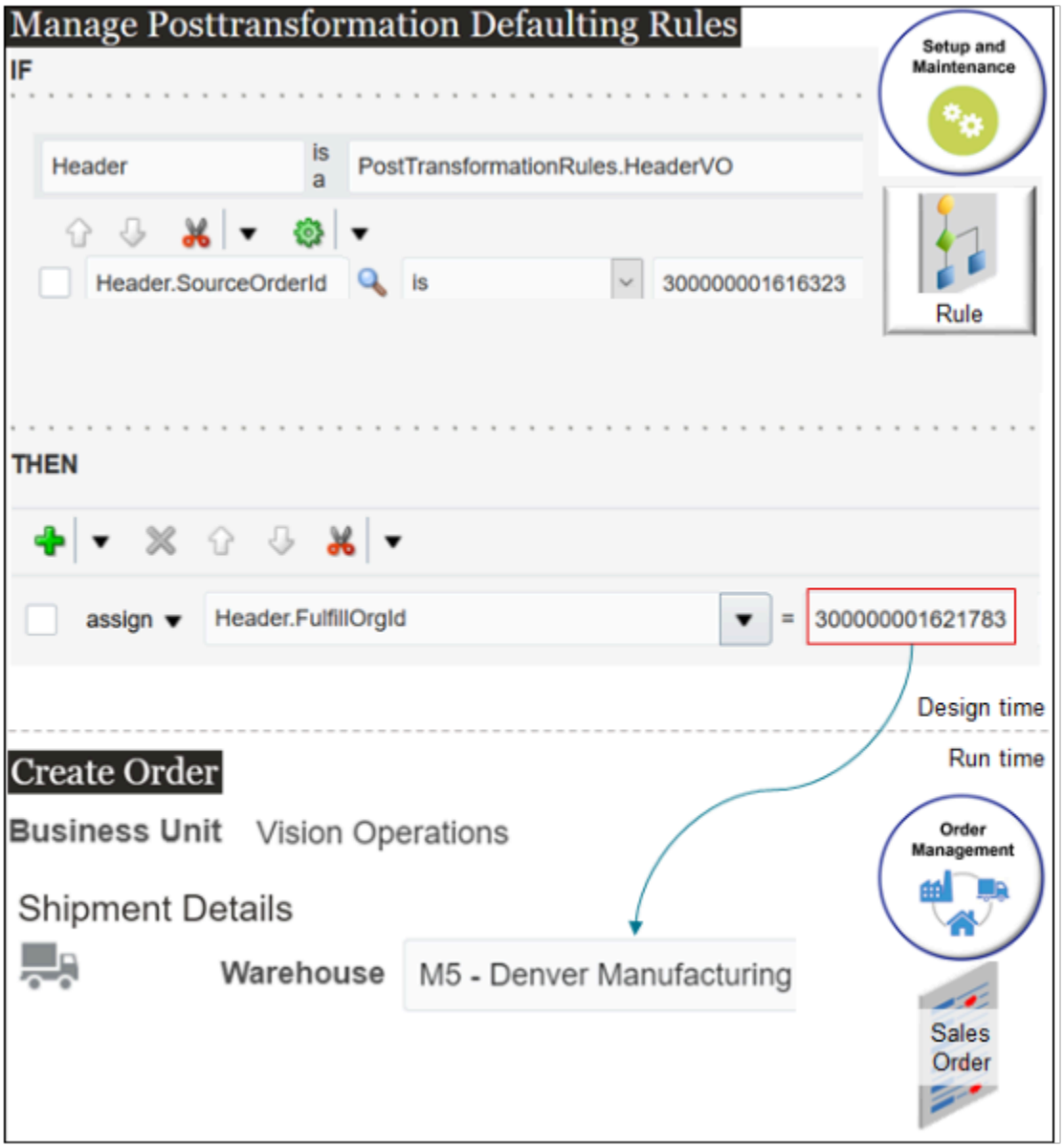
2. Get the ID for the warehouse.
   - Do an SQL.

```
SELECT
 haotl.NAME,
 iop.ORGANIZATION_CODE,
 iop.ORGANIZATION_ID
FROM
 fusion.Inv_Org_Parameters iop,
 fusion.HR_ORGANIZATION_UNITS_F_TL haotl
WHERE
 haotl.organization_id = iop.business_unit_id
 AND haotl.LANGUAGE = USERENV('LANG')
 AND upper(iop.ORGANIZATION_CODE) LIKE '&INVENTORY_CODE%'
ORDER BY haotl.NAME,
```

   - In the query result, locate the row that contains M5 Denver Manufacturing in the ORGANIZATION_CODE column. Assume the query returns these values.

ORGANIZATION_CODE	ORGANIZATION_ID
M5 Denver Manufacturing	300000001621783

Create a Posttransformation Rule



Note

- To create the And logic in the If area, add the `Header.SourceOrderId is 300000001616323` test, then to the right of `Header is a PosttransformationRules.HeaderVO`, click **Add Pattern**.
- At run time, if you create a sales order and set the Business Unit attribute on the order header to Vision Operations, then the posttransformation rule uses the Assign action that you specify to set the Warehouse attribute to M5 Denver Manufacturing.

Try it.

1. Open another browser.

   Use two browsers. One for end-user tasks and the other for administrative tasks. This way, you can toggle back and forth during setup and testing.

2. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules
3. On the Manage Posttransformation Defaulting Rules page, create a new rule.
4. In the If area, set the conditions.

Code	Description
`Header is a PosttransformationRules.Header`	Declare the Header variable into the PosttransformationRules dictionary. Get values for attributes of the order header that the orchestration process is currently processing from the header virtual object (VO), then store them in the Header variable.
`Header.SourceOrgId is 300000001616323`	Make sure the value of the SourceOrgId attribute on the order header contains 300000001616323. This value identifies the Vision Operations business unit that you identified when you ran the SQL. Order Management uses the SourceOrgId attribute to identify the business unit.

5. In the Then area, add an Assign action.

Code	Description
`Header.FulfillOrgId = 300000001621783`	Order Management uses the FulfillOrgId attribute to identify the warehouse. This value identifies the M5 Denver Manufacturing warehouse that you identified when you ran the SQL.

6. Click **Save > Release**.

    For details, see *Overview of Using Business Rules With Order Management*.

Test Your Setup

1. Go back to your other browser and create a sales order.

Attribute	Value
Business Unit	Vision Operations

2. Click **Shipment Details > Supply**, then verify that Order Management automatically set the value.

Attribute	Value
Warehouse	M5 Denver Manufacturing

Related Topics

- Overview of Using Business Rules With Order Management
- Manage Pretransformation Rules
- Import Shipping Method
- Use SQL to Query Order Management Data

Set Promised Ship and Arrival Dates on Fulfillment Lines

The Promised Ship Date and the Promised Arrival Date track the dates when you agree to ship the item to your customer. You can't set these dates in the Order Management work area, but you can set them through other technologies.

You use these dates only to track or report on fulfillment. You use the Actual Ship Date and Actual Delivery Date to determine when the item actually shipped and arrived.

Use File-Based Data Import

Use attributes on the DOO_ORDER_LINES_ALL_INT worksheet in the Order Import template.

- Promise Ship Date
- Promise Arrival Date

See instructions on the worksheet for format requirements. For details, see *Overview of Importing Orders Into Order Management*.

Use a Web Service

Use the createOrders operation of the Order Import Service to import the dates. Include them in the Line node.

Assume you generally require 3 days to finish shipping for an item, so you set the PromiseArrivalDate so it happens 3 days after RequestedShipDate.

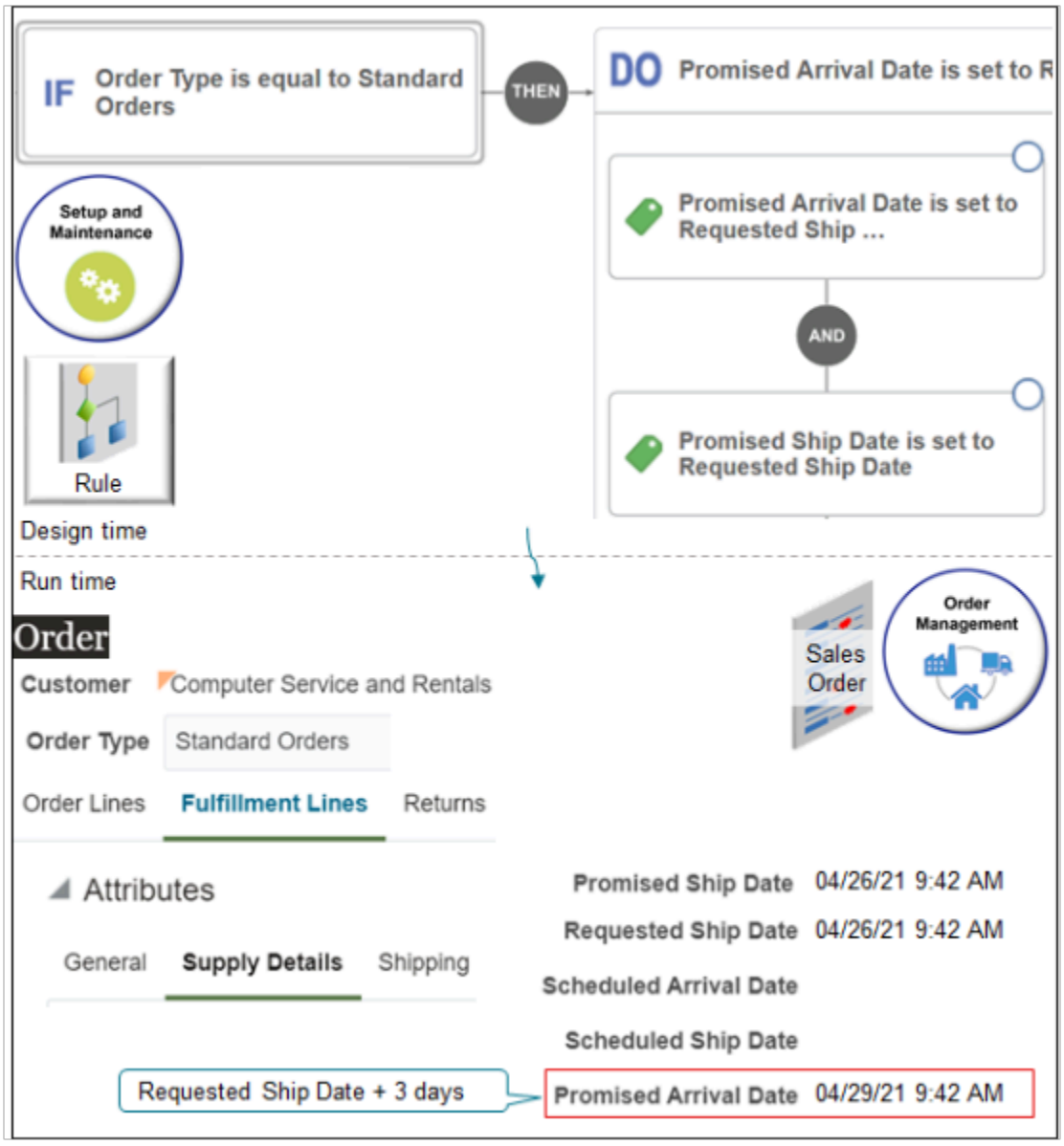
```
<ns2:Line>
 <ns2:RequestedShipDate>2021-04-26T09:42:00</ns2:PromiseArrivalDate>
 <ns2:PromiseShipDate>2021-04-26T09:42:00</Ns2:PromiseShipDate>
 <ns2:PromiseArrivalDate>2021-04-29T09:42:00</ns2:PromiseArrivalDate>
```

For details, see *Web Services That You Can Use to Integrate Order Management*.

Examine an example that uses the createOrders operation. For details, see *Import Returns When You Split the Original Order Line*.

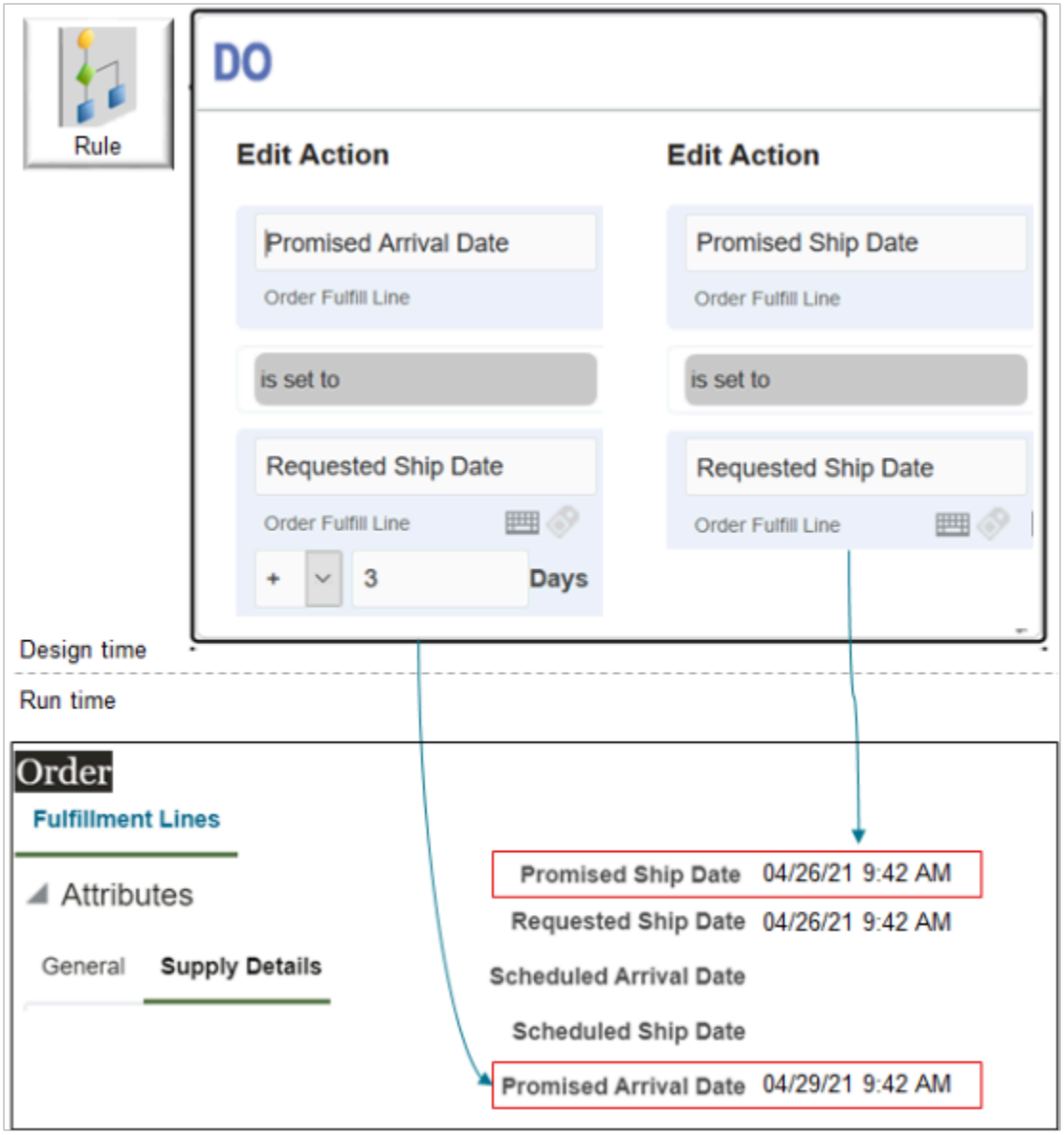
Use a Business Rule

Assume you must apply the rule to all standard orders.



You use the Setup and Maintenance work area to create a pretransformation rule. Order Management runs the rule on each standard sales order at run time.

Here's a detailed view of the actions.



They say.

- Set the Promised Arrival Date on the fulfillment line to the value that the Requested Ship Date attribute on the fulfillment line contains, plus 3 days.
- Set the Promised Ship Date on the fulfillment line to the value that the Requested Ship Date attribute on the fulfillment line contains.

Try it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Pretransformation Rules for Sales Orders
2. Create a new rule.

Attribute	Value
Name	Set Promised Ship and Arrival Dates

3. Create the If statement.
    - In the If area, click **New Condition**.
    - In the Create Condition dialog, enter `Order`, wait a moment, then click **Order Type (Order Header)**.

      Order Type (Order Header) indicates that you're referencing the Order Type attribute on the order header of the sales order.
    - Create the condition, then click **OK**.

      ```
      Order Type is equal to Standard Orders
      ```
4. Click **Then > Do**.
5. In the DO area, click **New Action > Set a Value**.
6. Create the Then statement for the Promised Arrival Date.
    - In the Create Action Dialog, enter `Promised Arrival Date`, then click **Promised Arrival Date (Order Fulfill Line)**.
    - Click **Attribute**.
    - Enter `Requested Ship Date`, then click **Requested Arrival Date (Order Fulfill Line)**.
    - Click `Add Arithmetics`. Its the f(x) icon.
    - Enter `3` in the window, then click **OK**.
7. Create the Then statement for the Promised Ship Date.
    - Click **And > Set a Value**.
    - Repeat step 6, except create this statement.

      ```
      Promised Ship Date is set to Requested Ship Date.
      ```
8. Publish your rule.

Test Your Setup

1. Go to the Order Management work area, create a sales order, and set the values.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Order Type	Standard Orders

2. Add the AS54888 to an order line, then click **Submit**.
3. On the Order page, click **Actions > Switch to Fulfillment View**.
4. Click **Fulfillment Lines**.
5. In the Attributes area, click **Supply Details**, then verify the attribute values.
    - Verify that the Promised Arrival Date happens 3 days after Requested Ship Date.
    - Verify that the Promised Ship Date equals the Requested Ship Date.

Related Topics

- Overview of Using Business Rules With Order Management
- Import Different Kinds of Data
- Web Services That You Can Use to Integrate Order Management
- Import Returns When You Split the Original Order Line

Guidelines for Managing Shipment Sets

Apply guidelines when you use shipment sets.

- If Order Management applies a hold on one or more lines in a shipment set, then it applies the hold on the entire shipment set.
- If the fulfillment line is part of a shipment set, then you can't:
    - **Split the fulfillment line**. To split the line, remove it from the shipment set, add a new line, then split your new line.
    - **Substitute an item**. To substitute, remove the line that contains the item you must substitute from the shipment set, then add a new line that contains the substitution.
- If you already shipped the fulfillment line, or if Order Management already created a request to invoice the line, then the Fulfillment Line Shipment Set Update constraint prevents you from updating the shipment set. The constraint comes predefined as enabled. You can disable it.
- If you remove an order line from a shipment set, then Order Management doesn't update the orchestration plan.
- If you manually schedule a shipment set, and if you select an option for one or more fulfillment lines, and if these lines are part of a shipment set, then Order Management applies the scheduling option that you select to all of the fulfillment lines that are part of the shipment set.
- If you include items that can ship with items that can't ship in the same shipment set, then you can include a pause step in your orchestration process only if you need the pause so you can invoice the shippable and nonshippable items together. For example, you add line 1 that contains the shippable AS54888 Desktop Computer and line 2 that contains the nonshippable warranty that covers the AS54888 in the same set.

Import Shipment Sets

Import your shipment set through the OrderImportService web service or through file-based data import.

Here's part of an example payload that includes a shipment set.

```
<ns2:Line>
 <ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
 <ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
 <ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
 <ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
 <ns2:ProductNumber>AS54888</ns2:ProductNumber>
 <!--Parameterize-->
 <ns2:OrderedQuantity>15</ns2:OrderedQuantity>
```

```
<!--Parameterize-->
<ns2:OrderedUOM>Each</ns2:OrderedUOM>
<!--ns2:OrderedUOMCode>Ea</ns2:OrderedUOMCode-->
<ns2:RequestingBusinessUnitName>Vision Operations</ns2:RequestingBusinessUnitName>
<ns2:ParentLineReference/>
<ns2:RootParentLineReference/>
<ns2:ShippingInstructions/>
<ns2:PackingInstructions/>
<ns2:ScheduleShipDate/>
<ns2:RequestedShipDate>2019-10-14T01:08:52Z</ns2:RequestedShipDate>
<ns2:RequestedFulfillmentOrganizationCode>M1</ns2:RequestedFulfillmentOrganizationCode>
<ns2:ShipToAddress1>102, CityView</ns2:ShipToAddress1>
<ns2:ShipToCity>CHATTANOOGA</ns2:ShipToCity>
<ns2:ShipToPostalCode>37401</ns2:ShipToPostalCode>
<ns2:ShipToState>TN</ns2:ShipToState>
<ns2:ShipToCountry>US</ns2:ShipToCountry>
<ns2:ScheduleShipDateTime>2016-12-20T10:10:10</ns2:ScheduleShipDateTime>
<ns2:ScheduleArrivalDateTime>2016-12-20T10:10:10</ns2:ScheduleArrivalDateTime>
<ns2:ShippingCarrier>DHL</ns2:ShippingCarrier>
<ns2:ShippingServiceLevel>2nd day air</ns2:ShippingServiceLevel>
<ns2:ShippingMode>Air</ns2:ShippingMode>
<ns2:ShipSetName>SS1</ns2:ShipSetName>
```

Note

- Use the ShipSetName attribute to specify the shipment set. Include the attribute in the Line hierarchy. This example sets the name to SS1, and its part of the hierarchy for source transaction line 101. It adds line 101 to shipment set SS1.
- Set ShipSetName to the same value in each line that you need in the set. Assume your set includes lines 101 and 102. Set ShipSetName in line 102 to SS1.
- Make sure each line that's part of the shipment set uses the same value for attributes that affect the set, such as Warehouse. For details, see *Ship Order Lines in Shipment Sets*.
- To get the entire payload for this example, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Payloads and Files attachment. Open the attachment, then open source_order_with_shipment_set.txt.

Assign Your Orchestration Processes

Make sure the assignment rule that you use to assign your orchestration process to a shipment set assigns the same process to each fulfillment line in the set.

Assume you sell the AS54888 laptop computer and the AS9000 Computer Monitor as a package and want them to arrive at your customer's site at the same time. You can use a shipment set to do this.

Use the Setup and Maintenance work area to:

1. Create a shipment set named Computer Shipment Set.
2. Create an orchestration process named My Orchestration Process.
3. Create an assignment rule that assigns any order line that has the Computer Shipment Set in the Shipment Set attribute to My Orchestration Process.
4. Create a sales order that includes the AS54888 on line 1 and the AS9000 on line 2. Set the Shipment Set attribute to My Shipment Set on line 1 and line 2, then click Submit.

Here's some logic that you can use in your assignment rule to make sure you assign the same orchestration process to each line in the set:

```
If Shipment Set (Order Fulfill Line) = Computer Shipment Set, then Process Name is set to
 MyOrchestrationProcess.
```

We recommend that you use the Shipment Set attribute in your If statement. Don't use any other attribute because you can't guarantee that it will always evaluate to the expected value.

For example, we recommend that you don't use a rule like this:

```
If Process Name is Blank And Customer PO Number is Not Blank, then Process Name is set to Customer PO
 Number
```

We recommend that you use Visual Information Builder to create the assignment rule. Don't use Oracle Business Rules.

For details, see *Guidelines for Assigning Orchestration Processes*.

Related Topics

- Cancel Order Lines That Remain in the Same Status
- Ship Order Lines in Shipment Sets

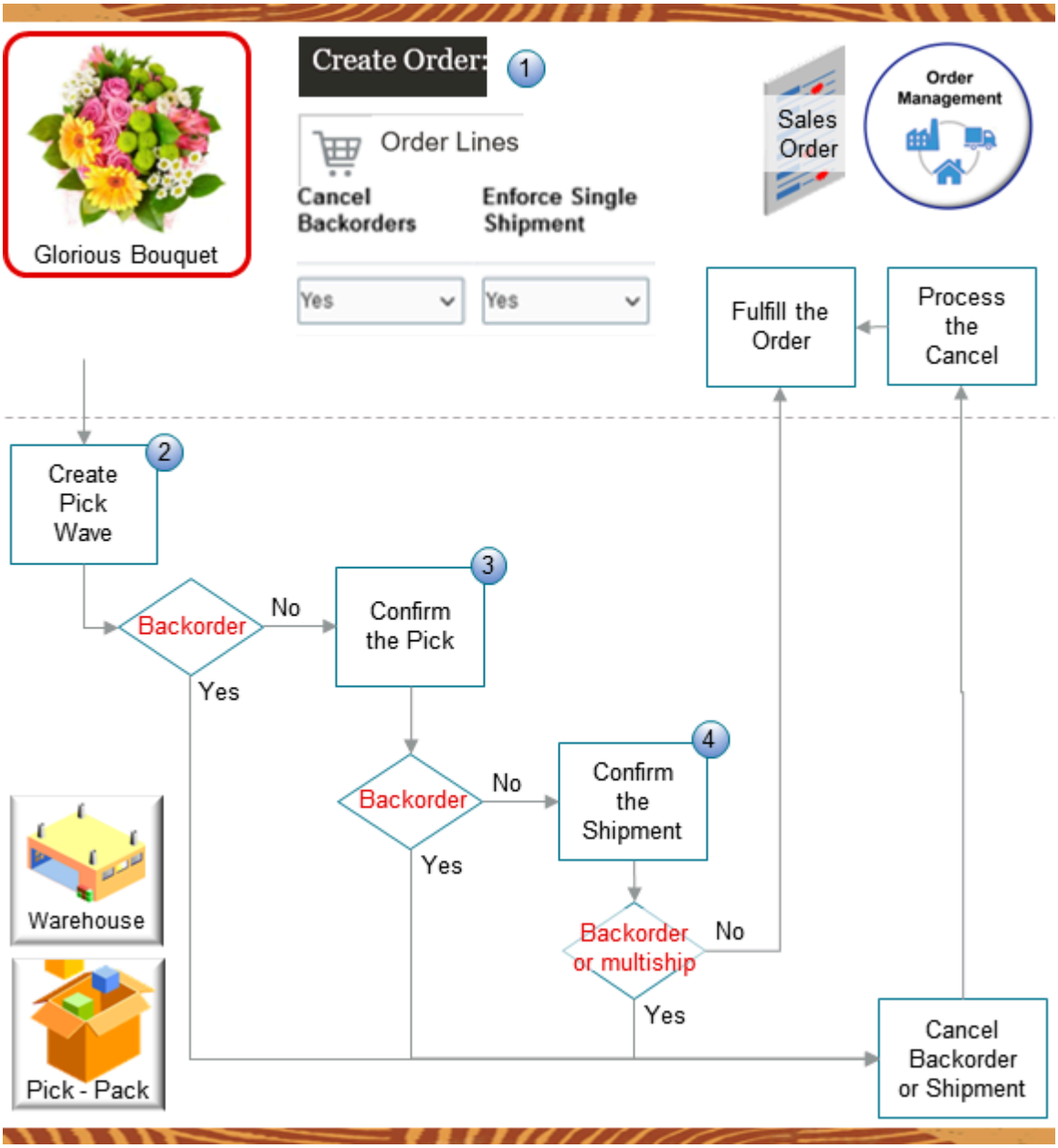
Cancel Backorders and Enforce Single Shipments

Set up Oracle Order Management so it automatically cancels backorders and enforces single shipments.

The contract that you have with your customer might include specific fulfillment requirements, such as whether your customer accepts a backorder, or will accept receiving the order line in more than one shipment.

You can set up Order Management so it considers the on-hand supply that's available to meet demand when you confirm the pick or confirm the shipment in Oracle Shipping. If supply can't meet demand, then you can automatically cancel the quantity on the line that Order Management can't fulfill. You can also automatically cancel the quantity that remains open after the first shipment even if on-hand quantity is available. For example, when your customer only accepts a single shipment for each order line.

Summary



Here's a summary of how it works:

1. Assume you sell an item named Glorious Bouquet. You create a sales order in the Order Management work area, add the bouquet to an order line, set these attributes on the order line, then submit the sales order:

Attribute	Value
Quantity	10
Cancel Backorders	Yes

Attribute	Value
	Automatically cancel any quantity that you can't fulfill because on-hand quantity can't meet demand when Shipping creates a pick wave, confirms the pick slip, or confirms shipment.
Enforce Single Shipment	Yes Automatically cancel any quantity that remains open on the order line after Shipping confirms and closes the first shipment.

2. You create a pick wave in the warehouse. If Shipping:
    - Hasn't put any quantity on backorder, then you proceed to confirm the pick.
    - Has put some or all of the quantity on backorder, then Shipping allocates the quantity that's in stock, cancels the backordered quantity, then sends details about the cancel to Order Management so Order Management can process it and update the order line.
3. You confirm the pick in the warehouse. If Shipping:
    - Hasn't put any quantity on backorder, then you proceed to confirm the shipment.
    - Has put some or all of the quantity on backorder, then Shipping prepares the quantity that's in stock, cancels the backordered quantity, then sends details about the cancel to Order Management so Order Management can process it and update the order line.
4. You confirm the shipment in the warehouse. If Shipping:
    - Hasn't put any quantity on backorder, then it finishes the shipment and sends the order line to Order Management so Order Management can update the line.
    - Has put some or all of the quantity on backorder, then Shipping ships the quantity that's in stock, cancels the backordered quantity, then sends the order line to Order Management so Order Management can process the cancel and fulfill the order.

Shipping does these checks during various pick and pack operations in the warehouse.

Order Management sends the attribute values to Shipping after you submit the sales order. You can then go to the Inventory Management work area and view them in the Cancel Backorders attribute and the Enforce Single Shipment attribute on the Manage Shipment Lines page.

For background, see *Using Shipping*.

Cancel Backorders

You might need to cancel a backorder at different times during fulfilment, such as when you release a pick, confirm a pick, or confirm a shipment. For example:

- Quantity isn't available when you release the pick.
- Quantity is available but Shipping finds that some of the quantity is damaged or there's a problem with packing when it confirms the pick.
- Quantity is available but you want to cancel the current line so you can use some or all of that line's quantity for a different, more urgent sales order.

Your customer might also have some specific requirements:

- Fulfill the quantity that's available but cancel the quantity that isn't.

- Cancel an order line that you can't fulfill.
- If you can't fulfill any line in the sales order, then cancel all the lines in the order.
- Ship all quantity in a single shipment and cancel any quantity that would need another shipment.

If you add a check mark to the Cancel Backorders attribute on the order line, and if there isn't enough on-hand quantity to meet demand, then Shipping automatically cancels the backorder when you start to release or confirm the pick, or confirm the shipment.

Shipping won't always cancel the shipment line. Here are some examples where enabling the Cancel Backorders attribute on the order line won't result in Shipping cancelling the shipment line:

- The shipment line is part of a pick-to-order configured item or kit, and Shipping backordered only part of that item or kit.
- The shipment line is part of a shipment set and Shipping backorders it when it confirms the pick.
- You go to the Inventory Management work area, then use the Backorder action or the Submit for Cycle Count action on the Manage Shipment Lines page to backorder the shipment line.

Enforce Single Shipment

If you enable the Enforce Single Shipment attribute on an order line, and if:

- Shipping can ship only part of the order line's quantity, Shipping will cancel the quantity that it can't ship.
- You use the Shipment Transaction Requests REST API or the Perform Shipping Transaction Import FBDI to ship only part of an order line, and if Shipping hasn't already sent the shipment line to the warehouse, then Shipping will cancel that line. If Shipping already sent the line to the warehouse, then Shipping won't cancel it.
- You assign more than one shipment to the sales order, then you can't confirm the shipment, send a manifest request, or create a fiscal document for the sales order.
- You use Oracle Transportation Management or some other transportation management system to plan the shipment's transportation, then you can't assign more than one shipment to the order line.

Cancel the Shipment Line

Shipping cancels the shipment line, and the Process Responses from Order Fulfillment scheduled process makes these updates on the fulfillment line:

- Uses the predefined Cancel Backorders reason to update the ordered quantity, canceled quantity, and the cancel's reason code. Order Management doesn't split the order line or revise the sales order, but it does update the order line.
- Uses the updated order quantity to recalculate the charges on the order line.

Note

- You must run the *Update Sales Order Totals* scheduled process to update the order header's totals.
- If an order line has a credit authorization, then you must run the Adjust Amounts for Credit Authorizations scheduled process so it can adjust the amounts according to the revised ordered quantity.

We recommend that you set up these scheduled processes so they run on a schedule. This way, you don't have to run them manually.

If the cancel happens in a back-to-back flow, then you must set the Cancel Supply When Canceling Backorders order management parameter to Yes to make sure you update the quantities on purchase order, work order, or transfer order. For more, see *Manage Order Management Parameters*.

If you enable the Cancel Backorders business event, then Order Management will raise that event when Shipping starts the cancel. For more, see *Overview of Sending Notifications from Order Management to Other Systems*.

Shipment Sets

You can use a shipment set to control how you want to cancel order lines. Assume you enable the Cancel Backorders attribute. Here's how Shipping handles the set:

Shipment Set	If Entire Quantity Isn't Available
All lines are in the same set.	For any line, then cancel all lines in the sales order.
Each line is in its own set.	For a line, then cancel only that line. Apply this logic to each line.
No line is in a set.	For a line, then ship the quantity that's available and cancel the remaining quantity on that line. Apply this logic to each line.

If the shipment line is a component in a configured item or kit, or is part of a shipment set, and if Shipping backorders the line when confirming the pick, the Shipping doesn't cancel the line.

For details about Enforce Shipment Set, see *Impact of Shipping Parameters for Kit Items on Transfer Orders* and *Considerations for Managing Shipping Parameters*.

Try It

1. Go to the Setup and Maintenance work area, then select the Manufacturing and Supply Chain Materials Management offering.
2. Click **Change Feature Opt In**.
3. In the row that has Shipping in the Name column, click the pencil.
4. Enable the Cancel Sales Order Backorders Automatically and Enforce Single Shipment feature, then click Done.

   After you opt in, you must not disable this feature at any future time. Disabling after you enable will cause problems with downstream fulfillment. This feature will always be enabled starting with update 24D.
5. Enable the Cancel Backorders business event.

   Order Management will automatically raise that event when Shipping cancels the line. It might also raise other events depending on processing requirements, such as the Fufillment Line Status Update event or the Order Attribute Update event. For more, see *Overview of Using Business Events with Order Management*.
6. Set the Cancel Supply When Canceling Backorders parameter. Use this parameter to control behavior in your back-to-back flow. Set a value:
   - Yes. If Shipping canceled only part of the shipment line, then the Process Responses from Order Fulfillment scheduled process will update Oracle Supply Chain Orchestration's supply order so the supply order has the updated the quantity. It will also update, cancel, or close any purchase order, work order, or transfer order that's attempting to fulfil the remaining quantity.
   - No. Don't update or cancel the supply order, and don't cancel or close the transfer order, purchase order, or work order. Set to No when you want to manually review and cancel the supply order.

   If Shipping confirms the shipment, and if there's no more quantity on the line waiting to ship, then the scheduled process will close the supply order even if you set this parameter to No.

   For more, see *Manage Order Management Parameters*.

7. Set up these scheduled processes. Set up their schedules so they run at frequent intervals:
    - **Adjust Amounts for Credit Authorizations**. If your order line will have a credit authorization, then you must run this scheduled process so it can adjust your customer's credit authorization amounts. If Shipping cancels:
        - Part of the quantity. This scheduled process will use the updated amount when it reauthorizes the credit according to the revised quantity that's on the line after the cancel.
        - The entire quantity. This scheduled process will cancel the credit authorization on the order line.
    - *Update Sales Order Totals*. Use this process to update the totals on the order header after Shipping cancels the backorder.
8. Collect and refresh.
    - Collect data for the Order Orchestration Reference Objects entity. This collection will collect the predefined Cancel Backorders reason code. For more, see *Collect Planning Data for Order Management*.
    - Run the Refresh and Start the Order Promising Server scheduled process. For more, see *Refresh the Order Promising Server for Order Management*.
9. Test your set up.

    To get detailed examples that walk you through the end-user experience for this feature, including helpful screen prints, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Cancel Backorder and Enforce Single Shipment attachment. The attachment has these examples:

    - Cancel Backorder
    - Cancel Backorder and Enforce Single Shipment
    - Use Business Events
    - Apply Credit Adjustments
    - Fulfill Back-to-Back Orders
    - Cancel Backorders with Shipment Tolerances

Guidelines

You can use this feature with:

- Standard items
- Assemble-to-order items
- Pick-to-order items
- Items that use dual units of measure
- Kits
- Shipping tolerances

You can't use this feature with covered items, coverage items, or subscriptions.

You can use this feature with these flows:

- Ship from warehouse
- Back-to-back

You can't use this feature with a drop shipment, transfer order, or return order.

Note

- You can set the Cancel Backorders attribute and the Enforce Single Shipment attribute only on the order line. You can't set them on the order header, and you can't default their values from the header.
- You can't modify the Cancel Backorders attribute or the Enforce Single Shipment attribute after you submit the sales order. However, you can create an order revision, add a new line to the revision, and then set them on the new line.
- If you enable the Enforce Single Shipment attribute, then you must also enable the Cancel Backorders attribute.
- If you use your own orchestration process instead of a predefined one to fulfill your item, then you must include the Shipped value and Canceled value in the Wait for Shipment task's exit criteria. If you don't, then the Shipment task might remain active even if Order Management cancels the order line. For more, see *Guidelines for Setting Up Orchestration Processes*.
- Cancel Backorders controls the cancel when you release the pick, and when you confirm the pick and confirm shipment. Enforce Single Shipment applies only when you confirm the shipment.

Cancel Backorder

- You can enable the Cancel Backorders attribute and also specify a shipping tolerance on the order line. Shipping will cancel the backordered quantity when it confirms the shipment, stage the remaining quantity, then cancel part of that staged quantity according to the shipping tolerance that you set on the line. For background, see *Set Up Shipping Tolerances in Order Management*.
- The Cancel Backorders attribute applies only on items that you can physically ship. It doesn't apply on items that you can't ship, such as a subscription.
- Shipping might cancel some of the quantity on an order line when it releases the pick, another quantity when it confirms the pick, and yet another quantity when it confirms shipment, but Shipping will only cancel backordered quantity. If you decide to keep some quantity in a staged status during shipping, then Shipping won't cancel that quantity.
- Oracle Pricing doesn't recalculate a volume discount after a cancel. Assume you have an order line where Pricing applies a $10 discount when the quantity exceeds 100. The quantity on the line is 110, so Pricing applies the $10 discount. Shipping cancels a quantity of 20, and now the line's quantity is 90. Pricing won't recalculate the discount. Pricing will keep the $10 discount even though the quantity is now 90.
- If you revise a sales order, cancel a quantity in that revision, and provide a reason for the cancel, then Order Management will use that reason, but it won't overwrite the predefined Cancel Backorders reason for any subsequent backorder that you cancel.

If you send the shipment line to your warehouse, and if you use the:

- Shipment Line Change Requests REST API to backorder a shipment line, then Shipping will cancel the backordered quantity.
- Shipment Transaction Requests REST API or the Perform Shipping Transaction Import FBDI to do a shipping transaction, and if you specify any quantity as backordered in your import payload, then Shipping will cancel that quantity.

Configured Items and Kits

A pick-to-order configured item or a kit is typically part of a shipment set. If you cancel a backorder for the item or kit, then Order Management will cancel only the entire item or kit. It won't cancel child components in the item or kit.

Behavior varies depending on when you cancel.

Cancel When You	Result
Release the pick.	If: • **There's no quantity on any child line.** Shipping cancels all the lines. **Quantity is available on some child lines but not all of them.** Shipping picks the lines that are available and place the lines that aren't available on backorder.
Confirm the shipment.	Shipping ships the lines that it can fulfill, and backorders the remaining lines, leading to one or more remnant lines. For more about remnants, see *Fulfill Configured Items*.

Import Your Data

You can set the Cancel Backorders attribute and the Enforce Single Shipment attribute in the Order Management work area. You can also import them through an order management extension, the Sales Orders for Order Hub Requests REST API, or file-based data import (FBDI).

You can't use a SOAP web service with this feature. If you import a web service payload, and if that payload sets the value for the Cancel Backorders attribute or the Enforce Single Shipment attribute, then Order Management will automatically set the value for those attributes to N. If you must import, don't use a web service. Instead, use an order management extension, REST API, or FBDI.

Use an Order Management Extension

Use the On Save or the On Start of Submit extension point. As an option, you can apply a condition. Assume you want to apply this feature only with purchase orders that have a Cancel_Backorders prefix:

```
def poNumber = header.getAttribute("CustomerPONumber" );
if (poNumber == null) return;
if (!poNumber.startsWith("Cancel_Backorders")) return;def lines = header.getAttribute("Lines");
while(lines.hasNext()){
def line = lines.next(); line.setAttribute("CancelBackordersFlag", "Y");
 line.setAttribute("EnforceSingleShipmentFlag", "Y");}
```

Assume you import sales orders that have these purchase orders:

- Cancel_Backorders-001
- Cancel_Backorders-002
- 003

Order Management will use this feature with the sales orders that have the 001 and 002 purchase orders, but not the 003 purchase order.

Use REST API

Assume you need to cancel back orders but don't want to enforce single shipments. Set these values:

- Set CancelBackordersFlag to true.
- Set EnforceSingleShipmentFlag to false.

For example, use this payload with the Sales Orders for Order Hub Requests REST API:

```
{
 "SourceTransactionNumber": "Cancel_Backorders-001",
 "SourceTransactionSystem": "GPR",
 "SourceTransactionId": "Cancel_Backorders-001",
 "BusinessUnitId": 204,
```

```
"BuyingPartyId": 1006,
"BuyingPartyContactId": 2663,
"TransactionalCurrencyName": "US Dollar",
"RequestedShipDate": "2024-01-14",
"RequestingBusinessUnitId": 204,
"RequestingLegalEntityId": 204,
"FreezePriceFlag": "N",
"FreezeTaxFlag": "N",
"FreezeShippingChargeFlag": "N",
"RequestedFulfillmentOrganizationCode": "M1",
"SubmittedFlag": true,
"PackingInstructions": "DOO_OrderFulfillmentGenericProcess",
"CustomerPONumber": "Cancel_Backorders-001",
"billToCustomer": [
{
"CustomerAccountId": 1006,
"SiteUseId": 1025,
"ContactFirstName": "Charles",
"ContactLastName": "Baker"
}
],
"shipToCustomer": [
{
"PartyId": 1006,
"SiteId": 1036
}
],
"lines": [
{
"SourceTransactionLineId": "1",
"SourceTransactionLineNumber": "1",
"SourceScheduleNumber": "1",
"SourceTransactionScheduleId": "1",
"ProductNumber": "AS54888",
"OrderedUOMCode": "Ea",
"OrderedQuantity": 10,
"CancelBackordersFlag": true,
"EnforceSingleShipmentFlag": false
}
]
}
```


Use File Based Data Import

Set these attributes to Y or to N on the DOO_ORDER_LINES_ALL_INT worksheet:

- Cancel Backorders. Allow the user to cancel order lines that have items that are on back order.
- Enforce Single Shipment. Allow the user to cancel the part of the order line's quantity that they can't ship in one shipment. If you set Enforce Single Shipment to Y, then you must also set Cancel Backorders to Y.

Note

- Use the SourceSalesOrderImportTemplate.xlsm template.
- You must import FBDI through REST API. For details, see *Use FBDI and REST API to Import Large Volumes of Sales Orders*.

Troubleshoot

You might encounter an error when Shipping can't process a fulfilment line because of a timing issue:

```
The fulfillment line couldn't be updated because the line isn't ready to process the response from the
 fulfillment system. Details: A wide range of issues might cause this problem, such as the line isn't on a
 wait step, the line is locked, there's a problem with the data, and so on. To fix it, go to the Scheduled
```

```
Processes work area and run the Process Responses from Order Fulfillment scheduled process. Set the Task
Type parameter to the Shipment task type, and set the From Load Request and the To Load Request parameters
to the 100104459930247 identifier when you run it.
```

You might also encounter a problem when Shipping attempts to fulfill more than one split cancel for a sales order. You might see more than one instance of the Process Response from Fulfillment System scheduled processing running at the same time, and each one with a different load request. As a result, you might have order lines that go into in error in Order Management. To fix this problem, wait for all the current instances of Process Responses from Order Fulfillment to finish running, then manually run Process Responses from Order Fulfillment Post. Set these parameters when you run it:

Attribute	Value
Task Type	Shipment
Record Set	Process Records That Failed
Incremental	Yes

For more, see *Process Responses from Order Fulfillment*.

I Don't See the Cancel Backorders Reason

If you don't see the Cancel Backorders reason after you cancel the backorder, then you might need to collect data. Run this SQL:

```
SELECT * FROM fusion.MSC_XREF_MAPPING WHERE entity_name = 'RETURN_REASON' AND attribute_name = 'LOOKUP_CODE'
AND source_value = 'ORA_CANCEL_BACKORDERS';
```

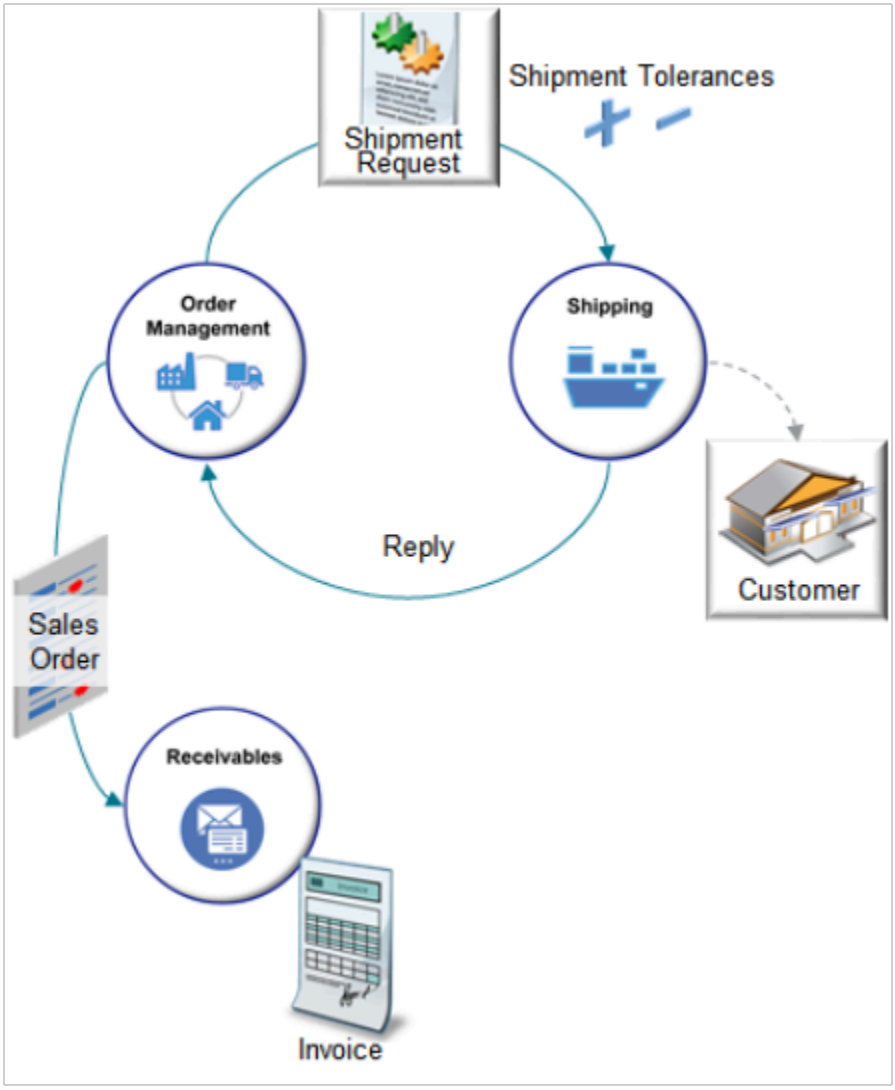
If it doesn't return the Cancel Backorders reason, then collect the Order Orchestration Reference Objects entity. See *Collect Planning Data for Order Management*.

Shipment Tolerances

Guidelines for Setting Up Shipment Tolerances

Use these guidelines to help you prepare for and set up shipment tolerances.

How it Works



1. Order Management creates and sends a shipment request to Oracle Shipping. The request includes shipment tolerances.
2. Shipping enforces the tolerances, ships the line, then sends a reply to Order Management.
3. Order Management evaluates the fulfillment line to determine whether to proceed to the next orchestration process step, or to split the line.
4. Order Management sends the sales order to Receivables.
5. Receivables invoices the transaction.

What Shipping Does

- Shipping uses each shipment advice to indicate whether it completely fulfilled the order line, or whether ship releases are still pending for the order line.
- If the shipment advice indicates that pending releases remain open for the order line, then Order Management splits the line so it can fulfill the remaining open quantity.
- If shipping ships the order line below the minimum tolerance, then it doesn't allow the Shipping Manager to close the order line. The Order Entry Specialist must cancel the line in the Order Management work area.
- Shipping sends the ordered quantity that it actually shipped to Order Management.

- Shipping confirms the shipment, sets the FinalShipmentFlag attribute to Y, then sends the value to Order Management.

What Shipping Doesn't Do

- Doesn't ship quantities above the maximum tolerance or below the minimum tolerance.
- Doesn't cancel a pending release that requires cancellation when fulfillment doesn't need another shipment to finish the order. The Order Entry Specialist must cancel the line in the Order Management work area.
- Doesn't allow the Shipping Manager to ship more than what the maximum tolerance allows.

What Receivables Does

Receivables invoices the transaction according to the quantity that Order Management sends.

- **Undershipment**. Order Management sends the actual, fulfilled quantity to receivables.
- **Overshipment**. Order Management sends the ordered quantity or fulfilled quantity depending on the value of the Quantity to Invoice for Overshipment parameter. For details, see *Manage Order Management Parameters*.

Points to Consider

- You can add tolerances to an order line on the Create Order page in the Order Management work area, through File-Based Data Import (FBDI), Electronic Data Interchange (EDI), a web service, or REST API.
- If your tax authority requires you to create a tax invoice before you ship the sales order, then consider how setting the Quantity to Invoice for Overshipment parameter affects your tax compliance requirements.
- If you decrease the order quantity, or if you change the tolerance behavior in a way that results in the shipped quantity meeting the tolerance behavior, then you must create another shipment to close the fulfillment line.
- Implementing tolerances doesn't affect your integration with Oracle Transportation Management or Oracle Global Trade Management. If a problem happens with one of these integration, you might need to manually troubleshoot it.

Revising Sales Orders

- You can't manually change shipment tolerances on an order line that already exists even if you change other attributes on the line when you revise a sales order.
- Order Management doesn't update the revised line according to changes you might have made to settings on Order Management parameters or in Product Information Management. But if you add a new line during a revision, then Order Management does use your latest settings.
- You can't use an order management extension or transformation rule to change the tolerance on a line that already exists. However, you can use an extensible flexfield and an extension to manually change the shipment tolerance.
- If you apply a tolerance, then you can't reprice a sales order after you submit it to fulfillment.

Back-to-Back Flows

Order Management sets tolerances on the order line during a back-to-back flow in the same way it sets them for a regular order line. Shipping fulfills an overshipment or undershipment during back-to-back in the same way it fulfills a regular order line.

Limitations

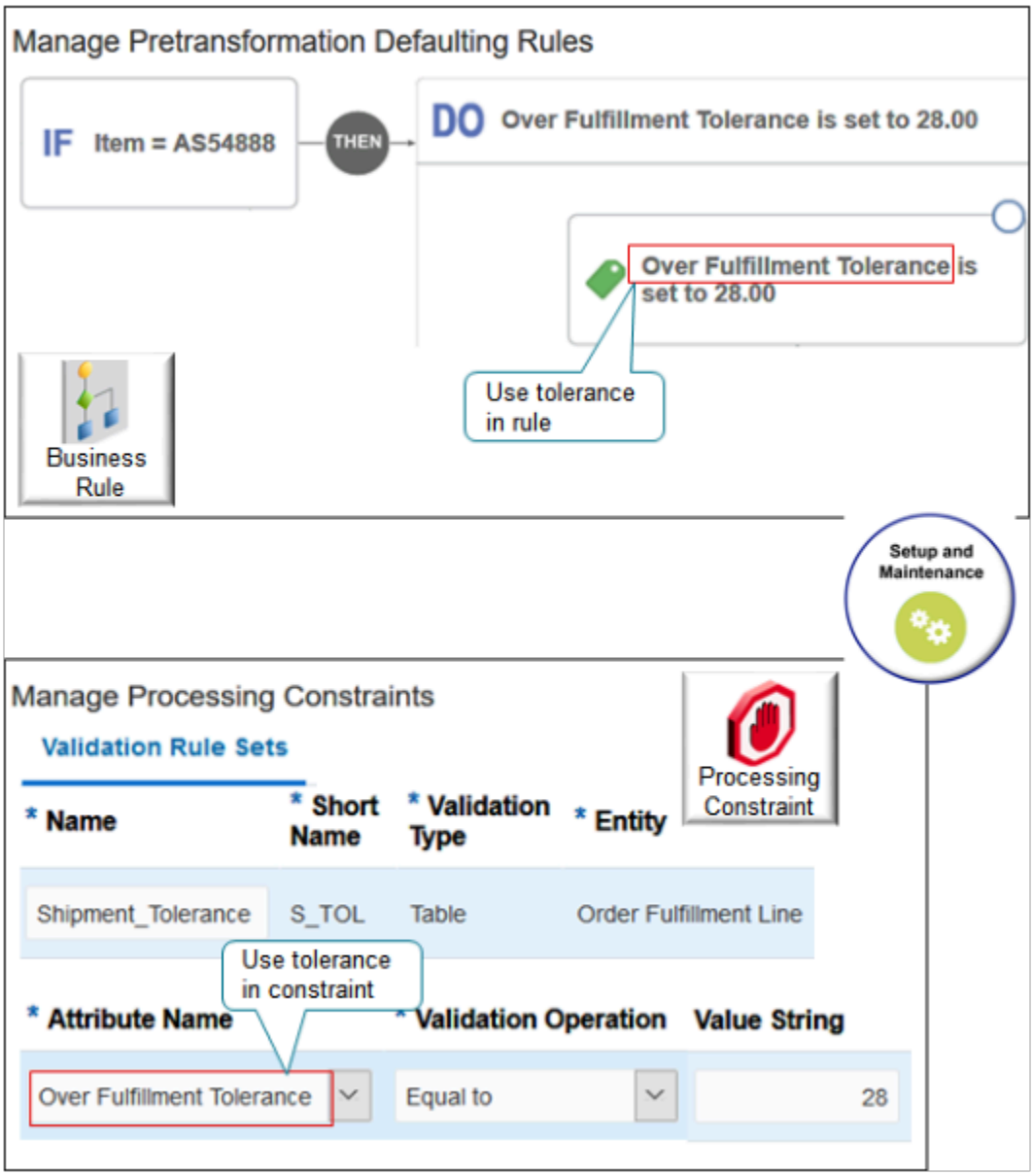
You can't apply a tolerance.

- On a configured item, assemble-to-order item, pick-to-order item, or kit.
- On an item that you can't ship, such as a service or warranty.
- On a drop ship, transfer order, or return order. You can set up Order Management to add tolerances to sales order in these flows, but downstream applications, such as receiving or shipping, won't recognize or use your tolerances.
- In the Order Management work area. You can specify tolerance only during set up, such as in Product Information Management, on the Manage Order Management Parameters page, through an order management extension, or a transformation rule.
- When you reprice the sales order in the Order Management work area.
- After you submit the sales order to order fulfillment. For example, after the user clicks Submit on the sales order.
- Through order import, such as file-based data import, or through an inbound web service. However, if you set the Source for Shipment Tolerance Values parameter to a value that isn't None, and if you create a pretransformation rule, then Order Management uses values that the parameter specifies for each order line that the import or web service creates. Order Management does this before it runs your pretransformation rule.

You can't opt out of the Manage Shipment Tolerances for Sales Orders feature after you submit a sales order that includes a shipment tolerance value that isn't empty or zero. At this point, you're committed to using the feature. You also can't revert application behavior at this point.

Transformation Rules and Constraints

Use tolerances in a pretransformation rule, posttransformation rule, or processing constraint. Use a tolerance value in a business rule or order management extension to create a condition and specify more detailed behavior than what you can set with Order Management parameters or in Product Information Management.



For example, add a business rule.

```
If the item is AS54888, then set Over Fulfillment Tolerance to 28.
```

Or add a processing constraint.

```
If the Over Fulfllment Tolerance attribute on the fulfillment line is equal to 28, then prevent the user from modifying the line.
```

Set Up Shipping Tolerances in Order Management

Set up shipment tolerances so you can use them in sales orders in Order Management.

Summary of the Setup

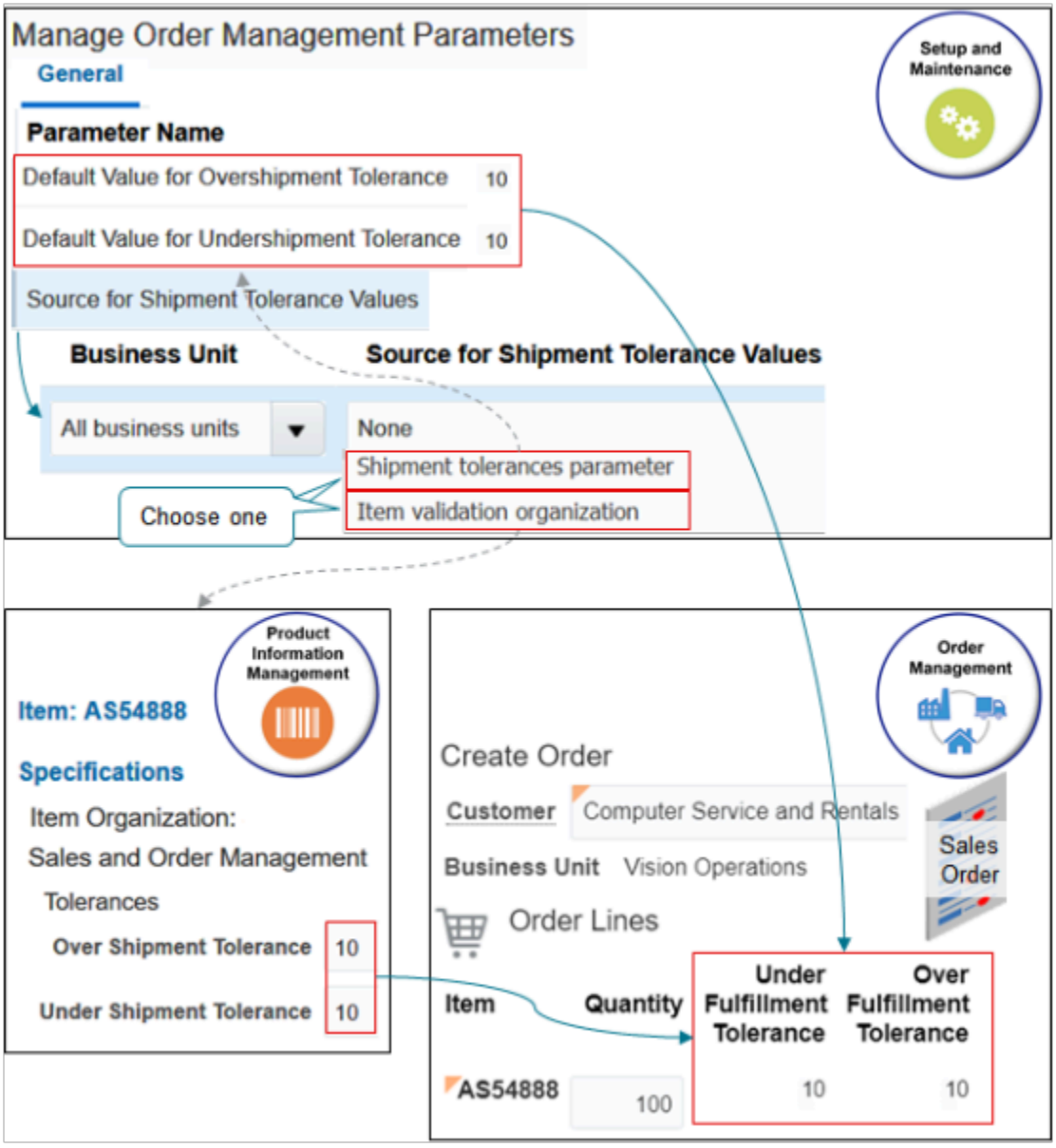
1. Enable the opt in.

2. Set up the order management parameters.
3. Set up tolerances in Product Information Management.
4. Troubleshoot your setup.

Enable the Opt In

1. Go to the Setup and Maintenance work area.
2. On the Setup page, select the Order Management offering, then click **Change Feature Opt In**.
3. On the Opt In page, in the Order Management row, click **Features**.
4. On the Edit Features page, in the Manage Shipment Tolerances for Sales Orders row, add a check mark to the **Enable** option.

Set Up the Order Management Parameters



Here's how it works.

Parameter	Description
Default Value for Overshipment Tolerance	Set the default value that you want Order Management to use as a percent for overshipment tolerance on the order line. • You must enter a positive, numeric, value. • You can enter a value with up to three decimal places, such as 10.463. • If you don't specify a value, then Order Management uses 0, by default. • If you use a defaulting rule or order management extension to set the value, then the rule or extension overrides the value you set for the parameter.
Default Value for Undershipment Tolerance	Set the default value that you want Order Management to use as a percent for undershipment tolerance on the order line. Use the same formatting requirements described for Default Value for Overshipment Tolerance, except don't exceed the value 99 for Default Value for Undershipment Tolerance.
Quantity to Invoice for Overshipment	Set the quantity to invoice when fulfillment ships more than the order line quantity. • **Shipped quantity**. Invoice the actual shipped quantity. This is the default value. • **Ordered quantity**. Invoice the order line quantity. Assume the ordered quantity on an order line is 100 tons of steel with each ton priced at $800. You over ship 103 tons for the order line. If you set Quantity to Invoice for Overshipment to. • **Shipped quantity**. The invoice will contain 103 tons with a total line value of $82,400. • **Ordered quantity**. The invoice will contain 100 tons with a total line value of $80,000. Note • If shipped quantity is equal to or less than ordered quantity, then Order Management bills for shipped quantity. If you under ship, then Order Management bills for shipped quantity and you can't bill for ordered quantity. • If Order Management splits the line into more than one line, or if shipping splits the line into more than one shipment, then Order Management splits the original fulfillment line. It also evaluates whether the sum of the shipped quantity across split fulfillment lines exceeds the sum of the ordered quantity across the same set of split fulfillment lines when it sends each split fulfillment line to Receivables. Its possible that Receivables will only recognize that the last split fulfillment line over shipped. • A change that you make to this parameter takes effect only on fulfillment lines that Order Management hasn't sent to Receivables when you save your change. Receivables will use the old parameter value for fulfillment lines that Order Management already sent to Receivables. **Caution** Changing this parameter affects downstream applications, such as Receivables and Costing. Make sure you thoroughly test you changes and determine how that affect your downstream applications in a test environment before you implement the change in your production environment.
Source for Shipment Tolerance Values	Set the source that provides default values for shipment tolerances.

Parameter	Description

	• **Shipment tolerance parameter**. Use the values that you set in these parameters.
	  ○ Default Value for Overshipment Tolerance
	  ○ Default Value for Undershipment Tolerance
	  For details, see *Manage Order Management Parameters*.
	• **Item validation organization**. Use the values you set in the Product Information Management work area.
	  If you specify item validation organization, and if you don't specify tolerances for the item in Product Information Management, then Order Management uses the values you specify for the Default Value for Overshipment Tolerance and Default Value for Undershipment Tolerance parameters.
	• **None**. Set the default values for shipment tolerances to zero.
	Any change you make to this parameter takes effect only on new sales orders that you create and submit after you save your changes to the parameter.

Setting the Business Unit

You can set the values differently for each business unit on each parameter. Click **Add Row**, then set the Business Unit attribute. For example, when you set the Source for Shipment Tolerance Values, to set the source only for sales orders that reference the Vision China business unit, click **Add Row**, then set the Business Unit attribute to Vision China in the new row.

Here's an example that references different business units.

Business Unit	Source for Shipment Tolerance Values
Vision China	Item validation organization If the business unit on the sales order is Vision China, then use your set up in Product Information Management.
Vision Japan	Shipment tolerances parameter If the business unit on the sales order is Vision Japan, then use your set up on the parameters. • Default Value for Overshipment Tolerance • Default Value for Undershipment Tolerance
All Business Units	None If the business unit isn't Vision China or Vision Japan, then set the default values for shipment tolerances to zero.

Set Up Tolerances in Product Information Management

If you set the Source for Shipment Tolerance Values parameter to Item Validation Organization, and if you need tolerances that are different from the values you set in the Default Value for Overshipment Tolerance and Default Value

for Undershipment Tolerance parameters, then you must set up tolerances in Product Information Management. For details, see *Manage Order Management Parameters*.

In this example, set up an overshipment tolerance of 10%, and an undershipment tolerance of 10% for the AS54888 item.

1. Go to the Product Information Management work area.
2. On the Product Information Management page, click **Tasks > Manage Items**.

   This topic assumes you already set up the AS54888 item.
3. On the Manage Items page, search for, then open the AS54888 item for editing.
4. On the Edit Item page, click **Specifications > Sales and Order Management**.
5. In the Tolerances area, set the values.

Attribute	Value
Over Shipment Tolerance	10
Under Shipment Tolerance	10

Note

- Enter a positive, numeric, whole number.
- Don't include any other symbols, such as %.
- Order Management treats the number you enter as a percent of the quantity ordered.
- You can't specify a quantity tolerance. You can only specify a percent of the ordered quantity.
- Set the Over Shipment Tolerance attribute to a positive number that's greater than zero.
- Set the Under Shipment Tolerance to a positive number that's greater than zero but less than 100. Using a value that's greater than 99 isn't a realistic business scenario because it means no shipment happens or a negative shipment happens.
- Use these guidelines regardless of whether you set the values in Product Information Management, an order management extension, or a business rule.

Troubleshoot Your Setup

Trouble	Shoot
I don't see the Over Fulfillment Tolerance or Under Fulfillment Tolerance attributes on the order line on the sales order in the Order Management work area.	They come predefined as hidden. On the order lines tab, click View > Columns, then make sure each of them contains a check mark. If they don't display in the menu when you click View > Columns, then make sure you enable the tolerances opt-in feature.
I don't see the parameters that I use to set up tolerances, such as Default Value for Overshipment Tolerance, Quantity to Invoice for Overshipment, or Source for Shipment Tolerance Values.	Make sure you enable the tolerances opt-in feature.

Trouble	Shoot

Examples of Setting Up Shipment Tolerances

Learn about what happens when you set up shipment tolerances for create order, revise order, and split line flows.

Create Order

You set the parameter values.

Parameter	Value
Default Value for Undershipment Tolerance	5
Default Value for Overshipment Tolerance	25
Source for Shipment Tolerance Values	Item Validation Organization

You set the values for item AS54888 in Product Information Management.

Parameter	Value
Under Shipment Tolerance	10
Over Shipment Tolerance	50

You don't set any values for item AS10000 in Product Information Management.

The Order Entry Specialist creates a new sales order in the Order Management work area and adds three items. Order Management displays a separate order line for each item.

Item	Under Fulfillment Tolerance	Over Fulfillment Tolerance
AS54888	10	50
AS10000	5	25

The Order Entry Specialist clicks Submit, and Order Management creates sales order 12345.

Copy Order

Continuing from the create order example, assume you change the parameter values.

Parameter	Value
Default Value for Undershipment Tolerance	0
Default Value for Overshipment Tolerance	30
Source for Shipment Tolerance Values	Shipment Tolerance Parameter

The Order Entry Specialist clicks Actions > Copy.

Order Management creates a new sales order that contains the same lines as order 12345, gets your updated parameter values, and displays your updated tolerances on the order lines.

Item	Under Fulfillment Tolerance	Over Fulfillment Tolerance
AS54888	0	30
AS10000	0	30

You set the Source for Shipment Tolerance Values to Shipment Tolerance Parameter, so Order Management doesn't get any tolerance values from Product Information Management. It gets them from the parameters for each item.

Revise Order

Instead of copying sales order 12345, assume the Order Entry Specialist clicks Actions > Create Revision, then creates a new order line for the AS20000.

Order Management creates a new sales order that contains the same lines as order 12345, and displays your updated tolerances on the order lines.

Item	Under Fulfillment Tolerance	Over Fulfillment Tolerance
AS54888	10	50
AS10000	5	25
AS20000	0	30

Order Management doesn't revise tolerances for order lines that it copies during a revision, but it does get the latest values from the parameters for new lines that the user adds.

Assume the Order Entry Specialist changes the quantity for the AS10000. Order Management continues to use 0 and 30 for the tolerances. It doesn't update tolerances when the user changes quantity.

Split Fulfillment Lines

Continuing our example, assume the Order Entry Specialist sets the quantity on the AS54888 line to 15, then clicks Submit.

The under tolerance for the AS54888 is 10%, and the over tolerance is 50%. The Shipping Manager is the user for the shipment application. Shipping doesn't allow the Shipping Manager to ship less than 13 because 10% of 15 is 1.5. Shipping rounds up to the nearest whole number, so the calculation is 15 minus 2. Shipping doesn't allow the user to ship more than 23 because 50% of 15 is 7.5, so the calculation is 15 plus 8.

Order Management sends sales order 12345 to shipping. Shipping splits the AS54888 order line into more than one delivery, over ships the line, then closes it.

The AS54888 line is in status Awaiting Shipping. The Shipping Manager creates and ships three deliveries with quantities of 12, 2, and 1. Some time later, the Shipping Manager adds a quantity of 3 to the second delivery. Shipping fulfills the second delivery, cancels the third delivery, sets the FinalShipmentFlag attribute to Y, then sends the attribute to Order Management.

The Shipping Manager examines the customer order and notices it includes two lines for the AS54888. One line includes an ordered quantity of 7 and a shipped quantity of 12. The other line includes an ordered quantity of 3 and a shipped quantity of 3. Both lines are awaiting billing.

Here's a summary.

Delivery ID	Requested Quantity	Shipped Quantity	Status	Fulfilled	Pending Requested Quantity or Ordered Quantity	Description
1	7	12	Order Management Sent Details to Invoicing	N	3	You can't ship more than 12 because two deliveries are open and they have a total quantity of three.
2	2	3	Order Management Sent Details to Invoicing	Y	0	You met the over tolerance.
3	1	-	Canceled	-	-	-

Examples of How Shipping Handles Tolerances

Shipping sends the requested quantity in the shipment advice for each shipment or delivery.

Shipping might split a fulfillment line to help meet the requested delivery date. It uses a formula.

- The ordered quantity on the original fulfillment line equals the requested quantity on the shipment.
- Shipping splits the fulfillment line.
- The ordered quantity on the new, split fulfillment line equals the total ordered quantity minus the requested quantity on the shipment.

- When the ordered quantity on the fulfillment line equals the requested quantity on the shipment, there are no more splits.

Under Shipment in One Shipment

Assume Order Management sends a fulfillment line to shipping.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	10	0%	20%	Empty	Awaiting Shipping

A truck load limit or transportation planning recommendation sometimes requires a split. There's only a quantity of 8 on hand, so the shipping clerk manually splits the shipment, creates shipment S1, splits the fulfillment line into two shipment lines, and ships SL1. Shipping receives confirmation that the customer received SL1, so it cancels SL2 because the quantity of 8 on SL1 falls within the under tolerance.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status
S1	SL1	8	8	Shipped
-	SL2	2	0	Canceled

where

- Ordered quantity is 10.
- Under tolerance is 20%.
- Under quantity is 10 multiplied by 20% equals 2.
- Minimum quantity is 10 minus 2 equals 8.

Shipping sends a ship confirmation to Order Management.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	10	0%	20%	8	Shipped or Awaiting Billing

What if only 5 are on hand? Shipping ships 5 and puts 5 more on backorder.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Pending Shipment Quantity	Status
S1	SL1	5	5	5	Shipped
-	SL2	5	-	-	Backordered

If shipping can't fulfill SL2 before the customer receives SL1, then it splits the fulfillment line, places the 5 it hasn't shipped on FL2, and sends the results to Order Management.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	5	0%	20%	5	Shipped/Awaiting Billing
FL2	5	-	-	-	Backordered

Over Shipment in One Shipment

Assume Order Management sends a fulfillment line to shipping.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	10	30%	0%	Empty	Awaiting Shipping

Shipping communicates with Global Order Promising and finds it has a quantity of 12 on hand, so it creates shipment S1 and ships the entire quantity.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status
S1	SL1	10	12	Shipped

where

- Ordered quantity is 10.
- Under tolerance is 30%.
- Over quantity is 10 multiplied by 30% equals 3.
- Maximum over quantity is 10 plus 3 equals 13.
- Only 12 are available, so shipping ships what's available.

Shipping sends a ship confirmation to Order Management.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	10	30%	0%	12	Shipped or Awaiting Billing

Over Shipment In More Than One Shipment

Assume Order Management sends a fulfillment line to shipping.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	10	30%	0%	Empty	Awaiting Shipping

Shipping communicates with Global Order Promising and finds it has a quantity of 6 on hand at warehouse A, 2 more at warehouse B, and 3 more at warehouse C, so it creates three separate shipments, S1, S2, and S3, ships S1, and sends a request to release inventory to warehouses B and C.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status	Shipped On
S1	SL1	5	6	Shipped	Monday
S2	SL2	2	-	Ready to Release	-
S3	SL3	3	-	Ready to Release	-

Shipping splits the fulfillment line into FL1 and FL2.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	5	30%	0%	6	Shipped
FL2	5	30%	0%	Empty	Awaiting Shipping

Some time later, shipping ships S2 and updates the status for S2 to Shipped.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status	Shipped On
S1	SL1	5	6	Shipped	Monday
S2	SL2	2	2	Shipped	Tuesday
S3	SL3	3	-	Ready to Release	-

Shipping splits FL2 into FL2 and FL3, and sets the status for FL3 to Awaiting Shipping.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	5	30%	0%	6	Shipped
FL2	2	30%	0%	2	Shipped

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL3	3	30%	0%	Empty	Awaiting Shipping

Some time later, shipping ships S3 and updates its status to Shipped, but S3 could only ship a quantity of 1, so shipping creates another shipment, S4.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status	Shipped On
S1	SL1	5	6	Shipped	Monday
S2	SL2	2	2	Shipped	Tuesday
S3	SL3	1	1	Shipped	Wednesday
S4	SL4	2	-	Ready to Release or Backordered	-

To track the backorder, shipping creates FL4.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	5	30%	0%	6	Shipped
FL2	2	30%	0%	2	Shipped
FL3	1	30%	0%	1	Shipped
FL4	2	30%	0%	-	Backordered

Some time later, shipping ships S4.

Shipment	Shipment Line	Requested Quantity	Shipped Quantity	Status	Shipped On
S1	SL1	5	6	Shipped	Monday
S2	SL2	2	2	Shipped	Tuesday
S3	SL3	1	1	Shipped	Wednesday
S4	SL4	2	3	Shipped	Thursday

Shipping sends an update to Order Management.

Fulfillment Line	Ordered Quantity	Over Tolerance	Under Tolerance	Shipped Quantity	Status
FL1	5	30%	0%	6	Shipped
FL2	2	30%	0%	2	Shipped
FL3	1	30%	0%	1	Shipped
FL4	2	30%	0%	3	Shipped

The total shipped quantity for the fulfilled sales order is 12, which is below the maximum over quantity of 13.

Manage Delay

Jeopardy Priority

Jeopardy indicates the severity of a delay of a task in an orchestration process. You can modify the predefined range of jeopardy scores for a jeopardy priority to control how Order Management calculates and displays jeopardy.

For example, you can set a minimum score of 0, and a maximum score of 100 for the Low jeopardy priority.

Note

- You can use jeopardy on planned dates or on actual dates. This set up allows an order manager to become aware that a jeopardy condition might exist before it actually occurs, and to take action to fix the condition and reduce jeopardy.
- You can't add or delete a jeopardy priority.
- You can't modify the value of a predefined jeopardy priority, such as Low.

Learn about jeopardy and how it works. For details, see *Jeopardy Score*.

Related Topics

- Jeopardy Threshold
- Jeopardy Score

Jeopardy Threshold

Set up a jeopardy threshold to measure and monitor an orchestration process.

Set up a set of ranges for each fulfillment task of an orchestration process, then assign a score that indicates the severity of the delay. Order Management uses your set up in the jeopardy attributes that it displays in the Order Management work area. These attributes help your users to quickly determine the severity of the delay, then take appropriate action.

You can set up a jeopardy threshold for any combination of these attributes.

- Task type
- Task name
- Process name
- Process version

If you leave these attributes at their default setting of All, then Order Management applies jeopardy threshold to all fulfillment tasks.

If you apply a jeopardy threshold, then you must first set up the orchestration process, fulfillment task, and task type that the threshold will reference so you can select these objects when you apply the threshold.

Related Topics

- Jeopardy Priority
- Set Up Jeopardy and Lead Time to Manage Delay
- Jeopardy Score

Set Up Jeopardy and Lead Time to Manage Delay

To manage delay, set up jeopardy priority, lead time, and jeopardy threshold for your orchestration process.

Summary of the Steps

1. Manage jeopardy priority.
2. Specify lead time.
3. Manage jeopardy threshold.

This topic describes how to do these steps because you often do them together. However, you can do them independently of one another.

This topic uses example values. You might need different values, depending on your business requirements.

Manage Jeopardy Priority

Assume you need a higher range for the MEDIUM priority, and a higher but more narrow range for the HIGH priority.

1. Enable the Enable Orchestration Process Planning and Calculate Jeopardy parameter.

   For details, see *Manage Order Management Parameters*.
2. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Jeopardy Priorities

3. On the Manage Jeopardy Priorities page, set the values, then click **Save and Close**.

Priority	Minimum Score	Maximum Score
LOW	10	199
MEDIUM	200	399
HIGH	400	600

Don't.

- Overlap scores across priorities. For example, don't set up a Low priority of 10 to 210, and a Medium priority of 200 to 400. An overlap exists between 200 and 210.
- Set up priorities that intersect at the same value. If you set up two priorities that intersect at the same value, then Order Management assigns the intersecting value to the lower priority.

  For example, if you set up Low priority of 10 to 200, and Medium priority of 200 to 400, then Order Management considers a score of 200 as Low priority.

Specify Lead Time

Assume you need a specific amount of lead time to finish each orchestration process step in an orchestration process that ships goods. Learn about lead times. For details, see *Guidelines for Setting Up Orchestration Process Steps*.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, search for an orchestration process, then open it.

   For this example, search for Orchestration_Process_1. Learn how to create this process. For details, see *Pause Orchestration Processes Until Events Happen*.
3. On the Step Definition tab, set the Default Lead Time attribute, and set the Lead Time UOM attribute to Days. Set them for each step.

Step	Default Lead Time
Schedule Goods	2
Reserve Goods step	1
Ship Goods	4

Step	Default Lead Time
Wait for Ship Goods	6
Prepare Documentation	1
Wait for Documentation	1
Invoice Goods	3
Wait for Invoice Goods	2

4. Click **Save and Close**.

Manage Jeopardy Threshold

Assume you can use a threshold on reservation tasks that your orchestration process runs to manage how long the process can delay each task.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Jeopardy Thresholds
2. On the Manage Jeopardy Threshold Definitions page, click **Actions > Create**.
3. On the Create Jeopardy Threshold Definition page, set the values.

Attribute	Value
Code	Enter any text that Order Management can use as an abbreviation for the threshold. Order Management uses this text as a code to identify the threshold throughout the Order Management work area, such as in lists. For this example, enter **`t-shirt_threshold`**.
Name	Clothing Jeopardy
Description	Jeopardy thresholds for company t-shirts.
Process Name	Select the name of the orchestration process where Order Management must apply the jeopardy threshold. For this example, select Orchestration_Process_1.

Attribute	Value
Task Name	Reserve
Task Type	Reservation

4. Set the threshold for the Low range of the jeopardy priority. Click **Actions > Add Row**, then set the values.

Attribute	Value
Maximum Delay	2
Maximum UOM	Days
Jeopardy Score	Enter a value that resides in the Low range of the jeopardy priority that you modified above, such as 100.
Description	Threshold for the Low range of the jeopardy priority.

5. Set the threshold for range Medium of the jeopardy priority. Click **Actions > Add Row**, then set the values.

Attribute	Value
Maximum Delay	5
Maximum UOM	Days
Jeopardy Score	Enter a value that resides in the Medium range of the jeopardy priority that you modified above, such as 300.
Description	Threshold for the Medium range of the jeopardy priority.

6. Set the threshold for the High range of the jeopardy priority. Click **Actions > Add Row**, then set the values.

Attribute	Value
Maximum Delay	8

Attribute	Value
Maximum UOM	Days
Jeopardy Score	Enter a value that resides in the High range of the jeopardy priority that you modified above, such as 500.
Description	Threshold for the High range of the jeopardy priority.

7. Click **Save and Close**.

Related Topics

- Jeopardy Priority
- Jeopardy Threshold
- Overview of Orchestration Processes
- Use Case to Pause for an Event
- Guidelines for Setting Up Orchestration Process Steps

Change Orders

Overview

Overview of Managing Change That Occurs During Order Fulfillment

Order Management comes predefined to process change, but you can modify the set up to meet your needs.

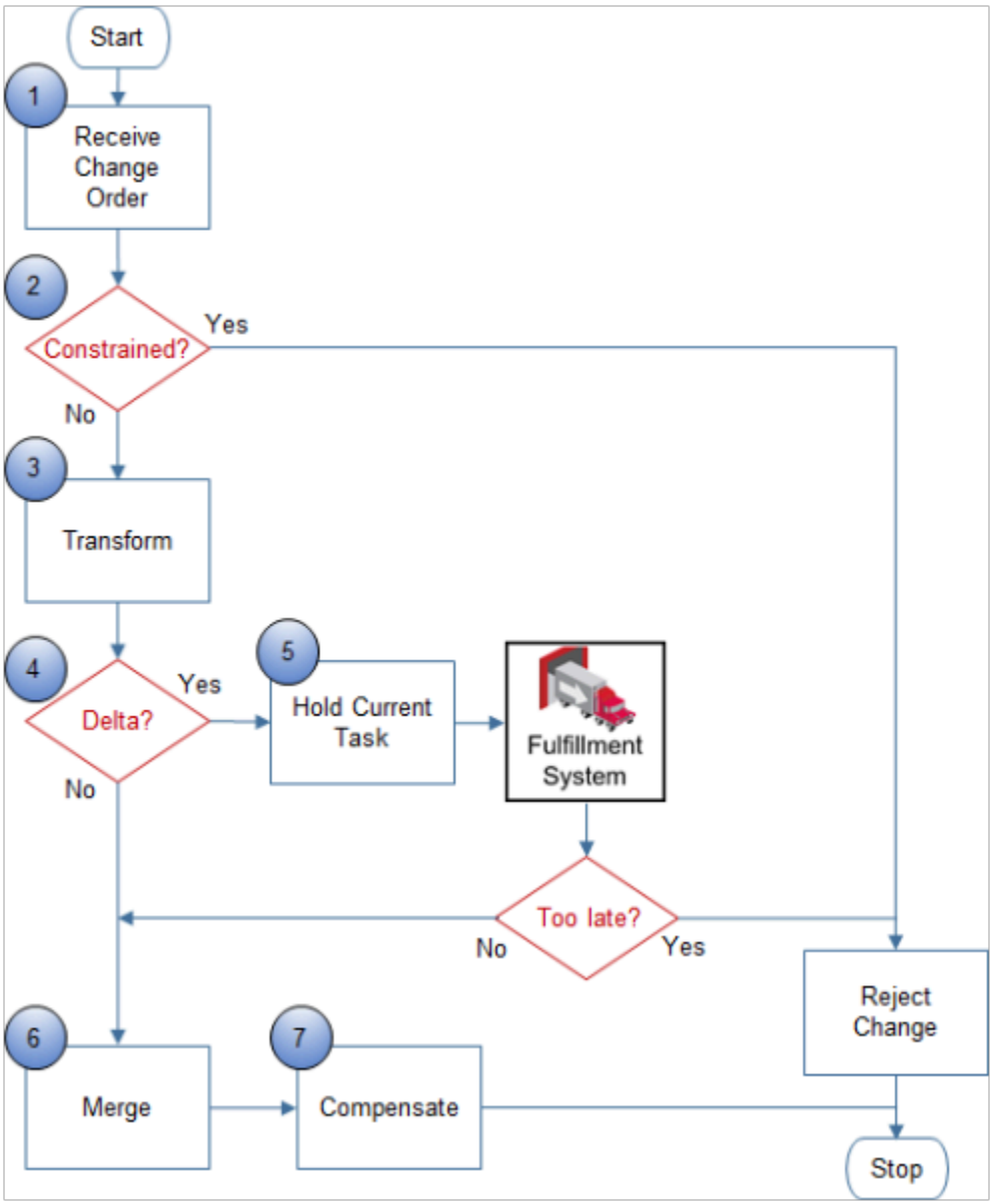
A change order is a change that affects a sales order during order fulfillment. It can come from a variety of sources, such as from a user through an order capture system, a user who uses the Order Management work area, or from change orders you import. For example, you might need to make changes to a sales order after you submit it to order fulfillment.

- Change the quantity and the ship-to address.
- Add more items to a sales order you created yesterday.
- Cancel a sales order or cancel an order line.
- Don't allow changes to a sales order after some point in time or after a condition occurs, such as after pick release.

Here are some examples of how change can occur.

- Order Entry Specialist uses the Create Order Revision action to revise a sales order or makes changes in fulfillment views. Order Management doesn't apply order management extensions when the user makes a change in a fulfillment view. For details, see *Revise Sales Orders That You Already Submitted*.
- You import change from a channel.

How Change Orders Work



Here's the sequence Order Management uses to process change.

1. **Receive Change Order**. Receive change order from channel.

2. **Constrained?** Apply processing constraint.

   A processing constraint is a rule that controls who can change a sales order, what can change in the sales order, and when the change can occur.

   Order Management examines the header and fulfillment lines.

Entity	Description
Order Header	Determine whether a processing constraint on header prevents change, and whether the sales order is closed. o If a constraint doesn't allow change, or if the order is closed, then reject the entire change and exit this sequence. o Evaluating order header before evaluating order lines prevents Order Management from unnecessarily processing a closed sales order. o Order Management evaluates attributes when the user creates the revision, and also when the user submits the revision to make sure other fulfillment changes didn't occur that might affect the order.
Fulfillment Line	Determine whether constraints on the fulfillment line allow change. o If a constraint doesn't allow change on any fulfillment line, then reject the entire change and exit this sequence. o Some constraints on a fulfillment line don't allow some changes, by default, such as updating an order line if fulfillment line status is Shipped.

3. **Transform**. Transform the change order.

   Order Management does the transformations you set up, such as transforming an attribute value in the source order to a value that your fulfillment system can understand. For details, see *Transformation Rules*.

   If you create an order management extension that affects a change order, then Order Management finishes transformation, then runs the extension.
4. **Delta?**. The delta is the difference between an attribute value in the original sales order and the new value for the attribute in the change order. For example:

Condition	Description
Quantity in original order is 1, and Quantity in revised order is 3.	A delta exists, and its value is 2.
Quantity in original order is 1, and Quantity in revised order is 1.	No delta exists.

Condition	Description

Here's how Order Management determines the delta.

- Examines set ups to determine which attributes to examine.
  - Uses values you specify on the Order Attributes That Identify Change page.
  - Examines predefined attributes that affect the task. You can't specify to examine or not examine these predefined attributes.
  - Examines attributes you add that affect the task. You can specify to examine or not examine attributes you add.
- Identifies orchestration process steps that reference these attributes.
- Analyzes the state of each step. The orchestration process records the state every time it runs a task. Order Management compares the change order to the existing sales order to determine whether the value of the attribute changed.
- Uses these state details to determine the processing needed to incorporate the change.

In this example, assume the user changed the quantity, and you specified Quantity as a change attribute, so Order Management starts compensation.

5. **Hold Current Task**. If a delta exists, then hold the task that's currently running.
   A change order requests to change a sales order that an orchestration process is already processing in order fulfillment. The orchestration process runs various tasks during order fulfillment, such as schedule, reserve, ship, or invoice.
   For example, assume the orchestration process sent a request to the shipping system to ship the item, is currently waiting for the shipping system to send a reply, and sets the status to Awaiting Shipping. The change order might affect shipping, so Order Management sends a request to the shipping system to temporarily stop processing the task.
   Stopping the task allows Order Management to finish the change and prevent the shipping system from shipping the sales order without the changes that the change order requests.
   If the fulfillment system can't accommodate the change, then it replies with a rejection, and the sequence ends. For example, if the fulfillment system already shipped the sales order, then its too late to accommodate the change. Instead, the Order Entry Specialist must create a return order to make the requested changes.
6. **Merge**. Merge the change order into the sales order that the orchestration process is currently processing.
7. Compensate.

Compensate

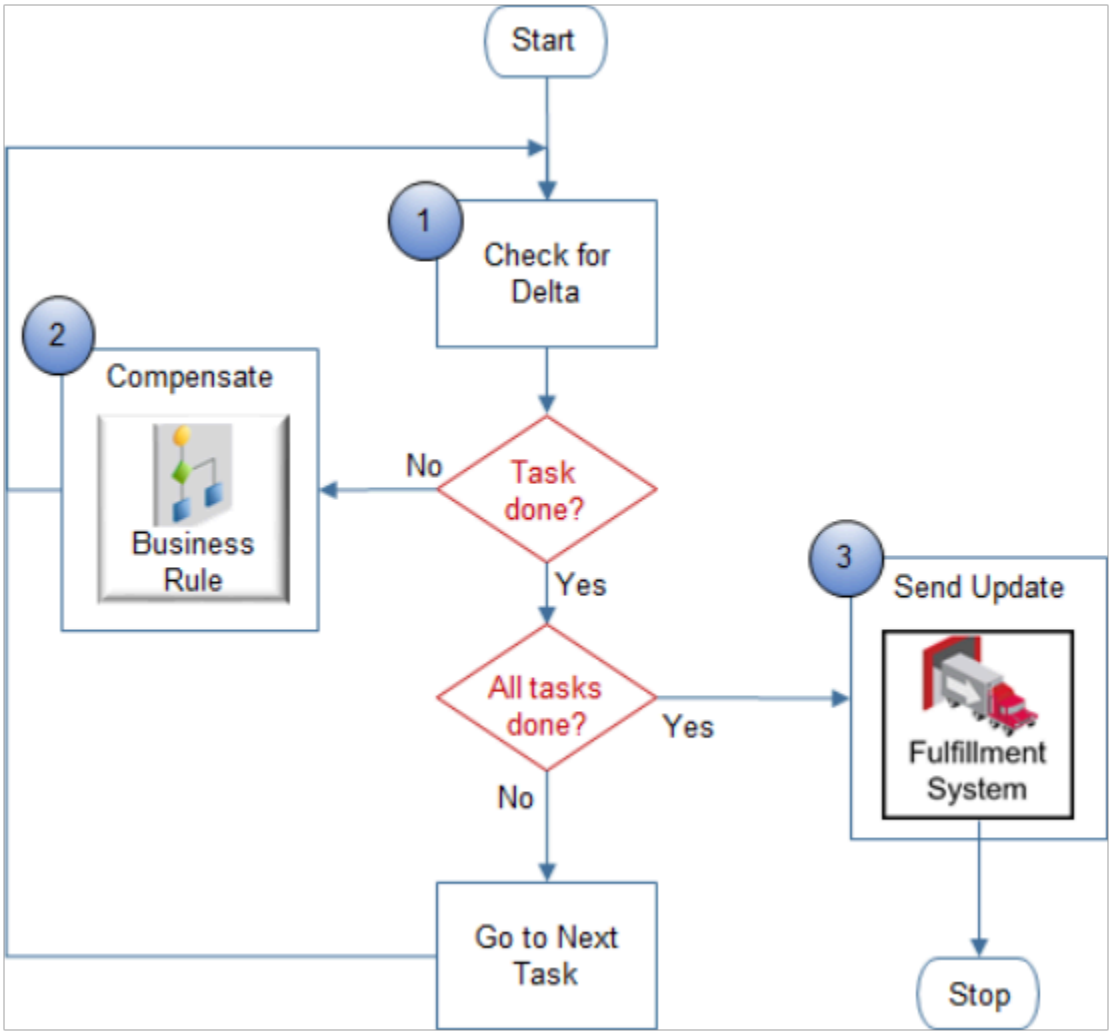
A compensation pattern is a rule that you set up on an orchestration process step that specifies adjustments to make when an order changes.

For example:

- Change order requests to use a different warehouse.
- Compensation pattern for the Create Shipment step is Redo.
- This step calls the Cancel service to cancel the current request, and the Create service to create a new request that includes the change order.

- If Order Management receives a change order that includes a new warehouse for this step, then it runs Cancel and Create again.

Here's how compensation works.



1. **Check for Delta**. Determine whether a delta exists that affects the current task. For example, if the current task is reservation, and if a delta exists for the Quantity attribute, then the delta affects the reservation task because the orchestration process must adjust the reserved quantity so it reflects the change order.

   If the delta determines that an attribute requires compensation, then the orchestration process uses the compensation pattern that the step references to compensate the step.

2. **Compensate**. Here's what Order Management does.
    - Runs the orchestration process again.
    - Sends updates to the fulfillment system for each task. If the changed attribute doesn't affect the task, then the orchestration process applies the attribute change to the sales order but doesn't send an update to the fulfillment system.

You can set up a business rule that determines the action to take according to the compensation pattern.

Most orchestration process steps don't include a compensation pattern, and they use Update, by default. In this example, Order Management compensates some steps.

Steps That Order Management Compensates	Description
Schedule	Cancels the Schedule step, then creates a new instance of this step. Order Promising determines availability. Order Management replans fulfillment, then assigns a revised date to each orchestration process step.
Create Reservation	Assume the orchestration process in this example includes a pattern for the Create Reservation step. - If Demand Class Code isn't Gold, then cancel the Create Reservation step, then create a new instance of this step. This rule instructs Order Management to release supply and create a new reservation for all customers except Gold customers. Order Management also updates Create Reservation according to the new dates, and updates the reserved quantity.
Create Shipment Request	Updates the Create Shipment Request step with new dates and new item.

The compensating services run, and then finish. These services use FIFO (first in, first out) sequence to compensate the sales order, according to the orchestration process sequence, by default. If Order Management must cancel the entire sales order, then it uses LIFO (last in, first out).

If the original orchestration process.

- **Can accommodate the change.** Order Management uses the original orchestration process to continue processing.
- **Can't accommodate the change.** Order Management cancels the original orchestration process, then starts a new one that can accommodate the change.

3. **Send Update**. Compensation finishes. Processing for the orchestration process is now at the same step it was on when Order Management received the change order.

   Order Management sends an update message to your fulfillment system to update the original message with the changed order that includes the changed attributes. For example:

Values	Description
Original Values	Quantity is 1 and Arrival Date is August 15, 2018
Values after compensation sends an Update Inventory message for the change order that includes the changed attributes.	Quantity is 2 and Arrival Date is August 18, 2018

Example of Processing Change

Assume you must set up Order Management to compensate change when the Order Entry Specialist changes quantity.

You create the ShipOrderGenericProcess orchestration process. Here are the set ups you use.

Set Up	Value
Change Mode	Advanced
Order Attributes That Identify Change	You include attributes. • Ordered Quantity • Demand Class Code • Requested Ship Date
Cost of Change	You specify a rule. If the fulfillment line status is. • **Reserved.** The cost of change is 15. • **Shipped.** The cost of change is 100. A lower number indicates a lower cost.

You add steps.

Step	Name	Task Type
1	Schedule	Schedule
2	Create Reservation	Reservation

Step	Name	Task Type
3	Create Shipment Request	Shipment
4	Wait for Shipment Advice	Shipment
5	Create Invoice	Invoice
6	Wait for Invoice	Invoice

Each step references a task type, and each task type references order attributes that Order Management uses to determine whether to compensate the sales order. In this example, the Schedule step and the Shipment step each reference a task type that references the Ordered Quantity attribute.

For example, if the user increases Quantity on the sales order, then Order Management must schedule and ship more supply.

- Assume the Order Entry Specialist submits a sales order for the AS54888 Desktop Computer with a quantity of 1.
- One day later, the Order Entry Specialist clicks Create Revision, changes Quantity to 2, then clicks Submit.

Order Management compensates the Schedule step and the Shipment step.

Related Topics

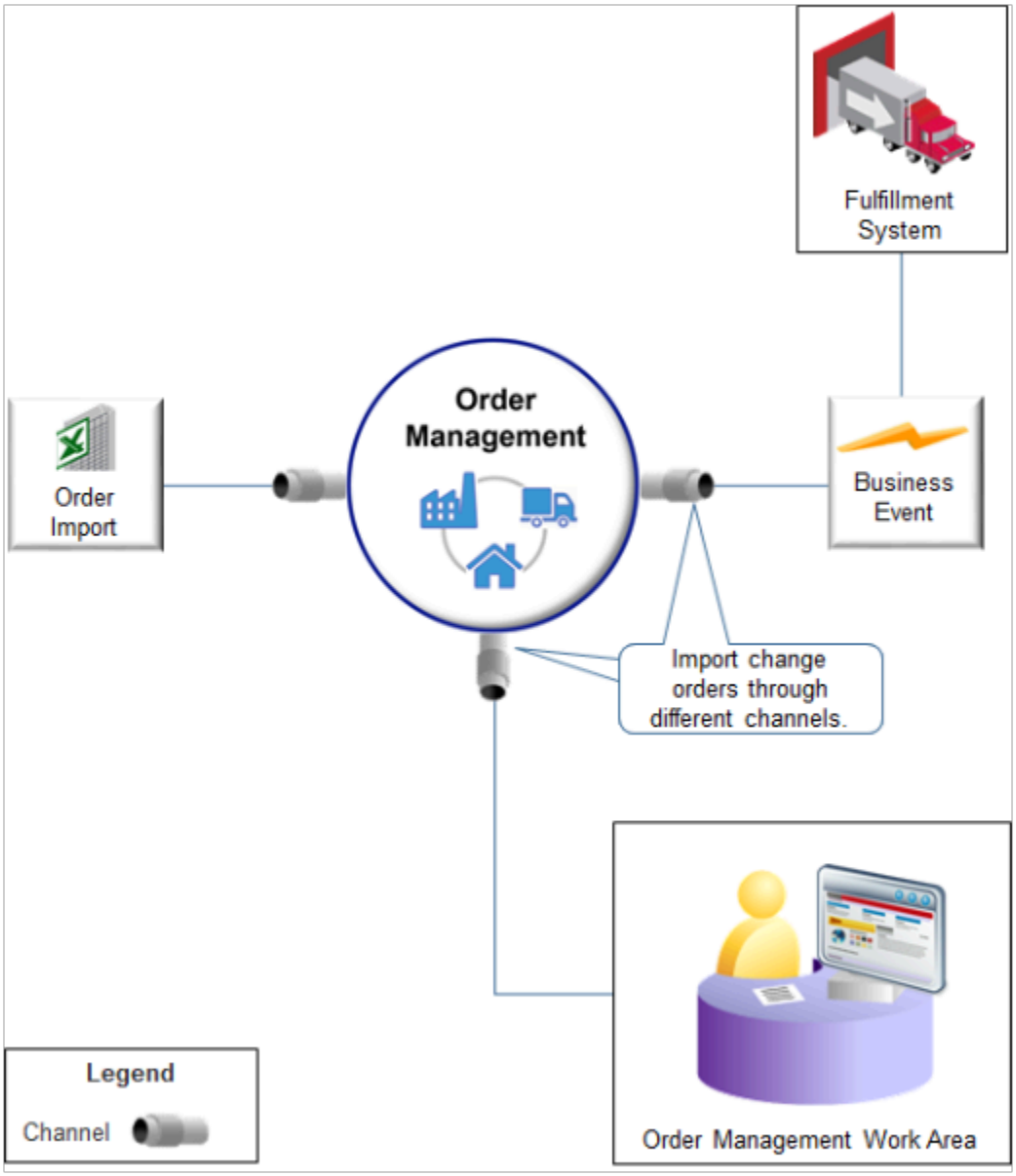
- Guidelines for Managing Change That Occurs During Order Fulfillment
- Processing Constraints
- Update Attributes on Split Order Lines for Partial Shipments

Guidelines for Managing Change That Occurs During Order Fulfillment

Apply guidelines when you specify how to process change that occurs during order fulfillment, including how to compensate change.

Import Change From Channel

Import change orders through various channels.



Note

- A channel can include an order capture system, fulfillment system, the Order Management work area, and so on.
- Order Management cross-references, transforms, validates, and orchestrates the change in a way that's similar to how it does this work for a new sales order. If rules that control how to process a change exist, then Order Management applies them. You can use setup options to modify how Order Management does this processing.
- Use the same Excel template that you use to import sales orders to import change orders. For details, see *Import Orders Into Order Management*.
- Use the same web service that you use to create a sales order to import a change order. For details, see *Use Web Services to Import Orders*.

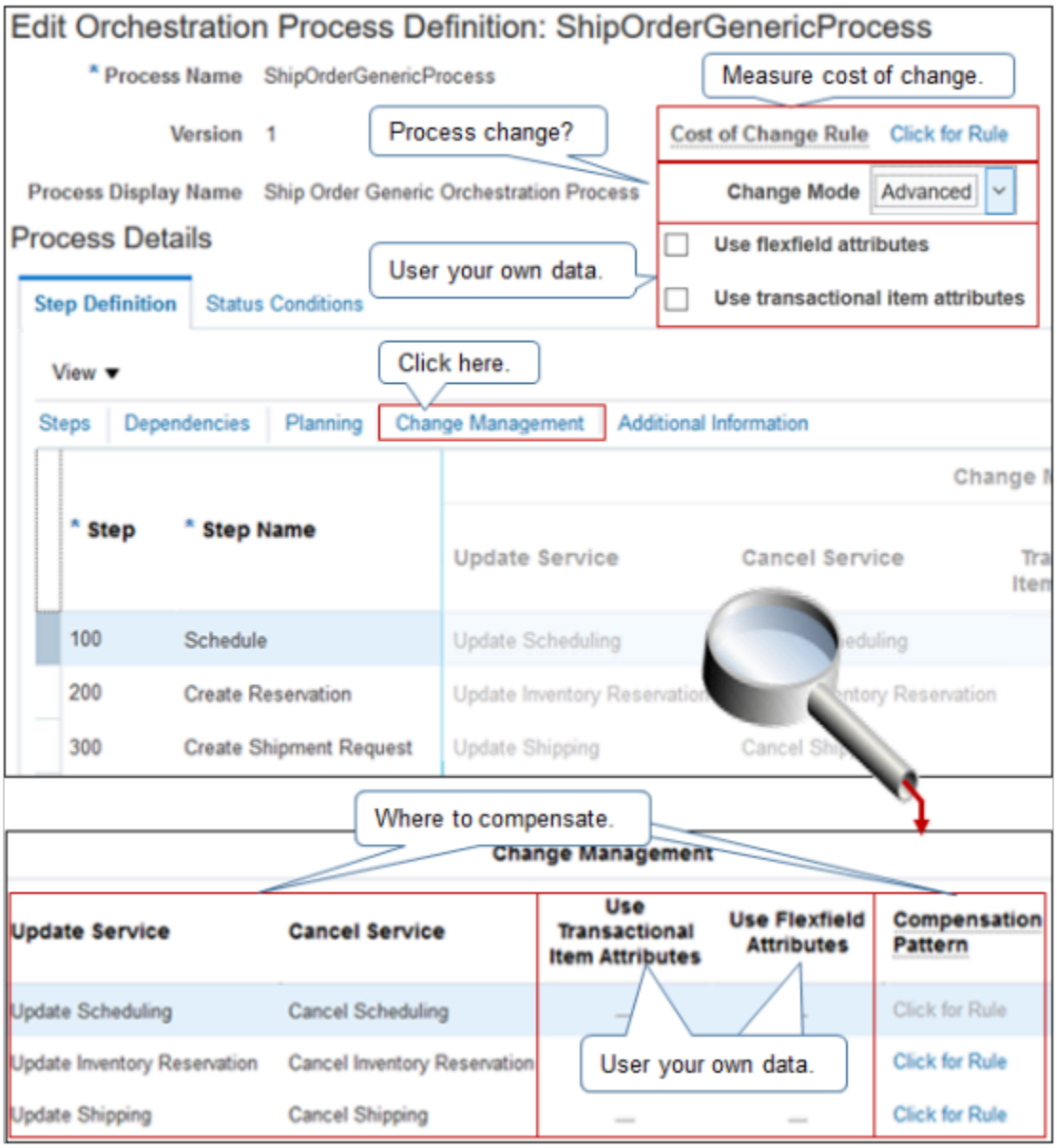
- Use only one channel to make changes. For example, use Create Order Revision to make changes, or use a web service, or use file-based import. Don't use Create Order Revision and web services and file-based import. Using different channels causes confusion when attempting to identify the source of the change.
- Some implementations price the order in a channel, then send it to Order Management.
    - You can't change pricing that your channel uses to pricing that Oracle Pricing Administration uses.
    - You can't change pricing that Oracle Pricing Administration uses to pricing that your channel uses.
- Order Management submits the change orders it receives through a web service to order fulfillment. You can't submit a change order through a web service and keep it in Draft status. Order Management only keeps a sales order in Draft if an error occurs in the change order.
- Order Management doesn't update a sales order that's in Draft status.
- If you use Create Order Revision in the Order Management work area, and must revise the order again, then use Create Order Revision again. Don't use a web service or file-based import to revise the order again.

Manage attribute values.

- If you change only one line in a sales order, then it isn't necessary to import all order lines.
- To cancel the order line, import a quantity of 0.
- Order Management sets the attribute value to empty for each order line attribute that the channel doesn't send. For example, if the original order includes a value for Shipping Method, and if the channel sends a change request that doesn't include Shipping Method, then Order Management sets Shipping Method to empty.
- If the channel revises only the order header, then Order Management revises the header but not the order lines.
- The channel must include a value for each required attribute.
- To cancel an order line, the channel must explicitly request the cancel. Order Management doesn't implicitly cancel order lines when it revises a sales order.

Set Up Orchestration Process

Use the Edit Orchestration Process Definition page in the Setup and Maintenance work area. For details, see *Set Up Orchestration Processes*.



You can administer features.

Feature	Description
Cost of Change Rule	A rule you set up on an orchestration process that specifies the business cost your organization will incur as a result of processing the change.
Change Mode	An option you set on an orchestration process. It determines when Order Management records the state of the process. It compares these states during order compensation. Choose a value. • **Advanced.** Process change. Record the state of the orchestration process at each orchestration process step.

Feature	Description
	• **Simple.** Process change only on the step that receives change. Record the state of the orchestration process when it starts and at the step where the orchestration process receives the change. • **None.** Don't allow change. Don't record the state of the orchestration process.
Use Flexfield Attributes	An option you set on an orchestration process or orchestration process step that specifies whether to examine flexfield attributes when compensating the sales order. If you enable this option, then Order Management examines flexfield attributes that the item references to determine whether it must compensate the orchestration process. For example, assume you create flexfield Color. Assume the user changes the value of Color from Red to Blue. You can use Color during compensation to allow your fulfillment system to reschedule the order line so it ships a blue item instead of a red one. Order Management disables Use Flexfield Attributes and Use Transactional Item Attributes on each predefined orchestration process, by default. You can't enable these options on a predefined orchestration process. However, you can create a copy of a predefined orchestration process, and then enable them on the copy.
Use Transactional Item Attributes	An option you set on an orchestration process or orchestration process step that specifies whether to examine transactional attributes when compensating a sales order. For details, see *Manage Transactional Attributes*. Enable the Use Flexfields Attributes option and the Use Transactional Item Attributes option only if you will send these attributes to your fulfillment system during compensation.
Use Dynamic Attributes	An option you set that specifies whether to examine dynamic attributes when compensating a sales order.
Compensation Pattern	A rule you set up on an orchestration process step that specifies the adjustment to make to the sales order when the order changes. For example, if Order Management receives a change order that specifies to ship an item from a different warehouse during the Create Shipment step, then it runs the Cancel service and the Create service again. If you don't specify a compensation pattern, then Order Management uses the predefined compensation pattern, by default. A predefined pattern uses Update, Cancel, or Create.
Task Type	An attribute you set on an orchestration process step. The task type that you select determines the attributes that Order Management uses when it determines whether it must compensate the sales order for this step. Order Management comes predefined to use a set of attributes for each predefined task type. You can't modify the predefined set up, but you can add more attributes. Order Management doesn't add these attributes to a new task type you add, and it doesn't evaluate this task until you add these attributes. To add them, you must click **Add All**.

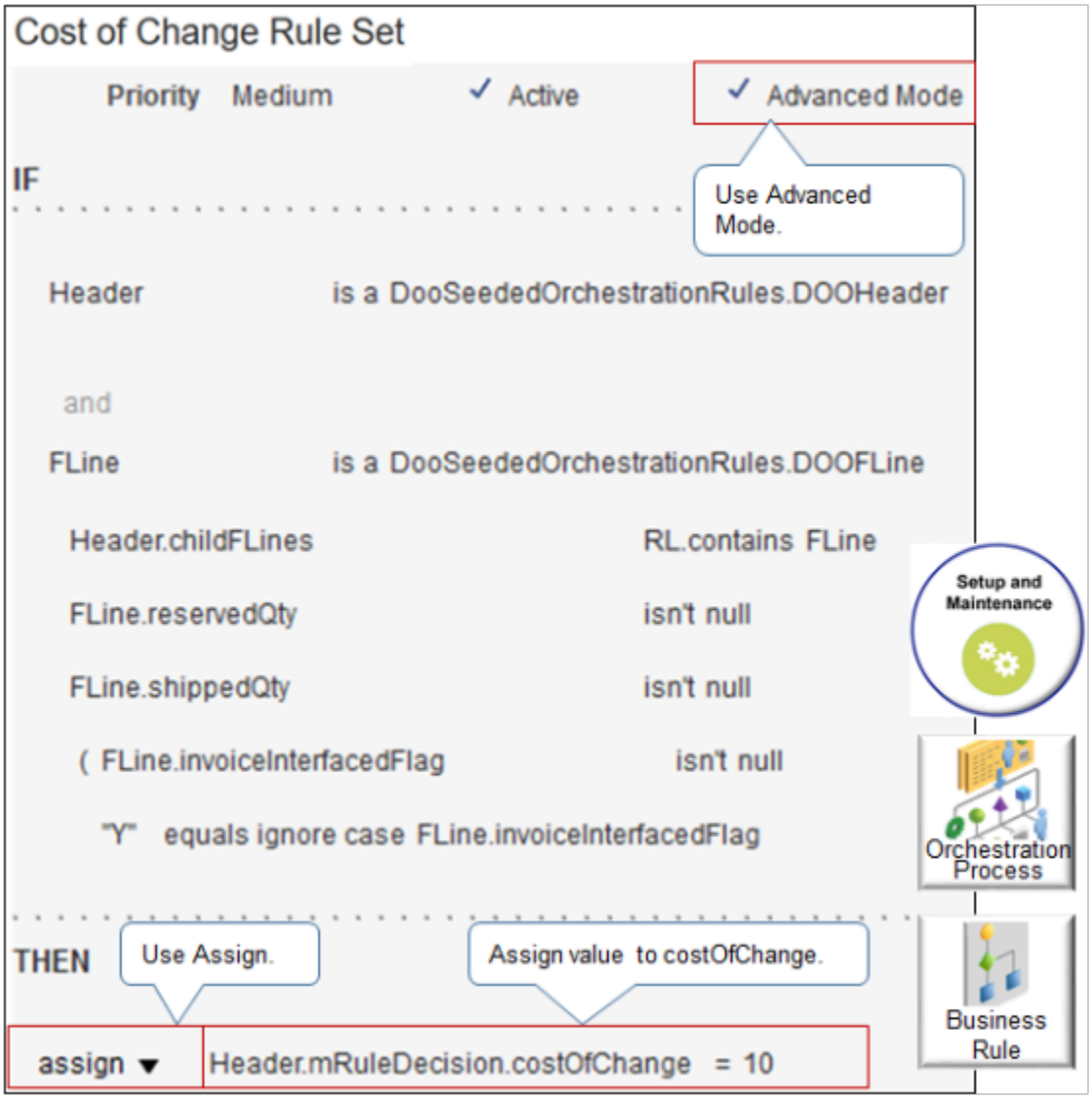
Reserve Supply

If you reserve or schedule an order line according to future supply, then consider whether your orchestration process needs the reserve or schedule step. For example, the reservation step examines supply that's currently available, and then reserves it to make sure your fulfillment system can fulfill the item. If you don't need to reserve supply, then don't

include the reservation step. If you do need to reserve, then consider making the reservation step a manual step, for example, so the user must click a button, such as Reserve.

You can also add a pause step before the reserve or schedule step. The pause task can pause the orchestration process until the shipment is almost ready to ship, such as 12 hours before scheduled shipment. When the reservation step runs, it will provide a more accurate picture of supply that's available because its closer to the ship date. Waiting to reserve supply in this way can also reduce the cost of holding inventory until its time to ship.

Set Up Rule for Cost of Change



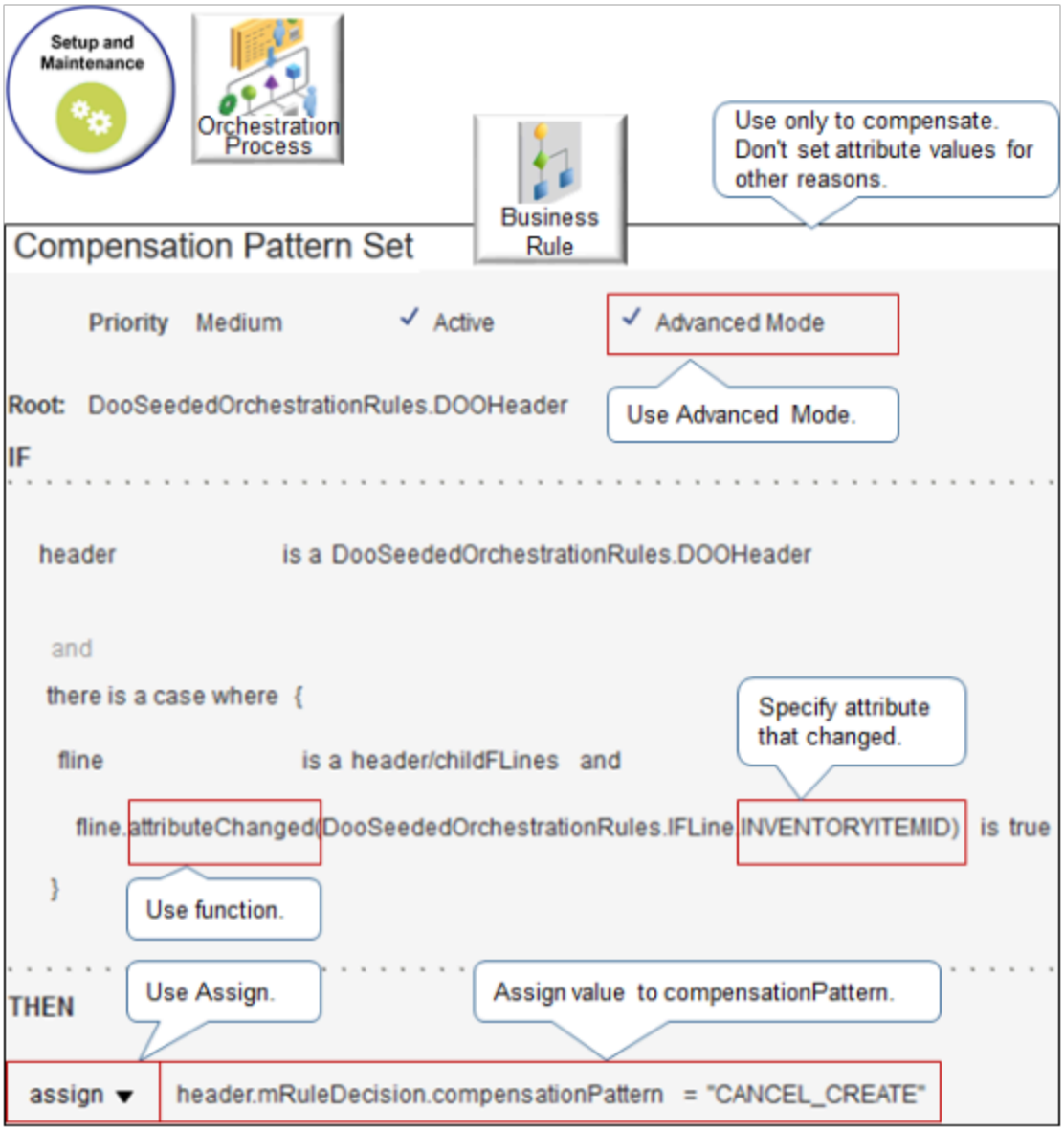
Note

- Set up a rule that measures the cost of change. If its too high, reject the change.
- In the header area of the orchestration process, next to Cost of Change Rule, click **Click for Rule**.
- Use Advanced Mode.
- Use the Assign action.

- Assign a numeric value to Header.mRuleDecision.costOfChange. For example:

```
Header.mRuleDecision.costOfChange = 10
```

Set Up Compensation



Note

- Use a compensation rule only to compensate change. If you must also set the values for other attributes, then use a posttransformation rule.
- In the steps area of the orchestration process, in the step you must compensate, in column Compensation Pattern, click **Click for Rule**.
- Use Advanced Mode.

- In the IF statement.
    - Use a change function, such as attributeChanged or Changed.
    - Specify the attribute that changed. For example, if the user changed the Item attribute, then use INVENTORYITEMID.
- For example:

```
Fline.attributeChanged(DooSeededOrchestrationRules.IFLine. INVENTORYITEMID)is true
```

  Get the list of attributes that you can specify. For details, see *Entities and Attributes That You Can Use When You Integrate Order Management with Other Oracle Applications*.
- In the THEN statement.
    - Use the Assign action.
    - For example:

```
header.mRuleDecision.compensationPattern = "CANCEL_CREATE"
```

    - Assign a value.

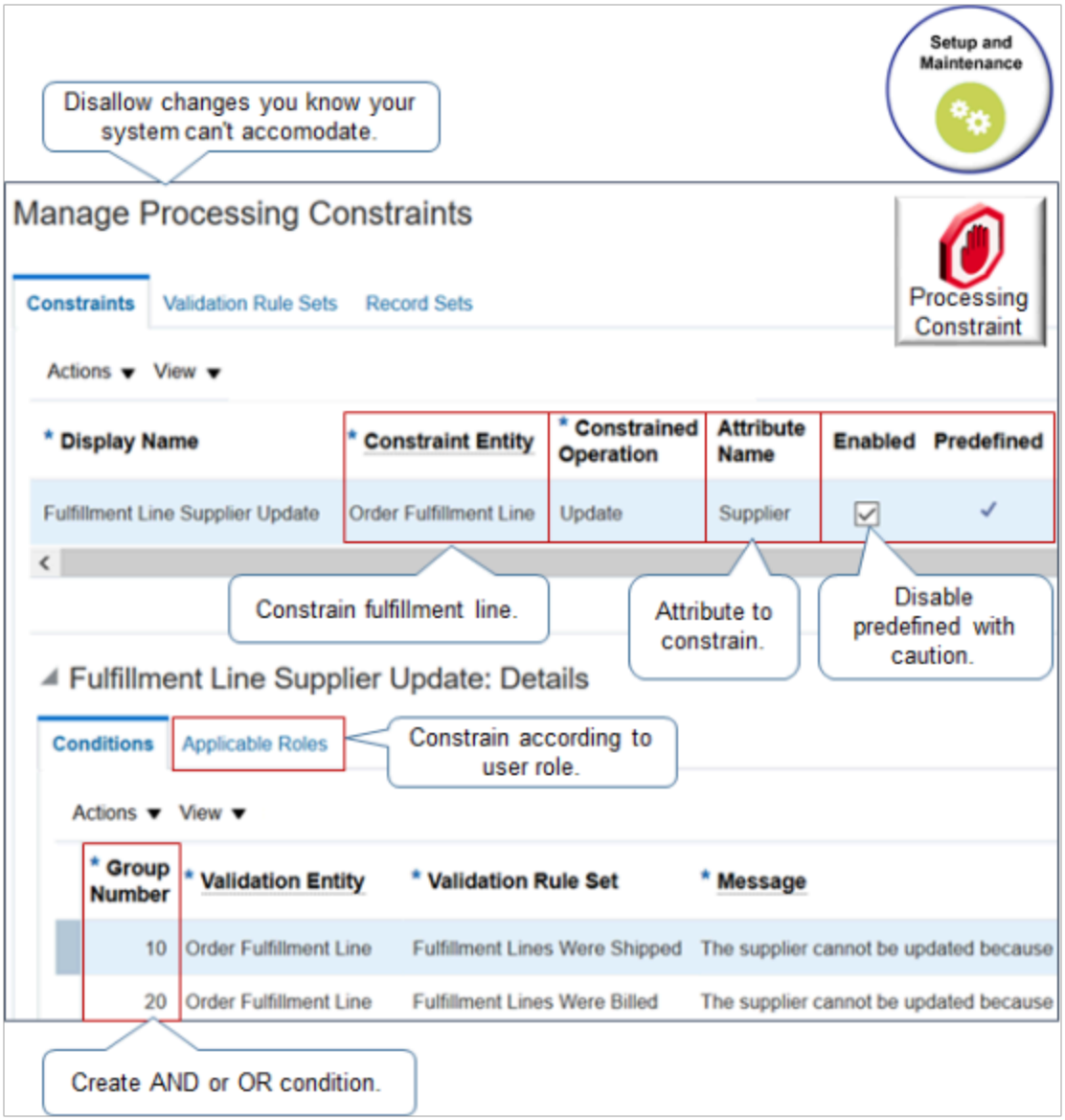
Value	Description
UPDATE	Update the current fulfillment request with the latest attribute value. Each orchestration process comes predefined to use UPDATE, by default. If you don't specify a compensation pattern, and if the orchestration process detects a change, then it will send an update request to the fulfillment system.
UPDATE_CREATE	Update the current fulfillment request with the latest attribute value, and also create a new request.
CANCEL_CREATE	Cancel the current fulfillment request, and then create a new request that includes the latest attribute value.
CANCEL_UPDATE	Cancel the current fulfillment request, and then update another fulfillment request with the latest attribute value.
CANCEL_UPDATE_CREATE	Cancel the current fulfillment request, update another fulfillment request with the latest attribute value, and then create a new request that includes the latest attribute value.
NOOP	NOOP (No Operation). Don't do any compensation.

  These values determine what request the orchestration process sends to the fulfillment system.

Constrain Change

Use a processing constraint.

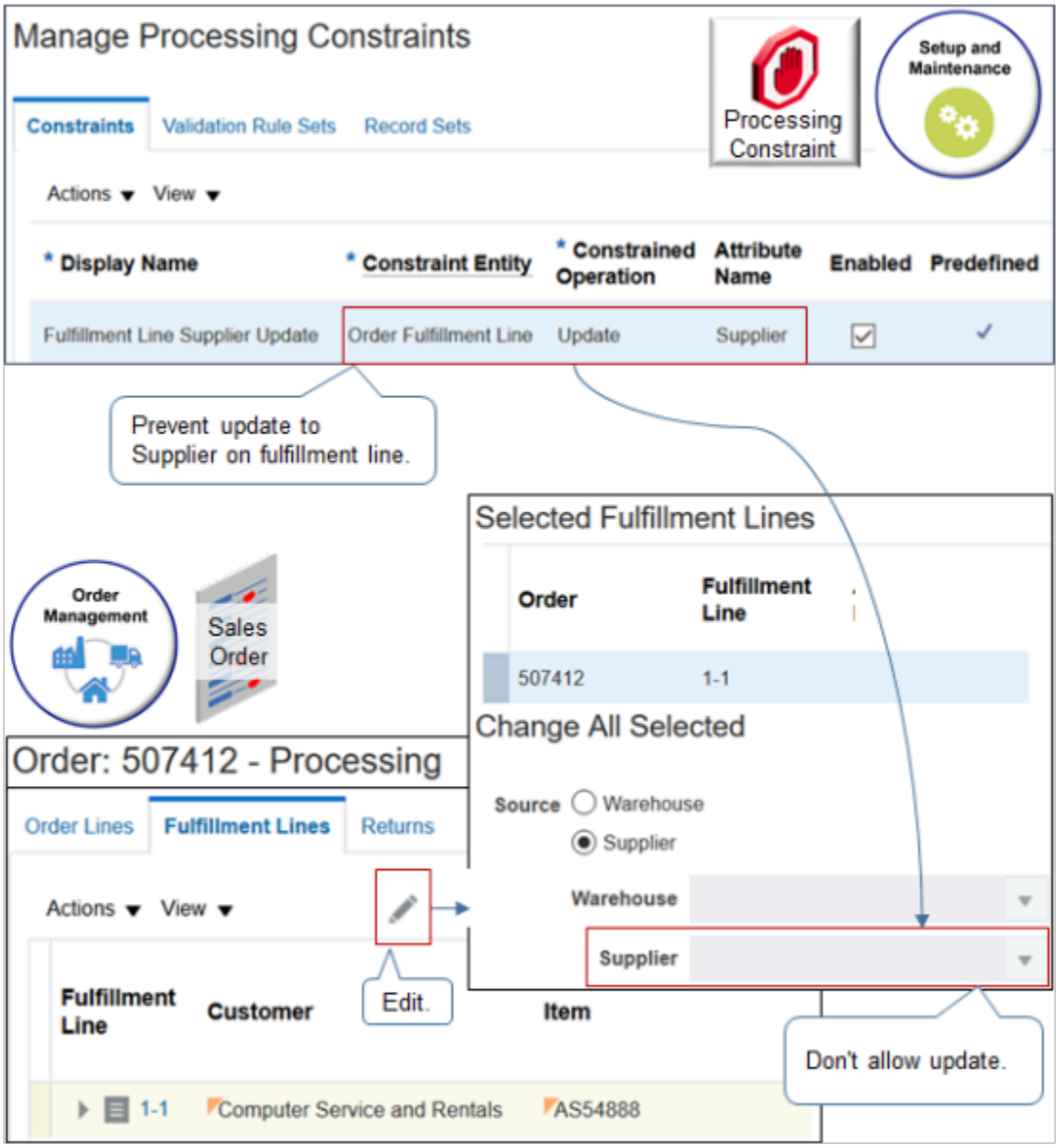
- Disallow change that you know your fulfillment system can't accommodate, such as changing the supplier for an item that requires significant lead time for the supplier to produce.
- Make sure a channel doesn't make a change that your fulfillment system can't accommodate.
- Make sure the user includes required attributes. For example, assume you add your own numeric attribute x for invoicing, and invoicing requires attribute x to calculate the invoice. You can use a constraint to make sure the user adds a value for x, and that the value is numeric.
- If a constraint doesn't meet your business requirement, then set up an order management extension. Use an extension to validate, then display a message or warning.



Note

- Set the Constraint Entity attribute to Order Fulfillment Line.
- If the Enabled attribute is active on a predefined constraint, then you can disable it. Disable a predefined constraint only after you make sure removing the constraint won't negatively affect your implementation.
- Use the same value in the Group Number attribute to create an AND condition across constraints. For example, if you set Group Number to 10 for condition x, and Group Number to 10 for condition y, and if conditions x and y both evaluate to True, then Order Management applies the constraint.
- Use different values in Group Number to create an OR condition. For example, if you set Group Number to 10 for condition x, and Group Number to 20 for condition y, and if condition x or y evaluates to True, then Order Management applies the constraint.
- Use the Applicable Roles tab to specify the user roles that can or can't edit an attribute according to the constraint.
- Use the Manage Processing Constraints page in the Setup and Maintenance work area.
- For details, see *Manage Processing Constraints*.

For example:



Here's the logic that this constraint implements.

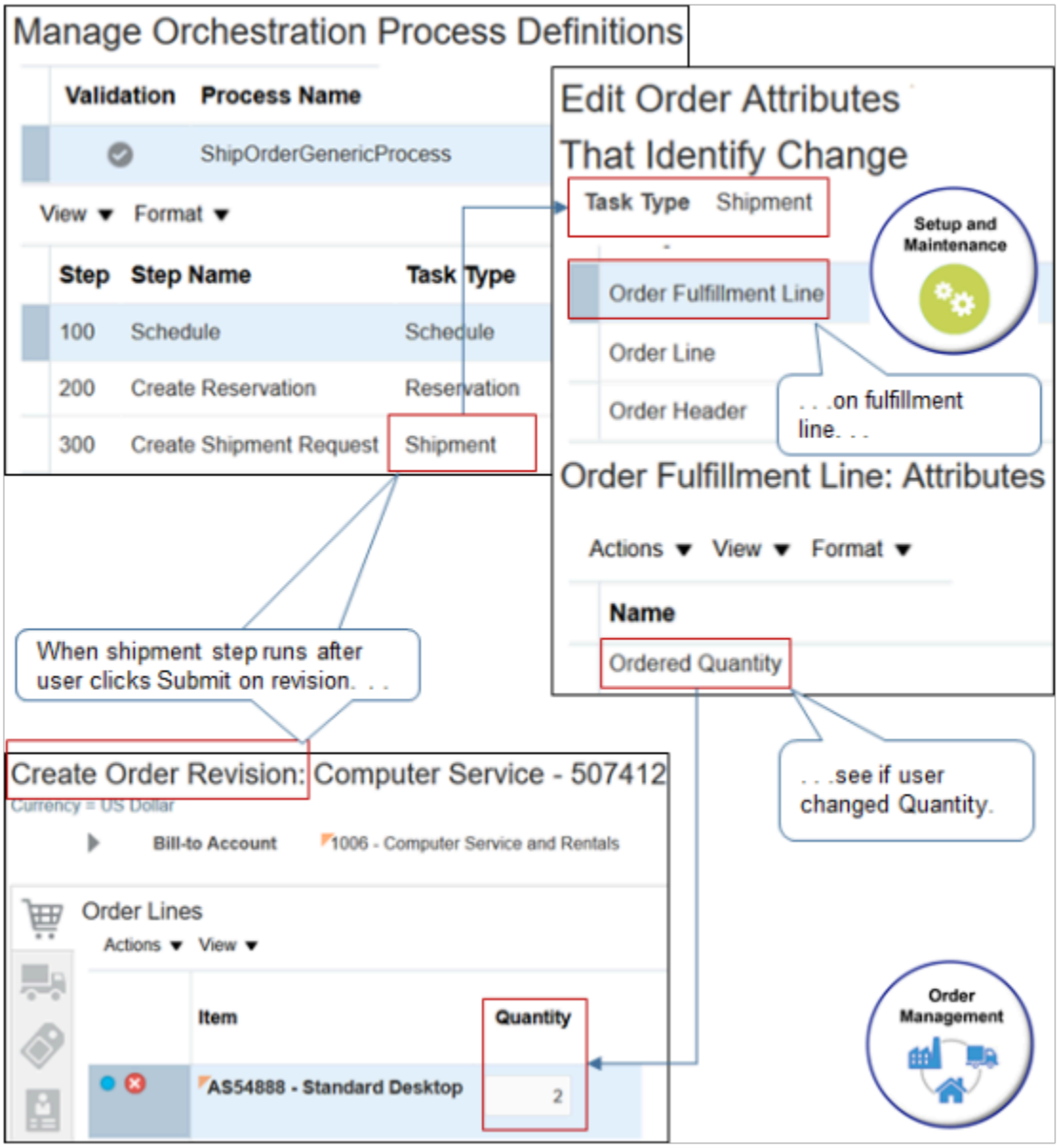
```
Prevent user from updating attribute Supplier on fulfillment line.
```

You set the values.

Attribute	Value
Constraint Entity	Order Fulfillment Line
Constrained Operation	Update
Attribute Name	Supplier

Attribute	Value
Enabled	Contains a check mark

Manage Attributes That Identify Change



This example implements a condition.

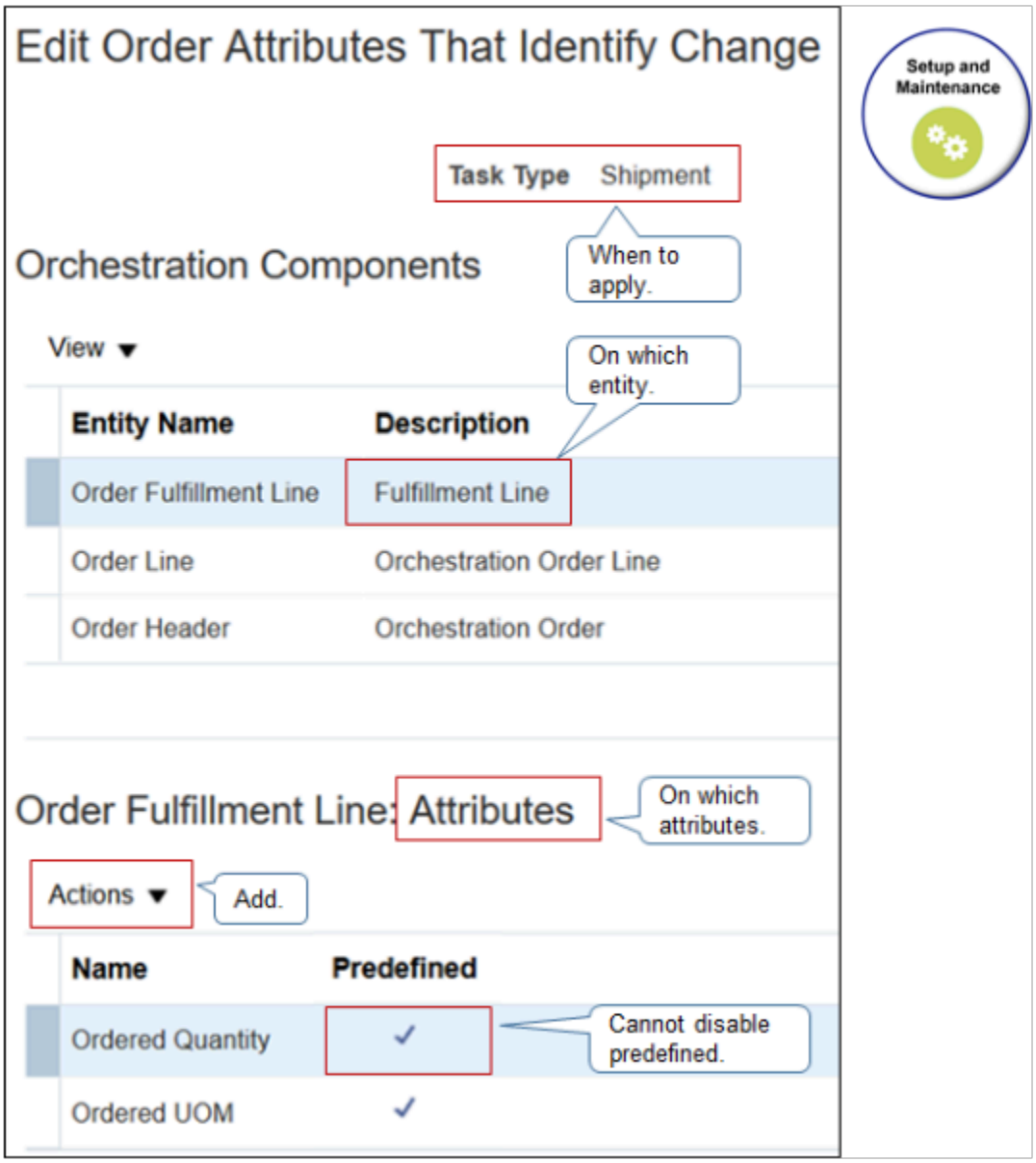
```
If the user changes Quantity on fulfillment line, then perform compensation during shipping.
```

Note

- The user creates a revision, clicks Submit, and then the orchestration process starts.

- The orchestration process reaches a step that uses the Shipment task type, then examines the Manage Order Attributes That Identify Change page in the Setup and Maintenance work area to determine whether to process the change.
- For details, see *Manage Order Attributes That Identify Change*.

For example:



Note

- Set the Task Type attribute to a value, such as Schedule, Reservation, Shipment, and so on, to determine when to examine the change.

- Specify an entity to determine where the attribute resides.
  - Order Fulfillment Line
  - Order Line
  - Order Header
- Specify the attribute to examine, such as Ordered Quantity.
- Use the Add action to add an attribute.
- You can add a predefined attribute, such as Ordered Quantity, but you can't disable it.

Other Guidelines

Order Management doesn't run pretransformation rules when you use Create Order Revision. If you must set the default attribute value on a new order line that the user adds when using Create Order Revision, then create a posttransformation rule that sets the default value.

Your users can use the fulfillment view to change a large number of fulfillment lines. If you set up a number of business rules that must run for each change, and if the user submits a large set of changes, then you might encounter degraded performance. You must test how many fulfillment line updates your environment can accommodate, then limit updates from the fulfillment view according to the optimal.

Related Topics

- Processing Constraints
- Manage Order Attributes That Identify Change

Manage Order Attributes That Identify Change

Set up Order Management to examine an order attribute that identifies change in a change order, then compensate the sales order.

Assume you must create an orchestration process step that uses the Ordered Quantity attribute to identify change.

This topic uses example values. You might need different values, depending on your business requirements.

Manage order attributes that identify change.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Attributes That Identify Change
2. On the Manage Order Attributes That Identify Change page, in the Task Type list, click a task type that looks like it might meet the needs that your orchestration process step requires.

   Order Management comes predefined with a number of task types. To reduce maintenance, use a predefined task type before you create a new one.
3. Click the row that includes Shipment, then click **Actions > Edit**.
4. On the Edit Order Attributes That Identify Change page, in the Orchestration Components list, click a **component** that you think might contain the attribute that Order Management must use to identify change.

   For this example, click **Order Line**.
5. Examine the Attributes list, then take action.

- The orchestration process step examines each attribute in this list to determine whether to compensate the sales order when it receives a change order. If the list includes the attributes that your step requires to identify change, then set the Task Type attribute on the step to this task type, then go to the next step in this procedure.

  For this example, the list does include Ordered Quantity, so set Task Type to Shipment.

  If the Predefined column in the Attributes list includes a check mark, then you can't remove this attribute.
- If the Attributes list doesn't include the attributes your step needs to identify change, then do these steps.
- Click **Actions > Select and Add**.
- In the Change Attributes dialog, click an attribute, such as **Scheduled Ship Date**.
- Click **Apply > OK**.

Here are some set ups you can do on the Manage Order Attributes That Identify Change page.

Set Up	Description
Add an attribute to all tasks that the Task Type list displays.	Click **Actions > Edit All**.
Add new task type.	Click **Actions > Create**. On the Create Order Attributes That Identify Change page, add attributes, as necessary. If you create a new task type, and if you don't use this page to add at least one attribute to the task type, then Order Management won't examine any attributes when it identifies change.

Related Topics

- Guidelines for Setting Up Orchestration Processes

Measure Cost and Compensate

Measure the Cost of Change

Set up Order Management to measure the cost of change.

Cost of change is a numeric value that measures how much a change impacts an orchestration process. For example, the monetary cost to your company, or the difficulty that's associated with incorporating the change. You can create a business rule that measures the cost of change for an orchestration process.

If the source system requests a determination for cost of change, then Order Management calculates the value, then returns it to the source system so the customer service representative can choose whether to proceed with the change. The source system can request the value before it submits the sales order. Order Management also calculates the cost after it compensates the sales order.

You use a business rule to assign the cost of change to an orchestration process. If you choose not to use values for the cost of change, then Order Management uses a value of zero to calculate cost.

Assume you need a business rule that measures the cost of change your company will incur when a customer requests a change. If fulfillment line status is.

- Scheduled, then cost is low
- Shipped, then cost is high

Here are the rules you will create.

- If fulfillment line status is Scheduled, then cost of change is 5.
- If fulfillment line status is Shipped, then cost of change is 50.

This example includes a simple business rule you can use with an orchestration process that includes only one line. You use advanced rules to write a rule that includes more than one line. For details, see *Overview of Using Business Rules With Order Management*.

Summary of the Steps

1. Set up your source system.
2. Create If statement for first rule.
3. Create Then statement for first rule.
4. Create If statement for second rule.
5. Create Then statement for second rule.
6. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Set Up Your Source System

Order Management doesn't display the result when it calculates the cost of change. Instead, you must set up your source system to get the result from Order Management, then display it so someone who uses the source system can take the necessary action. To get this result, you can use REST API.

Create If Statement for First Rule

Here's the first rule.

where

Object	Description
DooSeededOrchestrationRules	Dictionary that contains rule sets, facts, functions, variables and so on for order orchestration.
DOOFLine	Fact in DooSeededOrchestrationRules. It contains fulfillment line attributes. DOOFLine is an abbreviation for distributed order orchestration (DOO) fulfillment line (FLine). The phrase distributed order orchestration is an earlier name for Oracle Order Management.
statusCode	Fulfillment line attribute. The entire value isn't visible in the screen capture. Here's the full value.

Object	Description
	`DooSeededOrchestrationRules.DOOFLine.statusCode`
SCHEDULED	One value that statusCode might contain.
Result	Fact in DooSeededOrchestrationRules. You use it to store the results of the business rule you're defining.
resultObjKey	Property of the Result fact.
esultObjKey:5	Sets the value of resultObjKey to 5.

Create If statement for first rule.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the orchestration process where you must add the cost of change rule, then open it for editing.
3. In the Orchestration Process area, next to Cost of Change Rule, click **Click for Rule**.
4. In the Cost of Change Rule dialog, click **Add > General Rule > Properties**, then set the values.

Attribute	Value
Name	Cost of Change for Scheduled Lines
Description	This rule measures the cost to change a sales order when the fulfillment line status is Scheduled. It sets the cost of change to 5.

5. Click **Left Value**.
6. In the Condition Browser dialog, expand **DooSeededOrchestrationRules > DOOFLine**, then click **StatusCode > OK**.
7. In the Right Value attribute, enter `"SCHEDULED"`. You must include the double quotation marks.

Create Then Statement for First Rule

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.

4. In the Properties dialog, in the resultObjKey row, enter this value, then click **OK**.

Attribute	Value
Value	5

5. Click **Collapse**.

Create If Statement for Second Rule

Here's the second rule.

Create the If statement for the second rule.

1. Click **Add > General Rule > Properties**, then set the values.

Attribute	Value
Name	Cost of Change for Shipped Lines
Description	This rule measures the cost to change a sales order when fulfillment line status is Shipped. It sets the cost of change to 50.

2. Click **Left Value**.
3. In the Condition Browser dialog, expand **DooSeededOrchestrationRules > DOOFLine**, then click **StatusCode > OK**.
4. In the Right Value attribute, enter `"SHIPPED"`. You must include the double quotation marks.

Create Then Statement for Second Rule

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Properties dialog, in the resultObjKey row, in the Value column, enter `50`, then click **OK**.
5. In the Cost of Change Rule dialog, click **Save**.
6. On the Edit Orchestration Process Definition page, click **Save**.

Test Your Set Up

1. Change a sales order that includes a fulfillment line status that's Scheduled.
2. Wait for Order Management to process the change.
3. Make sure your source system displays a value of 5 for the cost of change.
4. Change a sales order that includes a fulfillment line status that's Shipped.
5. Wait for Order Management to process the change.
6. Make sure your source system displays a value of 50 for the cost of change.

Related Topics

- Set Up Orchestration Processes

Compensate Sales Orders That Change

Create a compensation pattern that makes an adjustment when a fulfillment task changes a sales order.

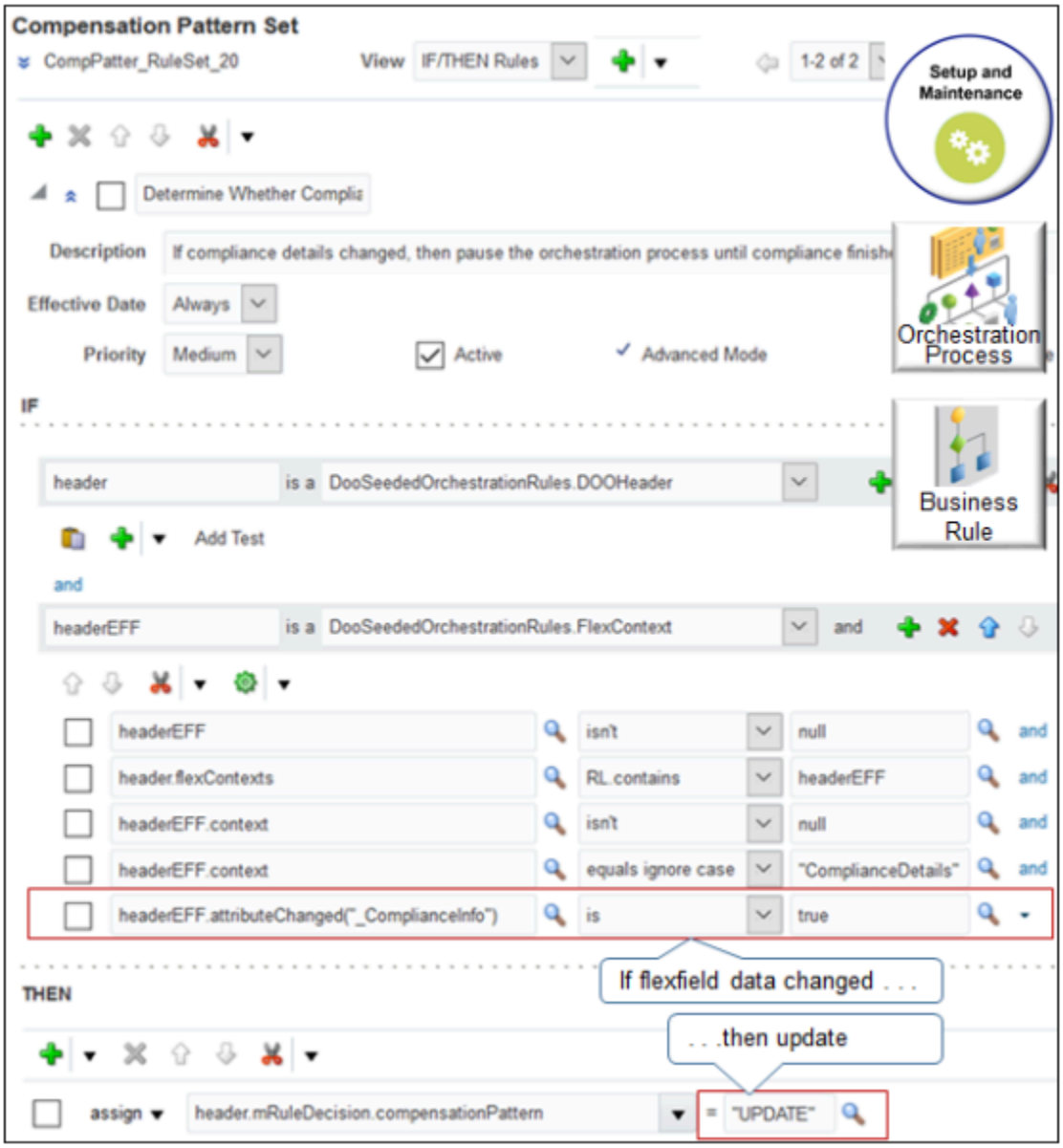
A compensation pattern is a rule you create on an orchestration process step. It specifies the adjustment to make when an order changes. Undo, Redo, Update, Cancel, and None are each an example of a compensation pattern.

For example, assume the compensation pattern for a Create Shipment step is Redo, and that the step calls the Cancel service and the Create service. If Order Management receives a change order that includes a new warehouse for the step, then it runs Cancel and Create again.

Assume you need a compensation pattern that pauses the orchestration process until compliance check finishes.

- If compliance details change, then pause the orchestration process until compliance is done.

Here's the rule you will create.



where

Code	Description
`DooSeededOrchestrationRules`	Dictionary that contains a set of predefined business rules for an orchestration process. You use it to store objects and their values. DOO means distributed order orchestration, which is a term that Order Management used for order orchestration in earlier releases.
`DOOHeader`	Object in DooSeededOrchestrationRules. It contains attributes for the sales order header and their values.

Code	Description
`headerEFF`	Extensible flexfield on the order header. For this example, assume you defined an extensible flexfield that stores details about the compliance check. For details, see *Overview of Setting Up Extensible Flexfields in Order Management*.
`DooSeededOrchestrationRules.FlexC`	Establishes FlexContexts in the dictionary.
`FlexContext`	Object that stores the value for the extensible flexfield.
`ComplianceDetails`	Object that stores compliance details.
`ComplianceInfo`	Object that stores compliance information.
`mRuleDecision.CompensationPatterr`	CompensationPattern is a property of object mRuleDecision. CompensationPattern stores the string UPDATE, which is the value that this rule uses to end the condition.

Here's a description of how the statement works.

Code	Description
`header is a DooSeededOrchestrationRules.DOOHe`	Declare **`header`** as a temporary variable that stores the value from object DOOHeader of dictionary DooSeededOrchestrationRules. Proceed to the next AND statement only after you declare **`header`**.
`headerEFF is a DooSeededOrchestrationRules.FlexC`	Declare **`headerEFF`** as a temporary variable that stores the value from object FlexContext of DooSeededOrchestrationRules. Proceed to the next AND statement only after you declare **`headerEFF`**.
`headerEFF isn't null`	Proceed to the next AND statement only if the extensible flexfield on the sales order header contains a value. This set up uses headerEFF to store the value of the extensible flexfield that stores attribute Compliance Details on the order header. If the value of Compliance Details is empty, then it indicates compliance isn't required, compensation isn't required, and the orchestration process continues to the next orchestration process step.
`header.flexContexts RL.contains headerEFF`	Declare headerEFF into the rules language (RL) dictionary, and set the value of headerEFF to the value that flexContexts contains. This condition makes sure you correctly declare the variable into the dictionary. Proceed to the next AND statement only if RL contains headerEFF. Do this test on each of your variables to make sure the declaration is correct. If you don't do the test, and if the declaration isn't correct, then the rule might fail in a subsequent step.
`headerEFF.context isn't null`	Make sure a context is defined for the order header. Proceed to the next AND statement only if **`context`** contains a value.

Code	Description
`headerEFF.context equals ignore case "ComplianceDetails"`	Get the context and segments for Compliance Details for this variable. Proceed to the next AND statement only if **`context`** contains the string ComplianceDetails. Ignore case sensitivity when examining. if **`context`** contains some other value, then it indicates that no other process requested a compliance check.
`headerEFF.attributeChanged("_ComplianceInfo") is true`	Make sure attribute ComplianceInfo of the extensible flexfield changed. You use attributeChanged to determine whether the value for ComplianceInfo is different in this revision.

This example includes a simple rule that processes only one line. Use an advanced rule to process more than one line. For details, see *Overview of Using Business Rules With Order Management*.

Summary of the Steps

1. Create If statement.
2. Create Then statement.
3. Release pause task.

This topic uses example values. You might need different values, depending on your business requirements.

Create If Statement

You will create a statement.

- If compliance details changed

Do it.

1. Create the rule.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Orchestration Process Definitions
    - On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.

      Learn how to create CallCustomerWhenLargeInvoice. For details, see *Add Branches to Orchestration Processes*.
    - On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, click **Change Management**.
    - In the Ship Item row, in the Compensation Pattern column, click **Click for Rule**.
    - In the Compensation Pattern dialog, click **Add Rule > Properties**, then set the values.

Attribute	Value
Name	Determine Whether Compliance Changed

Attribute	Value
Description	If compliance details changed, then pause the orchestration process until compliance is done.
Effective Date	Always
Priority	Medium
Active	Contains a check mark.
Advanced Mode	Contains a check mark.
Tree Mode	Doesn't contain a check mark.

2. Declare variable **`header`**.
    - In the If area, in the **window** to the left of Is A, enter **`header`**.
    - Set the field to the right of Is A to DooSeededOrchestrationRules.DOOHeader.
3. Declare variable **`headerEFF`**.
    - Click **Add Pattern**.
    - In the window to the left of Is A, enter **`headerEFF`**.
    - Set the field to the right of Is A to DooSeededOrchestrationRules.DOOHeader.
4. Make sure **`headerEFF`** contains a value.
    - Click **Add Test**.
    - In the window to the left of Is, enter **`headerEFF`**.
    - Change Is to Isn't.
    - In the field to the right of Isn't, click **Right Value > null > OK**.
5. Declare headerEFF into the rules language (RL) dictionary.
    - Click the down arrow located to the right of Right Value, then click **Simple Test**.
    - In the window to the left of Is, click **Left Value**, expand **header**, then click **flexContexts > OK**.
    - Change Is to RL.contains.
    - In the field to the right of RL.contains, click **Right Value > headerEFF > OK**.

6. Make sure `context` contains a value.
    - Click **Simple Test**.
    - Near the window to the left of Is, click **Left Value**, expand **headerEFF**, then click **context > OK**.
    - Change Is to Isn't.
    - In the field to the right of Isn't, click **Right Value > null > OK**.
7. Make sure `context` contains the string ComplianceDetails.
    - Click **Simple Test**.
    - Near the window to the left of Is, click **Left Value**, expand **headerEFF**, then click **context > OK**.
    - Change Is to Equals Ignore Case.
    - In the field to the right of Equals Ignore Case, click **Right Value**, enter `"ComplianceDetails"`, then click **OK**.

      Make sure you include the double quotation (") marks.
8. Make sure attribute ComplianceInfo of the extensible flexfield changed.
    - Click **Simple Test**.
    - Near the window to the left of Is, click **Left Value**, expand **headerEFF**, and then click **hasChanges**.
    - In the window in the Condition Browser, notice the value.

      Insert `"_ComplianceInfo"` between the parentheses, then click **OK**. For example:

      ```
      headerEFF.attributeChanged("_ComplianceInfo")
      ```
    - In the window to the right of Is, enter the value.

Create Then Statement

Create the statement that pauses the orchestration process until compliance is done.

1. In the Then area, click **Add Action > Assign**.
2. Click **Select a Target**, then click **header.mRuleDecision.compensationPattern**.
3. In the window to the right of the equal sign (=), enter `"UPDATE"`.

   You must include the double quotation marks.
4. Click **Validate**, make sure the Validation Log that displays doesn't contain any errors, then click **Save**.
5. On the Edit Orchestration Process Definition page, click **Save**.

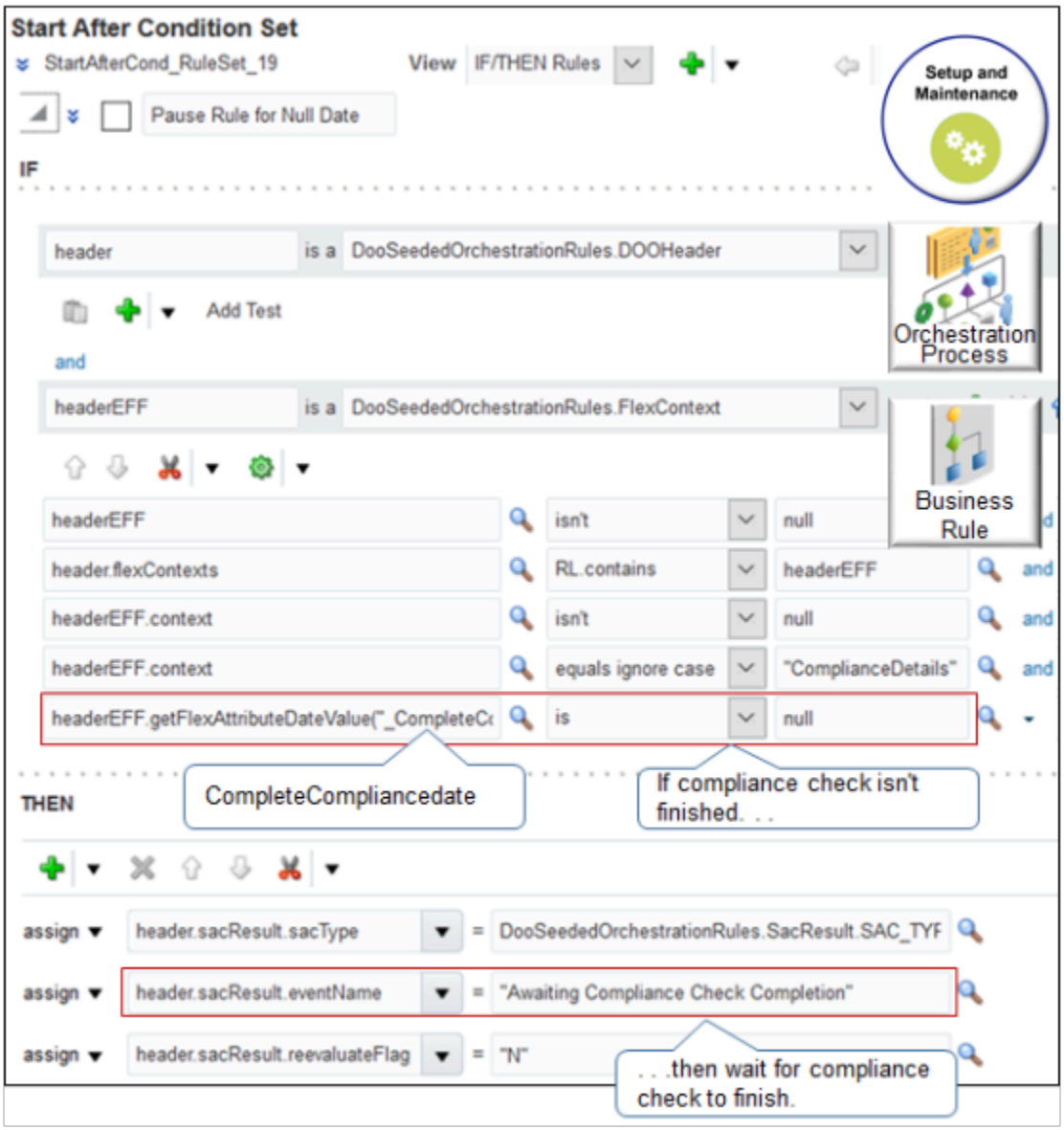
Pause the Task According to Date

You add this rule in the Pause Rule column on the same step where you added the compensation pattern.

Pause Task While Date Is Empty

Order Management sets the compliance completion date when compliance check is done. If the date is empty, then it indicates compliance check isn't done, and the rule doesn't release the pause task. For example:

- If CompleteCompliancedate is empty, then compliance check isn't done. Assign a business event to indicate we must wait.



Here's the code that the rule uses.

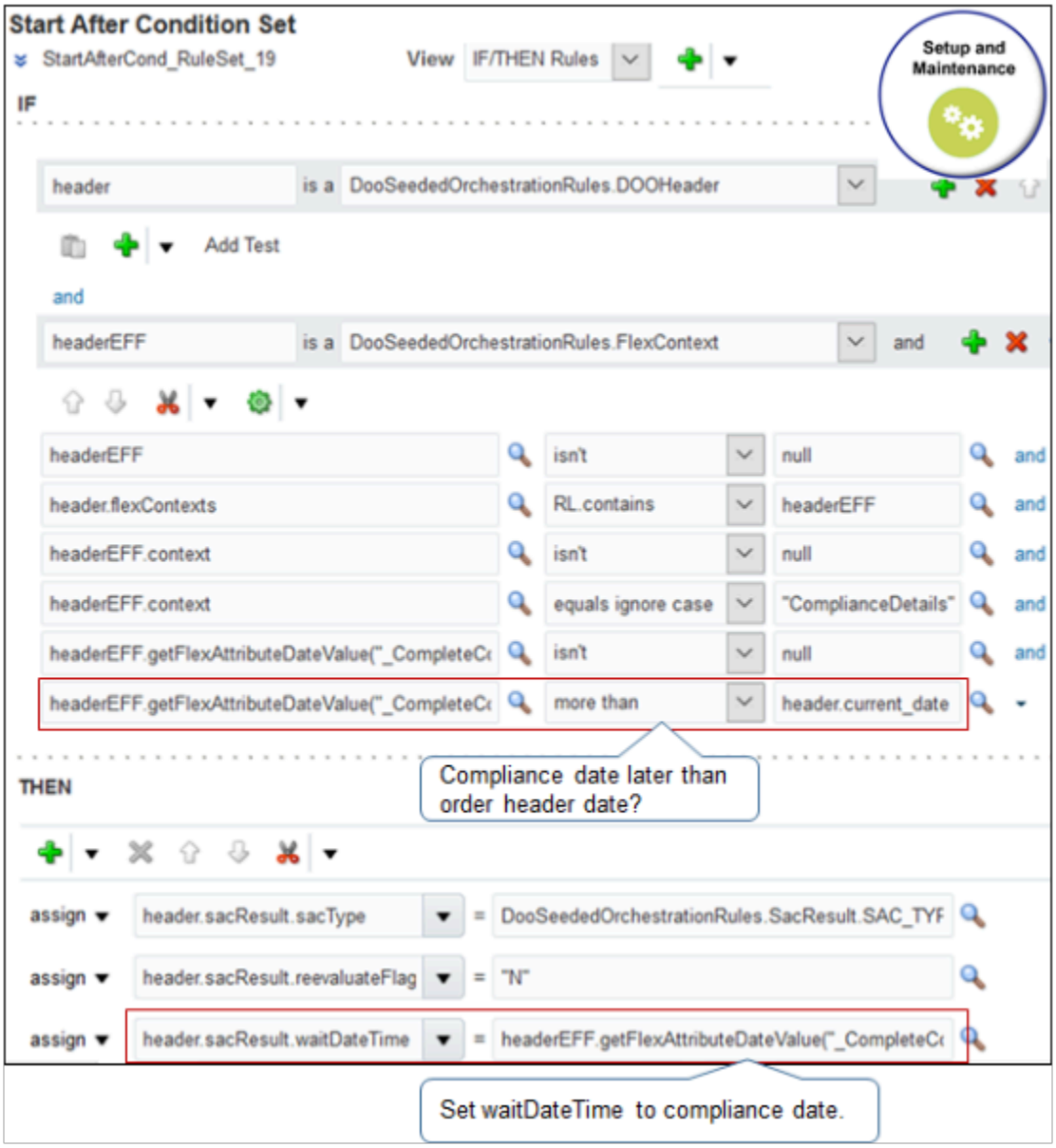
```
If header is a DooSeededOrchestrationRules.DOOHeader
and if headerEFF is a DooSeededOrchestrationRules.FlexContext and
 headerEFF isn't null and
 header.flexContexts RL.contains headerEFF and
 headerEFF.context isn't null and
 headerEFF.context equals ignore case "ComplianceDetails" and
 headerEFF.getFlexAttributeDateValue("_CompleteCompliancedate") is null

then
 assign header.sacResult.sacType = DooSeededOrchestrationRules.SacResult.SAC_TYPE_EVENT
 assign header.sacResult.eventName = "Awaiting Compliance Check Completion"
 assign header.sacResult.reevaluateFlag = "N"
```

Pause for Future Date

You can create a rule that pauses until compliance completion date is later than order header date. For example:

- If CompleteCompliancedate occurs after order header date, then compliance check is done. Prepare to release the pause task.

To prepare to release the pause task, you set waitDateTime to the compliance date. For details about waitDateTime and sacResults, see *Guidelines for Pausing Orchestration Processes*.



Here's the code that the rule uses.

```
If header is a DooSeededOrchestrationRules.DOOHeader
and if headerEFF is a DooSeededOrchestrationRules.FlexContext and
 headerEFF isn't null and
 header.flexContexts RL.contains headerEFF and
 headerEFF.context isn't null and
 headerEFF.context equals ignore case "ComplianceDetails" and
 headerEFF.getFlexAttributeDateValue("_CompleteCompliancedate") isn't null
```

```
    headerEFF.getFlexAttributeDateValue("_CompleteCompliancedate") more than header.current_date

then
    assign header.sacResult.sacType = DooSeededOrchestrationRules.SacResult.SAC_TYPE_TIMER
    assign header.sacResult.waitDateTime = headerEFF.getFlexAttributeDateValue("_CompleteCompliancedate")
```

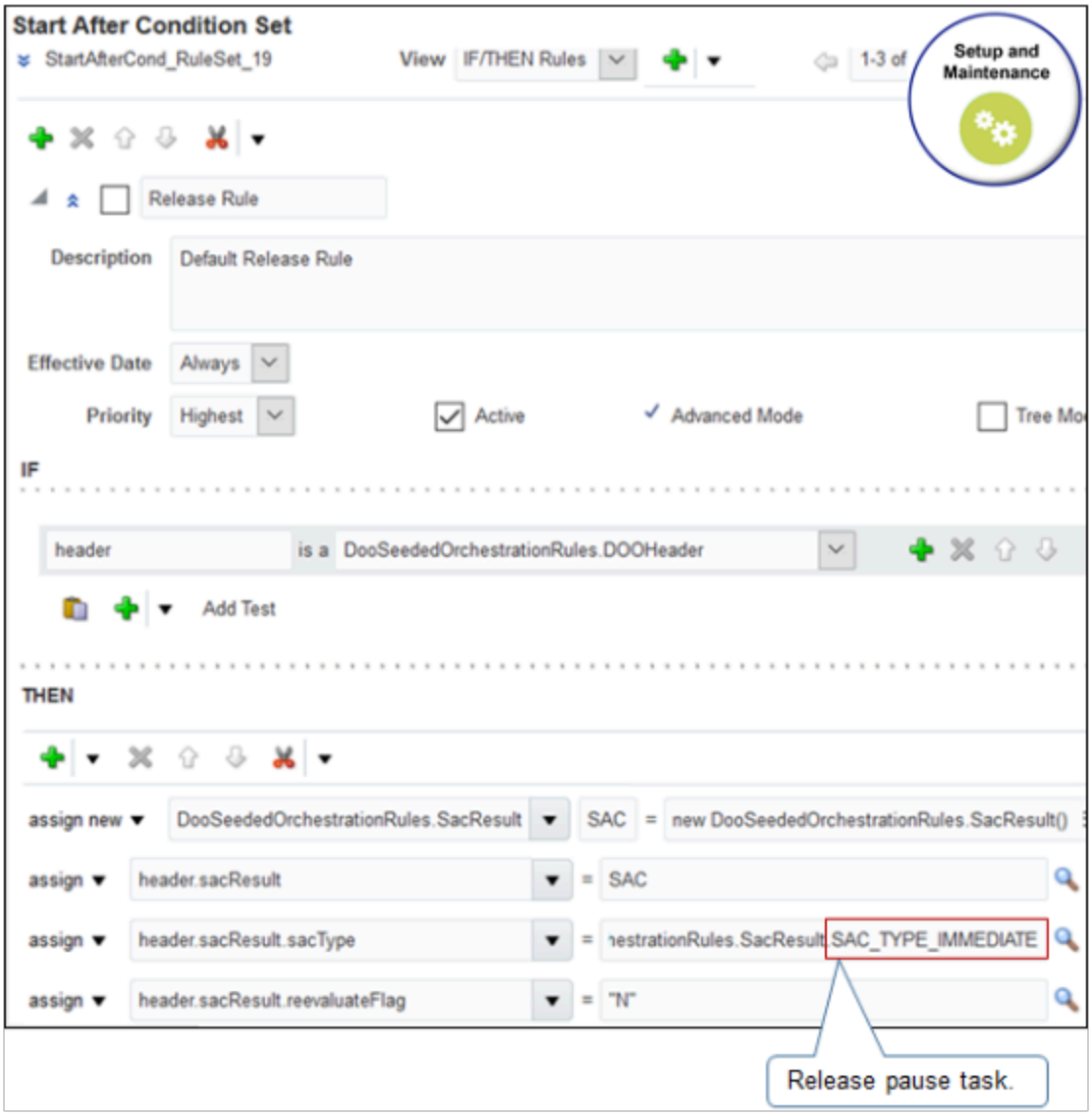
Release Pause Task

Create a rule that releases the pause task. It doesn't reference the extensible flexfield you set up earlier in this topic. Instead, it uses rules.

- Pause Rule for Null Date
- Pause Rule for Future Date

It uses the values that these rules set for SacResult to determine when to release the pause task.

Here's the rule that releases the pause task.



Here's the code that this rule uses.

```
If header is a DooSeededOrchestrationRules.DOOHeader

then
 assign new DooSeededOrchestrationRules.sacResult SAC = new DooSeededOrchestrationRules.sacResult()
 assign header.sacResult = SAC
 assign header.sacResult.sacType = DooSeededOrchestrationRules.SacResult.SAC_TYPE_IMMEDIATE
 assign header.sacResult.reevaluateFlag = "N"
```

You add this rule in the Pause Rule column on the same step where you added the compensation pattern.

Related Topics

- Roadmap for Setting Up Order-to-Cash

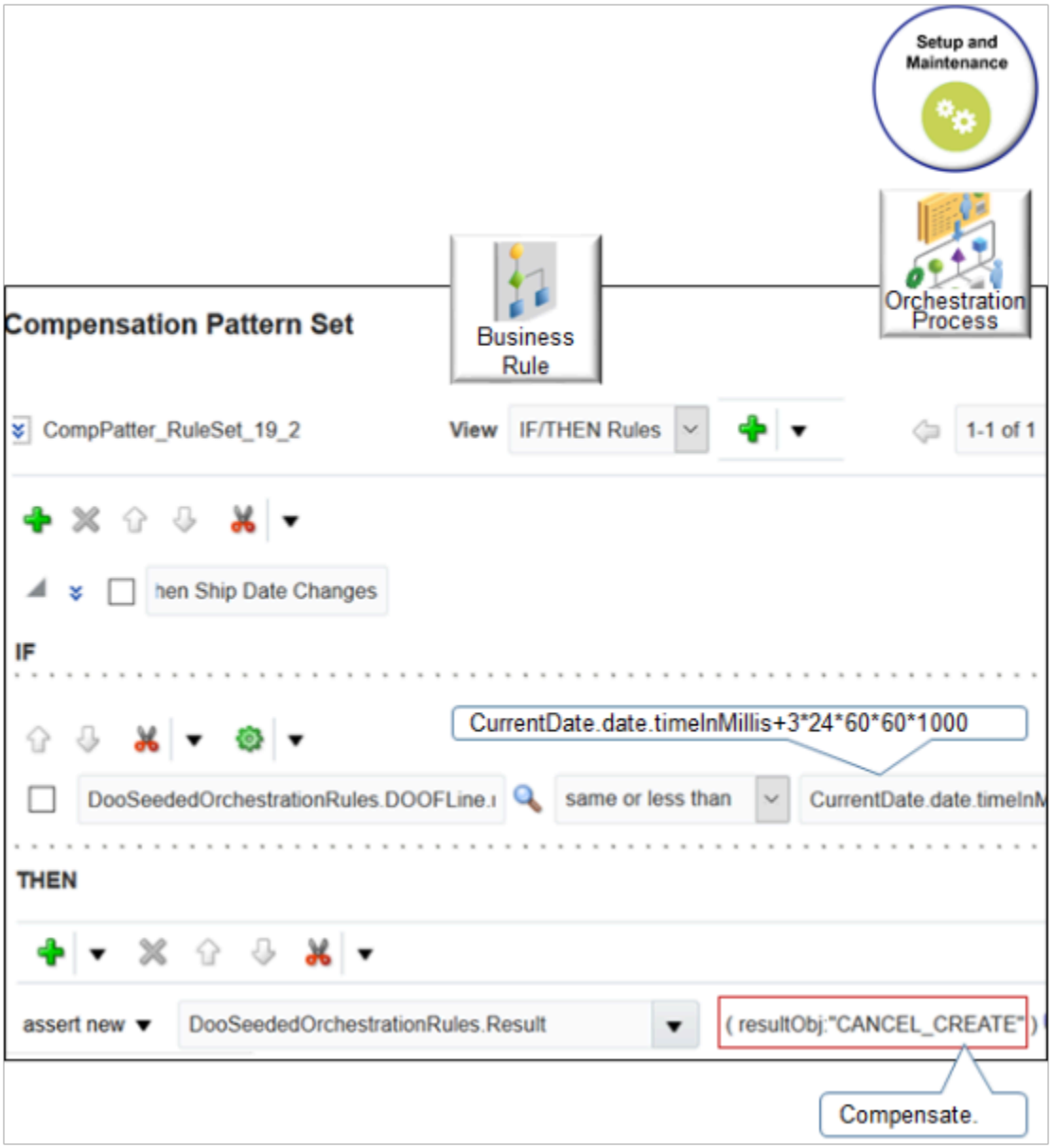
Another Example of Compensating Sales Orders That Change

Set up a compensation pattern that allows time to reschedule an orchestration process when a customer requests to change the ship date.

Here's the rule you will create.

- If the requested ship date is less than or equal to the current date plus three days, then cancel and redo the Ship Product step.

Here's where you create the rule.



Here's the code that the rule uses.

```
If
DooSeededOrchestrationRules.DOOFLine.requestShipDate same or less than CurrentDate.date.timeInMillis
+3*24*60*60*1000

Then
assert new DooSeededOrchestrationRules.Result (resultObj."CANCEL_CREATE")
```

where

Code	Description
`DooSeededOrchestrationRules`	Dictionary you use to store objects and their values.

Code	Description
	DOO means distributed order orchestration, which is term that Order Management used for order orchestration in earlier updates.
`DOOFLine`	Object in DooSeededOrchestrationRules. It contains fulfillment line attributes and their values.
`requestShipDate`	Fulfillment line attribute in DOOFLine.
`time`	Property of the toLocateString function. This function references the requestShipDate attribute.
`CurrentDate.date.timeInMillis +3*24*60*60*1000`	Equation that specifies to use the date that occurs three days after the current date.
`Result`	Object in DooSeededOrchestrationRules that you use to store the result of the If statement.
resultObj	Property of the Result object. It stores CANCEL_CREATE, which is the value that this rule uses to end the condition.

Summary of the Steps

1. Create If statement.
2. Create Then statement.

This topic uses example values. You might need different values, depending on your business requirements.

Create If Statement

You will create a statement.

- If the requested ship date is less than or equal to the current date plus three days

Do it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the CallCustomerWhenLargeInvoice orchestration process, then click **Actions > Edit**.

    Learn how to create CallCustomerWhenLargeInvoice. For details, see *Add Branches to Orchestration Processes*.
3. On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, click **Change Management**.
4. In the Ship Item row, in the Compensation Pattern column, click **Click for Rule**.

5. In the Compensation Pattern dialog, click **Add Rule > Expand**, then set the value.

Attribute	Value
Name	Rule to Compensate When Ship Date Changes

6. In the If area, click **Left Value**, expand **DOOSeededOrchestrationRules > DOOFline > requestShipDate**, then click **time > OK**.
7. In the Is list, click **Is > Same or Less Than**.
8. Click **Right Value**.
9. In the Condition Browser, expand **CurrentDate > Date > Time**, then click **timeInMillis**.

   Don't click OK.
10. Near the top of the Condition Browser, in the window, after `CurrentDate.date.timeInMillis`, add `+3*24*60*60*1000`.

    Make sure the window now contains this value:

    ```
    CurrentDate.date.timeInMillis+3*24*60*60*1000
    ```

    The time function for most business rules work in milliseconds, so you must convert time to milliseconds. The calculation 3*24*60*60*1000 calculates the total number of milliseconds that three days contains, where.

    - 3 is the number of days.
    - 24 is the number of hours in one day.
    - 60 is the number of minutes in one hour.
    - 60 is the number of seconds in one minute.
    - 1000 is the number of milliseconds in one second.
11. Click **OK**.

Create Then Statement

Create the statement that cancels, then restarts the Ship Product step so Order Management can reschedule the shipment.

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target**, then click **DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Properties dialog, in the ResultObj row, enter the value.

Attribute	Value
Value	"CANCEL_CREATE" You must include the double quotation marks (").

5. Click **OK**.
6. Click **Validate**, make sure the Validation Log that displays doesn't contain any errors, then click **Save**.

7. On the Edit Orchestration Process Definition page, click **Save**.

Related Topics

- Roadmap for Setting Up Order-to-Cash

Compensate Sales Orders That Are Awaiting Shipping

The compensation pattern for the Shipping task uses Cancel and Create to substitute an item during shipping because Oracle Shipping supports only cancel and create. You can modify your compensation rule to support shipping.

If you don't use Oracle Shipping, and if your shipping system:

- **Supports Update.** You don't need to modify anything.
- **Doesn't support Update.** You must modify the compensation rule so it uses Cancel and Create.

Note

- If you need different behavior, then you can write a compensation rule according to an item change. Note that business rules call the Update service for an item change.
- Change management can only identify change to an extensible flexfield. However, you can write a compensation rule that examines the extensible flexfield segment so it can determine whether to do compensation.

Here's the rule you will create.

- If the sales order is at the await shipping step, and if the Order Entry Specialist substituted the item, then cancel the existing request to ship, create a new request, schedule, reserve, and then send the new request to shipping.

Each shipping step and reservation step on a predefined orchestration process that integrates with shipping already use this logic. You add this logic only if you set up your own orchestration process.

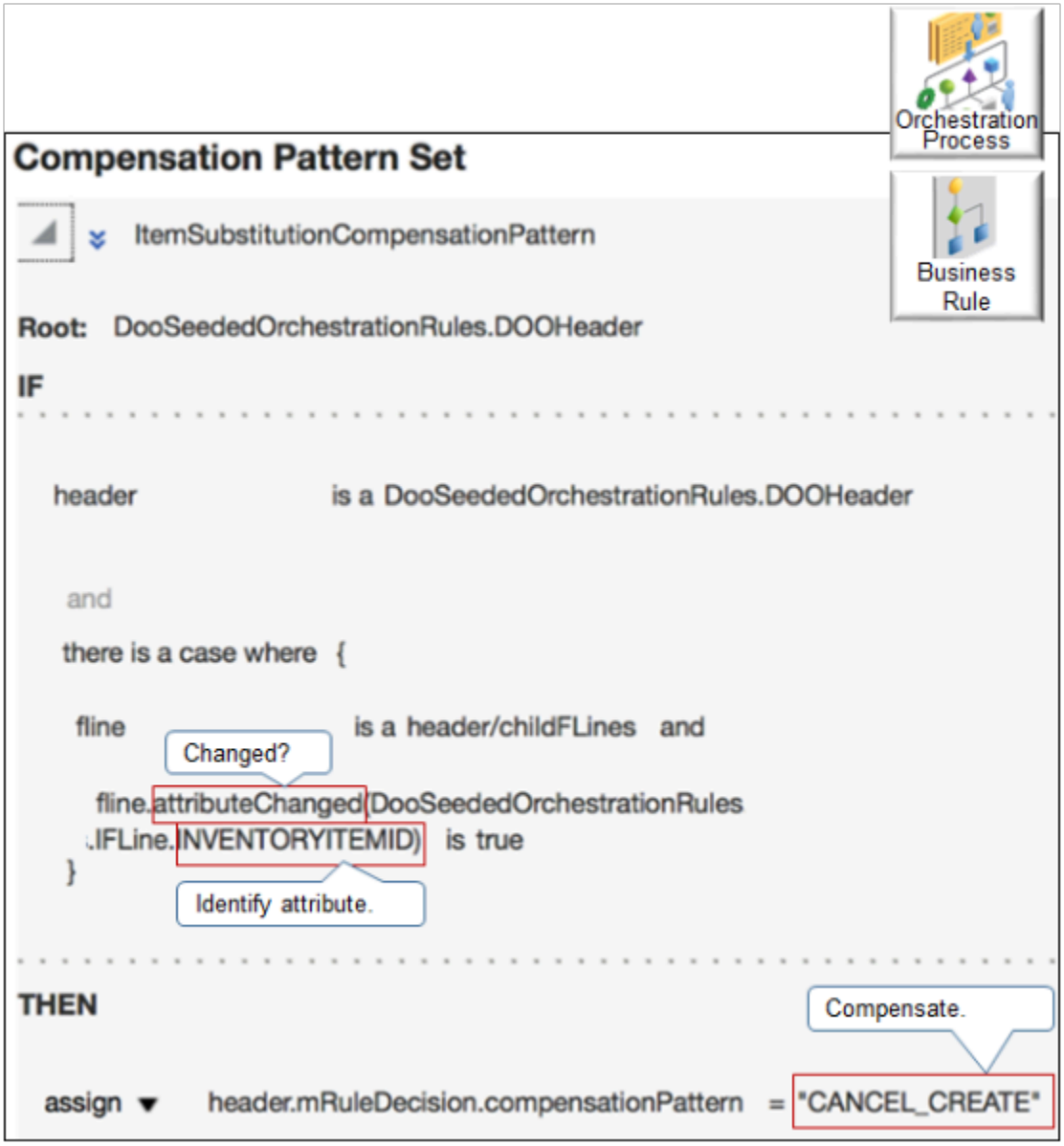
Compensate sales orders that are awaiting shipping.

1. Use the Manage Orchestration Process Definitions page to open and edit the orchestration process you must modify.
2. In the row that ships the item, in the Compensation Pattern column, click **Click for Rule**, then add the rule.

Here's the code to use.

```
Root: DooSeededOrchestrationRules.DOOHeader
If
 header is a DooSeededOrchestrationRules.DOOHeader
 and At least one
 fline is a header/childFlines
 and fline.attributeChanged(DooSeededOrchestrationRules.IFLine.INVENTORYITEMID) is true
then
 assign header.mRuleDecision.compensationPattern = "CANCEL_CREATE"
```

Here's what it looks like.



Related Topics

- Processing Constraints
- Manage Order Attributes That Identify Change

Notify Systems

Overview of Sending Notifications from Order Management to Other Systems

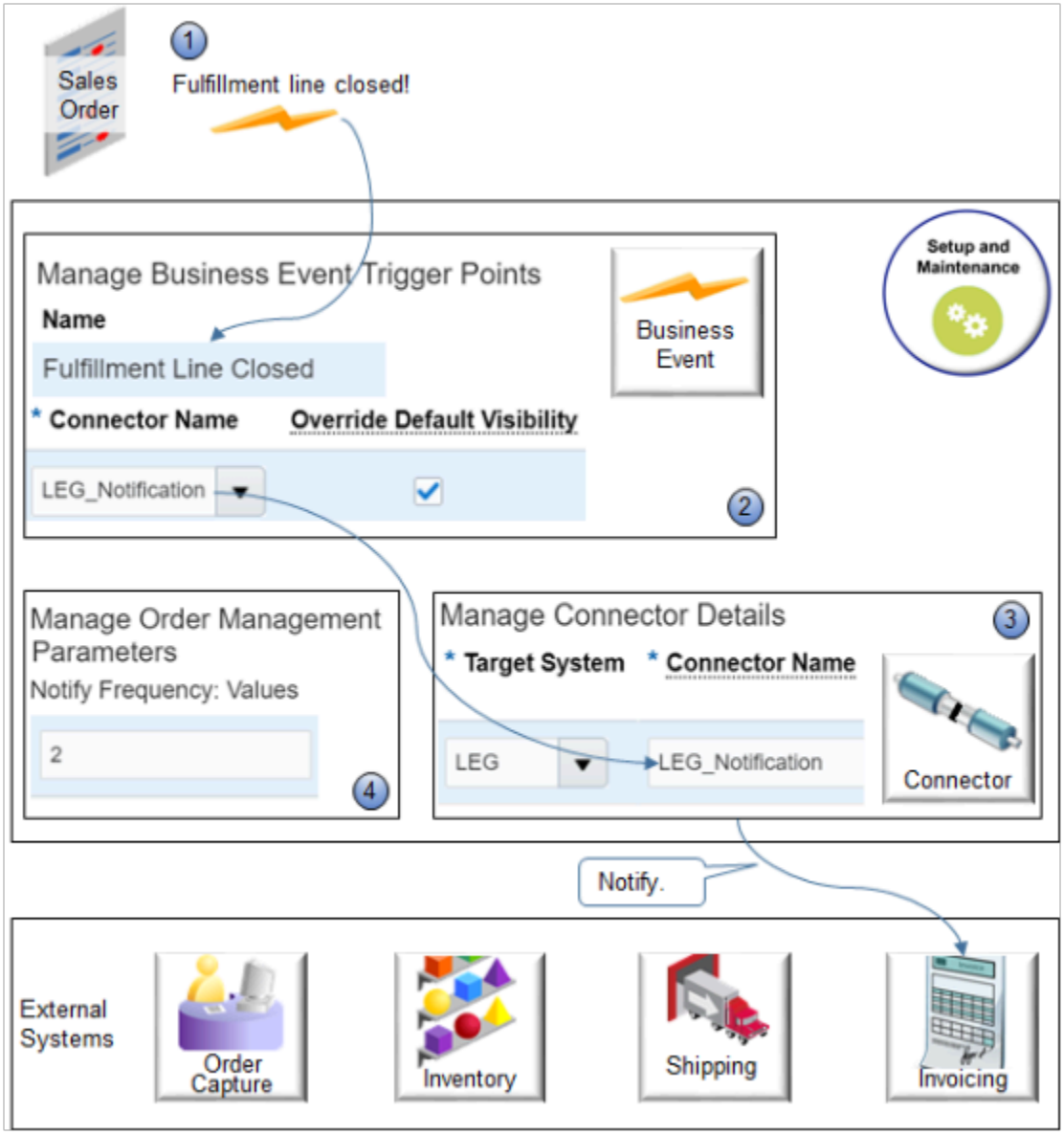
Use a business event to send a notification to a system that resides outside of Oracle Order Management when a change occurs, such as when details in a sales order or fulfillment line change.

You can send the notification to an upstream system, such as an order capture system, or to a downstream system, such as a fulfillment system or billing system.

A business event is something that occurs that's significant enough that requires Order Management to take an action. Here are some examples.

- A change to an order attribute occurs.
- Order Management applies a hold.
- A jeopardy priority changes.
- A fulfillment line splits.
- The status on an order header changes.
- A fulfillment line closes.
- A fulfillment line achieves a status.
- Compensation for an orchestration process finishes successfully or not successfully.

Here's how it works.



Note

1. Order Management monitors conditions that occur during order fulfillment according to the trigger points that you specify.

   For example, if you enable the Notify External System option for the Shipped status value on a status rule set, then Order Management communicates the business event each time it sets a fulfillment line status to Shipped, and it sends the business event to the endpoint URL of the connector that you set up for the Fulfillment Line Status Update business event. For details, see *Add Status Conditions to Fulfillment Lines*.
2. You use the Manage Business Event Trigger Points page in the Setup and Maintenance work area to specify the trigger point. If Order Management determines a trigger point occurred, then it uses the Connector Name attribute on the Manage Business Event Trigger Points page to identify the connector it will use to communicate with the system.
3. The connector communicates with the system. You use the connector to specify the URL that locates the system and the security credentials that your system needs to communicate data. For example, the fulfillment line status is Shipped, so you notify your invoicing system that the order is ready to invoice.

4. As an option, you can use the Notify Frequency parameter to specify how frequently Order Management sends notifications. For details, see *Manage Order Management Parameters*.

You can also use a web service to communicate details about the event.

Manage Business Event Trigger Points

A business event trigger point is the condition that causes a business event to occur.

- You must associate a connector with the business event that you specify. For details, see *Send Notifications from Order Management to Other Systems*.
- The business events that this page displays aren't active, by default, except for business events that monitor closing a fulfillment line.
- A modification that you make on this page takes effect immediately.
- The settings that you make affect only one instance of Order Management.
- You can't add or delete the business events that this page displays.
- Set up only the business event trigger points that you need. Each trigger point might affect performance.

Use Predefined Trigger Points

Predefined Business Event	Description
Change Order Compensation Complete	Send a notification when Order Management finishes processing a change order. If the change order results in an error, then this business event reports the error.
Fulfillment Line Status Update	Send a notification when Order Management changes a fulfillment line's status value. You can specify the status values that the trigger notification. 1. Navigate to the Manage Orchestration Process Definitions page. 2. On the Edit Status Rule Set page, add a check mark to the Notify External System option next to each status value that you want to trigger.
Fulfillment Line Closed	Send a notification when Order Management closes a fulfillment line. Some applications, such as Cost Management, can use these details to do downstream processing on the fulfillment line.
Hold	Send a notification when a user applies a hold, or when the order capture system or fulfillment system applies a hold on an entity. • Order • Order line • Fulfillment line Note • If a hold doesn't go into effect immediately, for example, if the hold applies to a future task, then Order Management communicates the business event when the hold goes into effect, not when the request to add the hold occurs. • If change management releases or applies a hold, then Order Management doesn't communicate the business event.
Jeopardy	Send a notification when Order Management changes a jeopardy priority value, such as High, Medium, or Low. You can specify the value to notify.

Predefined Business Event	Description
Order Attribute Update	Send a notification when Order Management changes a fulfillment line attribute during order fulfillment. You can choose each attribute from the set of attributes that the task references. • Each attribute that the Order Update Attributes That Trigger Events section displays can trigger a business event. These attributes are predefined. You can also set up an extensible flexfield that modifies these attributes. If you set up an extensible flexfield, then Order Management displays it in the Select and Add Attributes That Trigger Events dialog. If you select one or more segments, and then click OK or Apply, then Order Management displays the segments that you select in the Order Attribute Update Attributes That Trigger Events section. • If Order Management updates more than one attribute of the Order Attribute Update business event at the same time, then it sends only one notification. This notification includes details about the attributes that it updated. • The status values in the Order Header Status Update, Status Values That Trigger Events section start the Order Header Status Update business event. These values are predefined. You can't add status values in this section.
Order Header Status Update	Send a notification when Order Management cancels, partially closes, or closes a sales order. You can specify the status value to notify.
Split	Send a notification when a partial shipment occurs or when an integrated system splits a fulfillment line.

What Happens If I Need More Than One Event?

Assume you need to send an event when you modify an attribute's value and when you split a line. So you set up the Order Attribute Update business event to update the Scheduled Ship Date attribute, and you set up the Split business event to send a notification when you split a fulfillment line. However, you notice that you receive the split event but you don’t receive the event to update the attribute. This happens because Order Management only sends one business event at a time. It will send only the split event.

Related Topics

- Send Notifications from Order Management to Other Systems
- Set Up Extensible Flexfields in Order Management

Send Notifications from Order Management to Other Systems

Set up Order Management to send a notification to a system that resides outside of Order Management.

Summary of the Steps

1. Add the connector.
2. Manage business event trigger points.
3. Set the notification frequency.

This topic uses example values. You might need different values, depending on your business requirements.

Learn how to do the set up. For details, see *Guidelines for Integrating Order Management* and *Use Integration Cloud Service with Order Management*.

Add the Connector

Add the connector that Order Management uses to communicate with your system.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage External Interface Web Service Details
2. On the Manage Connector Details page, click **Actions > Add Row**, then set the values.

Attribute	Value
Target System	Select the system that must receive the notification. This system subscribes to the business event.
Connector Name	Enter text that describes the purpose of the connector. For example, to indicate that you use this connector to send notifications to a legacy system, enter `LEG_Notification`.
Connector URL	Specify an endpoint URL. The URL is the address of the web service that you deploy on a system that resides outside of Oracle Order Management. Order Management will call this web service, and the web service must accept the payload that Order Management sends. Make sure the web service uses this WSDL: *https://server:port/soa-infra/services/default/DooTaskExternalInterfaceVirtualPartnersComposite/businesseventsconnetor_client_ep?WSDL*
User Name	Enter the user name that the system requires.
Password	Enter the password that the system requires.

Manage Business Event Trigger Points

Manage the business event trigger points that determine when to communicate business events to your system.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Business Event Trigger Points
2. On the Manage Business Event Trigger Points page, select a trigger point.
   At run time, if the sales order reaches the trigger point you specify, then Order Management calls the web service you select in the Connector URL attribute. The web service then sends details of the business event to

the system. For example, to send a notification when a fulfillment line closes, select the Fulfillment Line Closed trigger point.

3. Click **Actions > Add Row**.

   Some trigger points include the Associated Connectors tab in the details area. If this tab displays, then click it to add the connector.

4. Set the values.

Attribute	Value
Connector Name	Select the connector you added earlier in this topic.
Override Default Visibility	Select one. - **Contains a check mark.** Send a notification about each sales order even if the system that receives the notification doesn't have any knowledge about the sales order. - **Doesn't contain a check mark.** If the system doesn't have any knowledge about the sales order, then don't send the notification about the sales order. Here's an example. - Assume your implementation includes Source System 1 and Source System 2, and that each of these systems can send a source order to Order Management. - Assume Source System 2 has no knowledge of source orders that originate in Source System 1, and that you don't want to notify Source System 2 when an event happens that's associated with a source order that originates in Source System 1. - If Override Default Visibility doesn't contain a check mark, then Order Management won't call the connector for Source System 2, and Source System 2 won't receive the notification.

5. Optional. Repeat step 3 and step 4.

   You can associate the same connector with more than one business event, as necessary.

6. Optional. Repeat step 2 through 5 for each trigger point you must administer.
7. Click **Save and Close**.

Set the Notification Frequency

Use the Notify Frequency parameter to specify how frequently Order Management sends notifications. For details, see *Manage Order Management Parameters*.

Related Topics

- Overview of Sending Notifications from Order Management to Other Systems
- Add Status Conditions to Fulfillment Lines
- Manage Order Management Parameters
- Guidelines for Integrating Order Management
- Overview of Using Business Events with Order Management

Return Orders

Guidelines for Processing Return Orders

Use these guidelines to help you process return orders.

Set Up the Item So Its Returnable

Assume you must return the AS54888 item.

1. Make sure you have the privileges that you need to administer Product Information Management.
2. Go to the Product Information Management work area, then click **Tasks > Manage Items**.
3. On the Manage Items page, search for the AS54888 item, then open it for editing.
4. On the Edit Item page, click **Specifications > Sales and Order Management**.
5. Set the value.

Attribute	Description
Returnable	Yes

6. Click **Save and Close**.

Note that you don't need to do this for a subscription item.

Integrate With Your Source System

Use a web service or import template to integrate your source system so it supports return orders.

Include attributes in your web service payload.

Attribute	Description
UnreferencedReturnFlag	Specify whether the payload references the original return. • Y. The payload doesn't reference the original return. • N. The payload does reference the original return.
Covered attributes, such as CoveredProductIdentifier	Specify covered item details.
CancellationEffectiveDate	Date that reverse billing goes into effect.

Use Web Services

Use web services to integrate Order Management with another system in your deployment.

- OrderInformationService
- OrderFulfillmentResponseService

For details, see *Use Integration Cloud Service with Order Management*.

Use the Order Import Template

Use the DOO_ORDER_DOC_REFERENCES_INT worksheet in the order import template to identify the original order and the item you're returning. For details, see *Overview of Importing Orders Into Order Management*.

Specify the Return-to Location

If you set the Line Type attribute on a return line to Return for Credit and Return the Item, and if the item is shippable, then you must provide the return-to location in the Return Location attribute on the order line. If you create your own return type and use it on the return line, and if you must physically return the item, then we recommend that you create an order management extension or processing constraint that makes sure you provide a value in the Return Location attribute. Order Management uses the Warehouse as the return-to location.

Assign Your Orchestration Process

You must use Oracle Business Rules, an order management extension, or a web service to assign a return order to an orchestration process. You can't use Visual Information Builder to do this.

You must use the ItemCategoryCode attribute to assign the orchestration process. You can't use the CategoryCode attribute or any other attribute in your assignment rule.

Don't Assign Outbound Lines to an Orchestration Process That Processes Returns

If the Line Category attribute on the fulfillment line contains Order, but the orchestration process processes returns, then the predefined DOO_VALIDATE_CREATE_RETURN processing constraint prevents Order Management from sending the line to Oracle Receiving, and you might encounter an error.

```
The request failed because an assignment rule assigned an order line where the Line Category attribute on
 the line contains Order, but the orchestration process processes returns.
```

You can read the constraint as.

```
If the Return Order Contains Standard Order Line rule set is true, and if the Request Validation for Create
 Receipt Advice rule set is true, then constrain the Create Return Service constraint entity according to
 the Default Record Set for the Fulfillment Line.
```

If you have an assignment rule that assigns an order line where the Item Category Code attribute on the line contains Order but the orchestration process processes returns, then we recommend that you modify it so it assigns the orchestration process only when the Item Category Code attribute on the fulfillment line equals Return.

```
If the ItemCategoryCode attribute on the fulfillment line is Return.
```

Specify the originalOrderReference Entity

You must include the originalOrderReference entity only on a line that's a return line.

Return Line?	Then You Must
Yes	Include the originalOrderReference entity.
No	Not include the originalOrderReference entity.

For example, including this code on a line that isn't a return line will fail.

```
"originalOrderReference": [
 {
 "OriginalSourceOrderNumber": 1115463768988074,
 "OriginalSourceLineNumber": 1
 }
```

```
]
```

For another example, including this code on a line that isn't a return line will also fail:

```
"originalOrderReference": [
 {
 "OriginalSourceOrderNumber": null,
 "OriginalSourceLineNumber": null
 }
 ]
```

Avoid Runtime Problems

You can't use the CategoryCode attribute or any other attribute in your assignment rule to assign the orchestration process. You must use the Item Category Code attribute.

Assume you use the Generic Fulfillment Process orchestration process to process your sales orders, and you create your own orchestration process named Return Orders to process return orders. You then create an assignment rule in Visual Information Builder:

```
If CategoryCode equals ORDER, then use the Generic Fulfillment Process orchestration process.
ELSE
If CategoryCode equals RETURN, then use the Return Orders orchestration process.
```

At run time, Order Management might still process the return but it won't assign the correct orchestration process, and you won't see the return behavior that you expect. For example, it might use Generic Fulfillment Process to process all returns, and the status for the return might never change.

Prevent Billing from Rejecting Accounting and Invoicing Rule

If you use an accounting and invoicing rule when you create a sales order, and if you must create a return for an order that references return lines, then do these steps.

1. Disable the predefined DOO_RMA_BILLING_ATTR_CHANGED constraint.
   It verifies the accounting and invoicing rules between the referenced return line and the original order line. You don't need it.
2. Write an order management extension that nullifies the accounting and invoicing rule for referenced return lines.
   - Use the On Start of Submission Request event.
   - Make sure your extension runs only on return lines that Order Management hasn't yet sent to billing, and on lines that aren't closed or canceled.
3. If you already submitted the return and you don't want to revise it, then remove the accounting and invoicing rule in your billing system.
   - Remove the rule only after Order Management sends the line to your billing system.
   - If you use Oracle Receivables, then use Manage AutoInvoice Lines to remove the values.

Example Payload That References the Original Return

Here's part of an example payload that references the original return. Use the Create Order Operation of web service Order Import Service. For brevity, this example includes only the attributes that are relevant for a return. Its an incomplete payload. Examine the complete payload in the return_order_with_reference.xml file. For details, see *Example Web Service Payloads That Integrate Order Management*.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
```

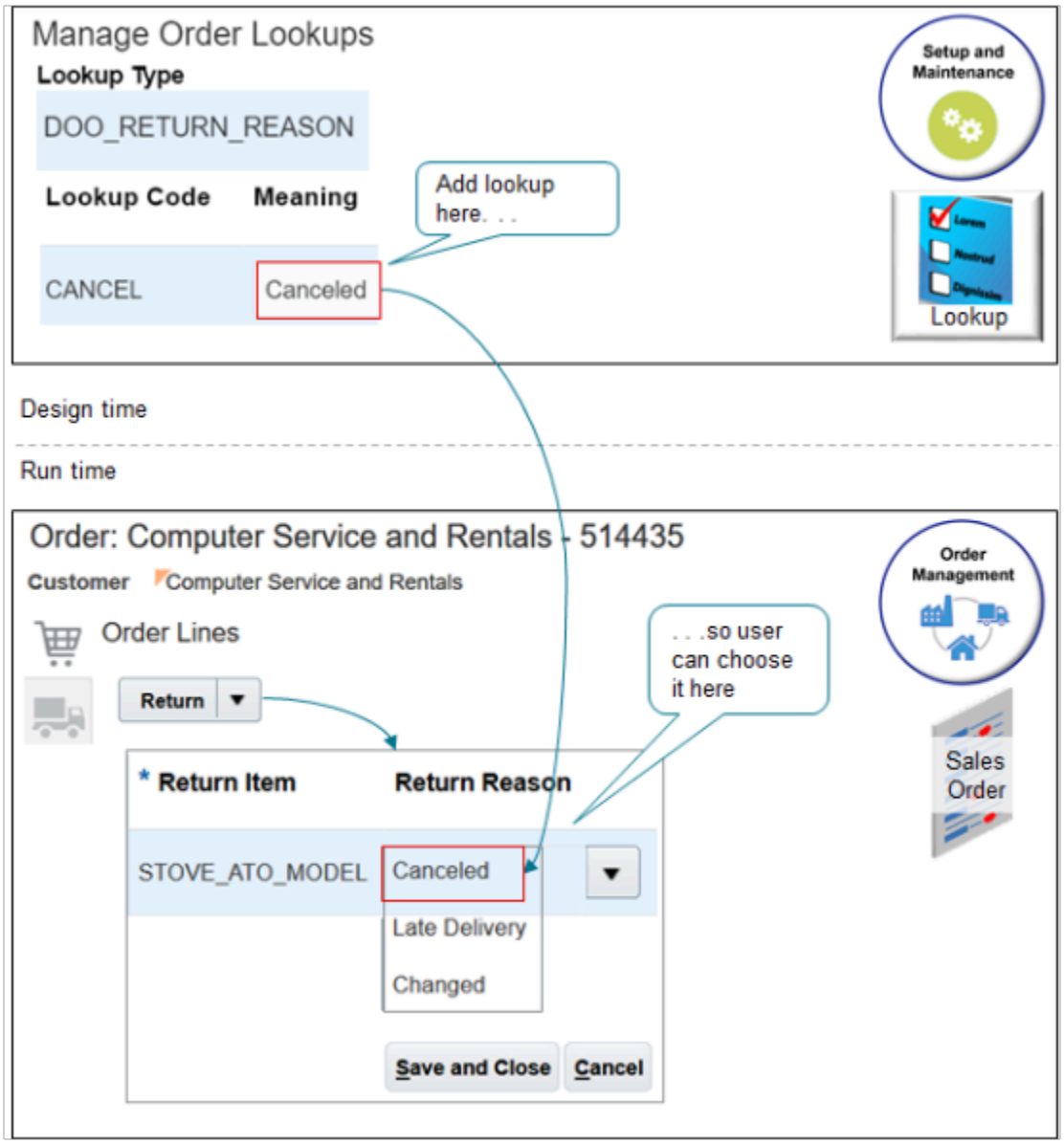
```
<ns1:createOrders xmlns:ns1="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/types/">
<ns1:request xmlns:ns2="http://xmlns.oracle.com/apps/scm/fom/importOrders/orderImportService/">
<ns2:BatchName/>
<ns2:Order>
<ns2:SourceTransactionIdentifier>AP_RETURN_ORDER_01</ns2:SourceTransactionIdentifier>
<ns2:SourceTransactionSystem>GPR</ns2:SourceTransactionSystem>
<ns2:SourceTransactionNumber>AP_RETURN_ORDER_01</ns2:SourceTransactionNumber>
<ns2:TransactionOn>2019-1-20T06:08:52</ns2:TransactionOn>
<ns2:SourceTransactionLineIdentifier>101</ns2:SourceTransactionLineIdentifier>
<ns2:SourceTransactionScheduleIdentifier>101</ns2:SourceTransactionScheduleIdentifier>
<ns2:SourceTransactionLineNumber>1</ns2:SourceTransactionLineNumber>
<ns2:SourceTransactionScheduleNumber>1</ns2:SourceTransactionScheduleNumber>
<ns2:ProductNumber>AS54888</ns2:ProductNumber>
<!-- TransactionCategoryCode can use one of two values. Use ORDER for regular order line. Use RETURN
for return order line. If RETURN, you must provide the reference to original order and line under
LineDocumentReference-->
<ns2:TransactionCategoryCode>RETURN</ns2:TransactionCategoryCode>
<!-- ReturnReasonCode and Return Reason are optional for a return order line. -->
<ns2:ReturnReasonCode>ORA_QTY_CHANGE</ns2:ReturnReasonCode>
<ns2:OrigSysDocumentReference>ORIGSYS</ns2:OrigSysDocumentReference>
<ns2:OrigSysDocumentLineReference>ORIGSYSLINE</ns2:OrigSysDocumentLineReference>

<!Send the Line Document Reference only for a return line. This entity references the original order line
for the item you're returning.-->
<ns2:DocumentReference>
<ns2:DocumentReferenceType>ORIGINAL_SALES_ORDER</ns2:DocumentReferenceType>
<!-- SourceOrderId for original order -->
<ns2:DocumentIdentifier>ORIG_SALES_ORDER_01</ns2:DocumentIdentifier>
<!-- Source Order System for original order identifier-->
<ns2:DocumentAdditionalIdentifier>GPR</ns2:DocumentAdditionalIdentifier>
<!-- SourceOrderNumber of original order -->
<ns2:DocumentNumber>ORIG_SALES_ORDER_01</ns2:DocumentNumber>
<ns2:DocumentAdditionalNumber/>
<!-- SourceLineId for originl order -->
<ns2:DocumentLineIdentifier>101</ns2:DocumentLineIdentifier>
<ns2:DocumentAdditionalLineIdentifier/>
<!-- SourceLineNumber for original order -->
<ns2:DocumentLineNumber>1</ns2:DocumentLineNumber>
<ns2:DocumentAdditionalLineNumber/>
<ns2:DocumentAdditionalSubLineIdentifier/>
<ns2:DocumentSubLineNumber/>
<ns2:DocumentAdditionalSubLineNumber/>
</ns2:DocumentReference>
</ns2:Line>
</ns2:Order>
</ns1:request>
</ns1:createOrders>
</soap:Body>
</soap:Envelope>
```

Related Topics

- Constrain Return Orders
- Import Return Orders
- Opt Into Features in Order Management
- Example Web Service Payloads That Integrate Order Management
- Return Items Without Original Sales Order

Create a Return Reason

Create a return reason so the Order Entry Specialist can select it when returning an order line.



Assume you need to add a reason that the Order Entry Specialist can select to indicate that the customer feels the description that you have for the item on your website isn't accurate.

Summary of the Set Up

1. Create lookup.
2. Collect data.
3. Test your set up.

This topic uses example values. You might need different values, depending on your business requirements.

Create Lookup

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups
2. On the Manage Order Lookups page, enter the value, then click **Search**.

Attribute	Value
Lookup Type	DOO_RETURN_REASON

3. In the Lookup Codes area, click **Actions > New** then set the values.

Attribute	Value
Lookup Code	NOT_ACCURATE
Display Sequence	1
Start Date	The current date.
End Date	Leave empty.
Meaning	Item's Description Isn't Accurate
Description	The item's description on the website isn't accurate.

4. Click **Save and Close**.

You can't disable or change the End Date on predefined lookup codes that have an ORA_ prefix, such as ORA_NOT_ORDERED. Order Management needs these codes to maintain the integrity of the data model.

Collect Data

1. Go to the Plan Inputs work area.

   Don't use the Plan Inputs task that's available in the Setup and Maintenance work area. Use the Plan Inputs work area instead.
2. In the Plan Inputs work area, click **Tasks > Collect Planning Data**.
3. In the Collect Planning Data dialog, set your source system, then move reference entities to selected entities.

- Order Orchestration Reference Objects

4. Click **Submit**.

For details, see *Collect Planning Data for Order Management*.

Test Your Set Up

1. Go to the Order Management work area, then click **Tasks > Manage Orders**.
2. Search for a sales order.

Attribute	Value
Open	No
Status	Closed

You can only return a sales order that isn't open and that's closed, or an order line that Order Management has shipped but not closed.

3. In the search results, in the Order column, click the **sales order** you must return.
4. On the Order page, in the Order Lines area, select the order line you must return, then click **Return**.
5. In the Return Items dialog, verify that you can set the value.

Attribute	Value
Return Reason	Item's Description Isn't Accurate

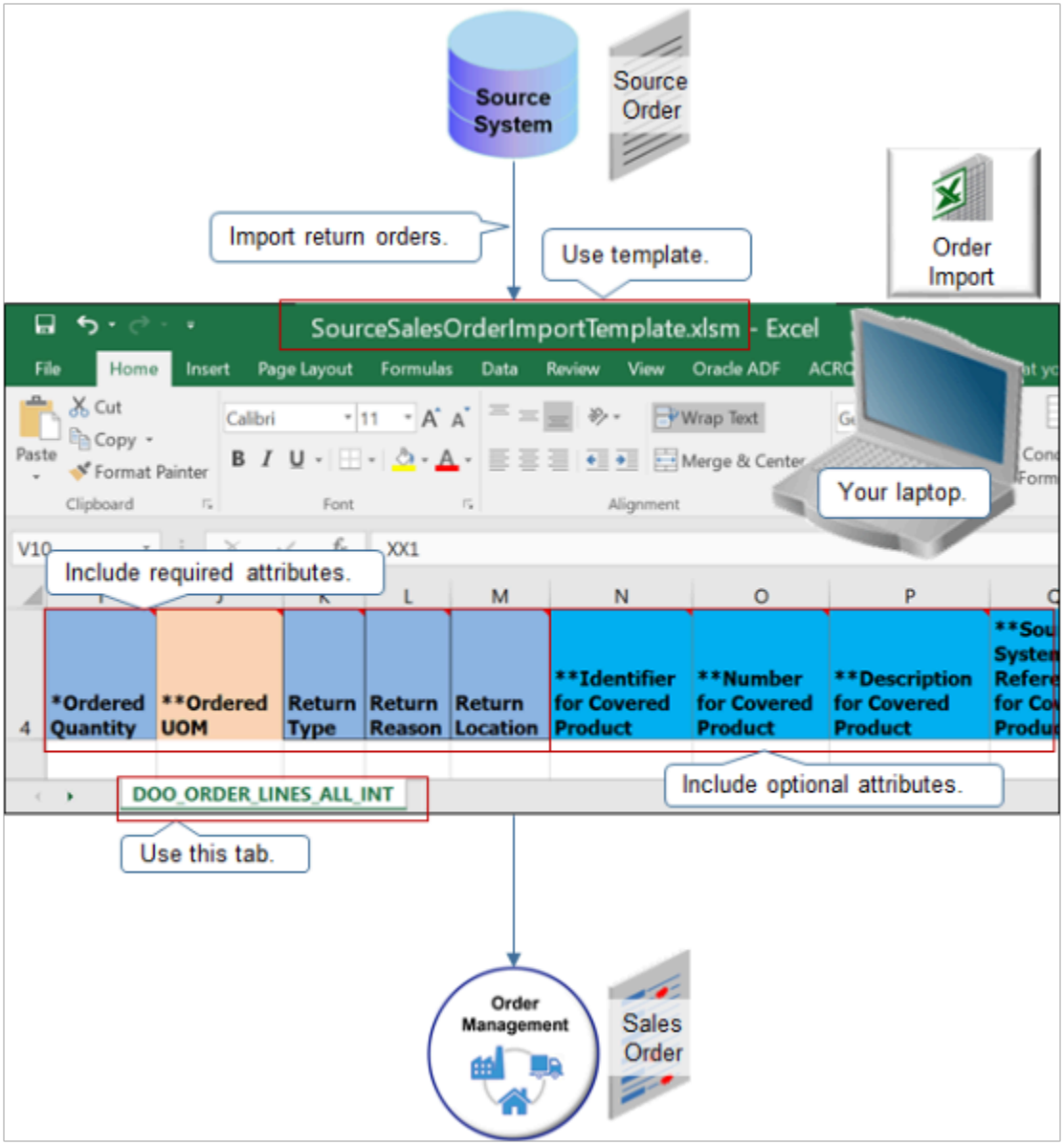
Related Topics

- Overview of Collecting Promising Data for Order Management

Import Return Orders

Import a source order that returns an item.

Here's how you do it.



Use a technology to import the return.

- Order import template. The template is SourceSalesOrderImportTemplate.xlsm. Many of the return attributes are on tab DOO_ORDER_LINES_ALL_INT.
- Web service in Oracle Application Development Framework (ADF).
- Web service in Oracle Service-Oriented Architecture (SOA).

Required Attributes

Here are the attributes that your import must include.

Attribute	Description
Attribute that identifies the item.	Select from the group. For example, Product Number.

Attribute	Description
Ordered Quantity	Make sure quantity is greater than 0 for a new source order. You can use 0 for a revise order. If your set up. • **Allows decimals**. Make sure the number of digits after the decimal point doesn't exceed the maximum number of digits allowed according to profile Quantity Decimal Precision. For details, see *Control Decimal Precision*. • **Doesn't allow decimals**. Make sure the source order doesn't include decimal values.
Ordered UOM	Make sure the unit of measure is appropriate for the item. For example, Quart is appropriate for a liquid, but Amperage isn't because amperage measures electrical current. If the item returns a coverage, then make sure the unit of measure is appropriate for the covered item.
Return Type	Identifies type of order line. Use a value from the Return Order Line Types lookup. If you use the order import template, then use the Transaction Line Type Code attribute in column AB of tab DOO_ORDER_LINES_ALL_INT.
Return Location	If Return Type is `Return for credit and return the item`, then the import requires Return Location. If you use the order import template, then use the Requested Fulfillment Organization Identifier attribute in column N of tab DOO_ORDER_LINES_ALL_INT.
Return Reason	If you use the order import template, then use the Return Reason attribute in the DJ column and the DK column of the DOO_ORDER_LINES_ALL_INT tab.

Coverage

If you cancel a coverage, then include the Covered Item attribute, the Source System Reference for Covered Product attribute, and at least one of these attributes.

- Identifier for Covered Product
- Number for Covered Product
- Description for Covered Product
- Identifier for Covered Customer Product
- Number for Covered Customer Product
- Description for Covered Customer Product

Cancel a Service

A service can be a coverage and subscription. For details, see *Import and Update Source Orders That Include Coverages and Subscriptions*.

Here are the attributes you can use when you cancel a service.

Attribute	Description
Service Duration, Service Duration Code, or Service Duration UOM	If Service Duration Type is. • Fixed or variable, include Service Duration, and include Service Duration Code or Service Duration UOM. • Open ended, and if canceling coverage, then Service Duration Code or Service Duration UOM is optional. You must set Period for a coverage item, or if you set Service Duration. For details, see *Set Up Coverages for Sales Orders*. If you include Service Duration. • You must include Service Duration Code or Service Duration UOM. • Make sure decimal precision doesn't exceed 3.

Note

- If your source system already priced the source order, and if source order includes Duration, then you must also include the extended amount for duration.
- If the import includes a coverage, then you must also include the covered item.

Not Allowed

Make sure your import doesn't:

- Include a configured item.
- Include recurring billing. The source order must specify one time billing.
- Modify a return item and its covered item or add or remove the original sales order revision.

Related Topics

- Constrain Return Orders
- Allow Users to Return Items Without the Original Sales Order
- Return Items Without Original Sales Order

Allow Users to Return Items Without the Original Sales Order

You can return an item or cancel a service without referencing the sales order that originally ordered the item. You can administer this feature.

An unreferenced return is a sales order that includes a return line that doesn't reference the sales order that your customer used to purchase the item.

Examples

Your Customer Returns a Whole Bunch of Orders

Vision Corporation is a wholesaler who sold 8,000 items in 475 sales orders to retailer Fantastic Laptops.

- Each order included more than 100 order lines.

- Vision shipped and closed most of these orders.
- Each order used different prices.

One year later, Fantastic Laptops returns 50 items in a large container. To simplify order entry, billing, and through agreement with Fantastic Laptops, Vision Corporation credits all items at the same price. The Order Entry Specialist uses the Order Management work area to create a single unreferenced return order for all the returned items. This approach allows the Order Entry Specialist to complete the return without having to spend many hours looking up the original sales order for each item.

Your Customer Sends Returns Without the Original Sales Order

First Software maintains over 50 outlets in a fast-paced environment. They sell to an established set of clients, and maintain a contractual, working relationship with each client. Each outlet receives return items that they sold, and also returns items that other outlets sold. However, personnel who work at the outlets don't have time to look up the original sales order for each return.

First Software trusts their clients and don't feel its necessary to confirm the original sales order. At the end of the week, one person at each location enters all returns they received into a single return order that doesn't reference any of the original sales orders.

You Accept Competitor's Orders

Computer Service and Rentals is a new company who sells laptops in a retail marketplace. They are interested in gaining market share, so they accept items that their customer purchased from a competitor as part of a promotion to sell new laptops. Computer Service and Rentals enters these items as unreferenced returns, then sells them later as refurbished.

You Didn't Migrate Data During an Upgrade

Green Corporation sells magazine subscriptions. Customers call the call center to cancel subscriptions. Green Corporation recently upgraded to Order Management, but didn't migrate order data from their legacy system to Order Management.

The Order Entry Specialist creates an unreferenced return to cancel the subscription and credit the remaining part of the subscription that the customer already paid but didn't use. The Order Entry Specialist sets details on the order line, such as item, quantity, return type, and then uses attribute Duration and attribute Period to specify details about the return. Order Management calculates the price when the Order Entry Specialist adds the return to the order.

The Item Doesn't Include Identifying Markings

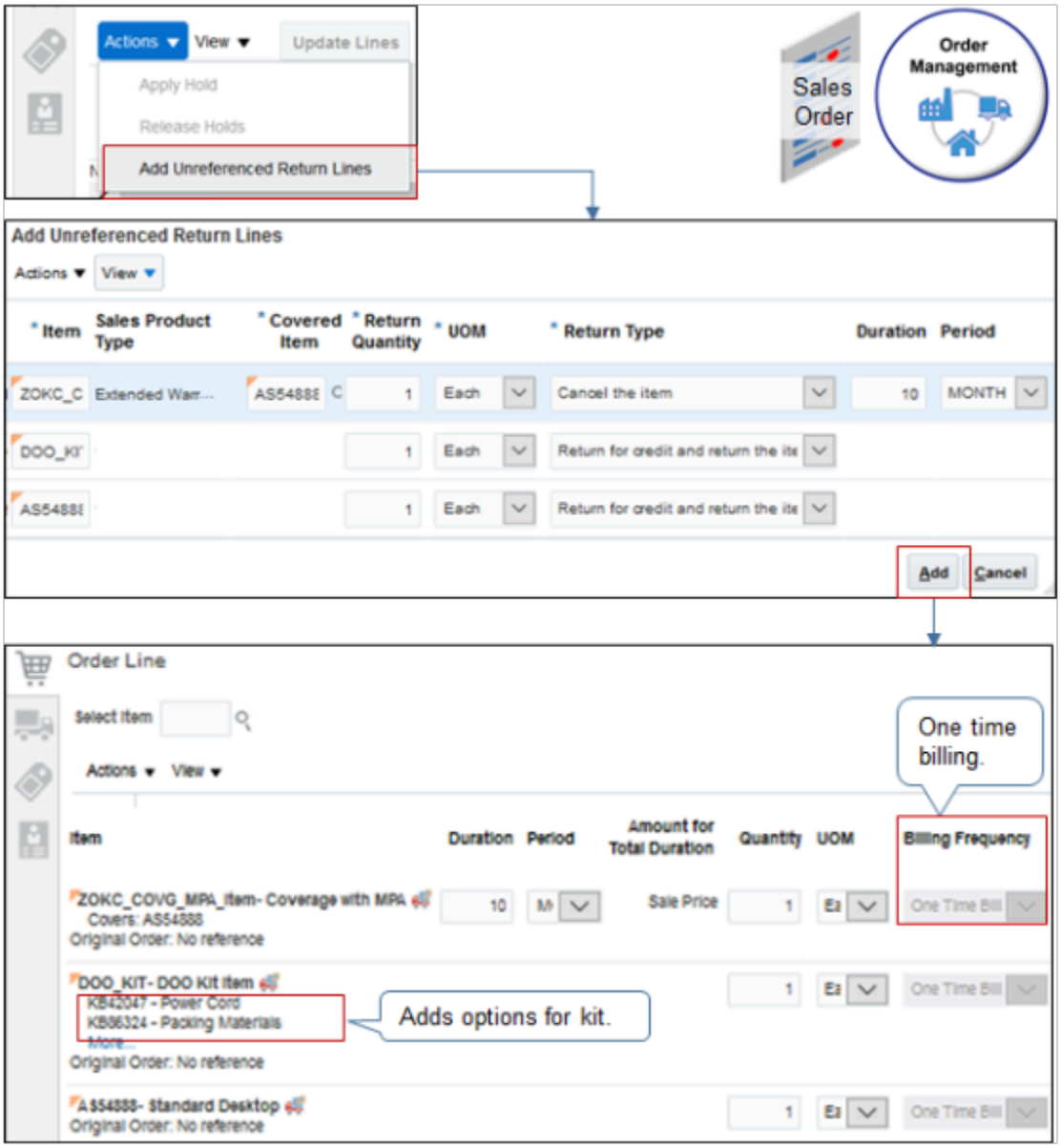
Computer Service and Rentals is a retailer who sells directly to the public. Their customer returns an item but doesn't have the original receipt, and the item doesn't include a serial number or other marking that the Order Entry Specialist can use to look up the item when creating the return. The Order Entry Specialist uses the Add Unreferenced Return Lines action to add the item.

Small Incorporated is a retailer who sells the Long Life Tea Set, which is a pick-to-order, configured item that includes configure options, such as steeping pot, pouring vessel, and set of tea cups. Customers return items but no longer have the receipt, and the item doesn't include an identifying marking that the Order Entry Specialist can use to look up the item when creating the return. The Order Management work area doesn't allow the Order Entry Specialist to return a configured item. Instead, the Order Entry Specialist creates a return line for each configure option.

If you have an unreferenced return, and if you're returning a configured item that's:

- **Pick-to-order**. You can return one or more individual configure options or the entire item.
- **Assemble-to-order**. You can't return any configure option. You can't return the entire item.

How It Works



Here's what you do.

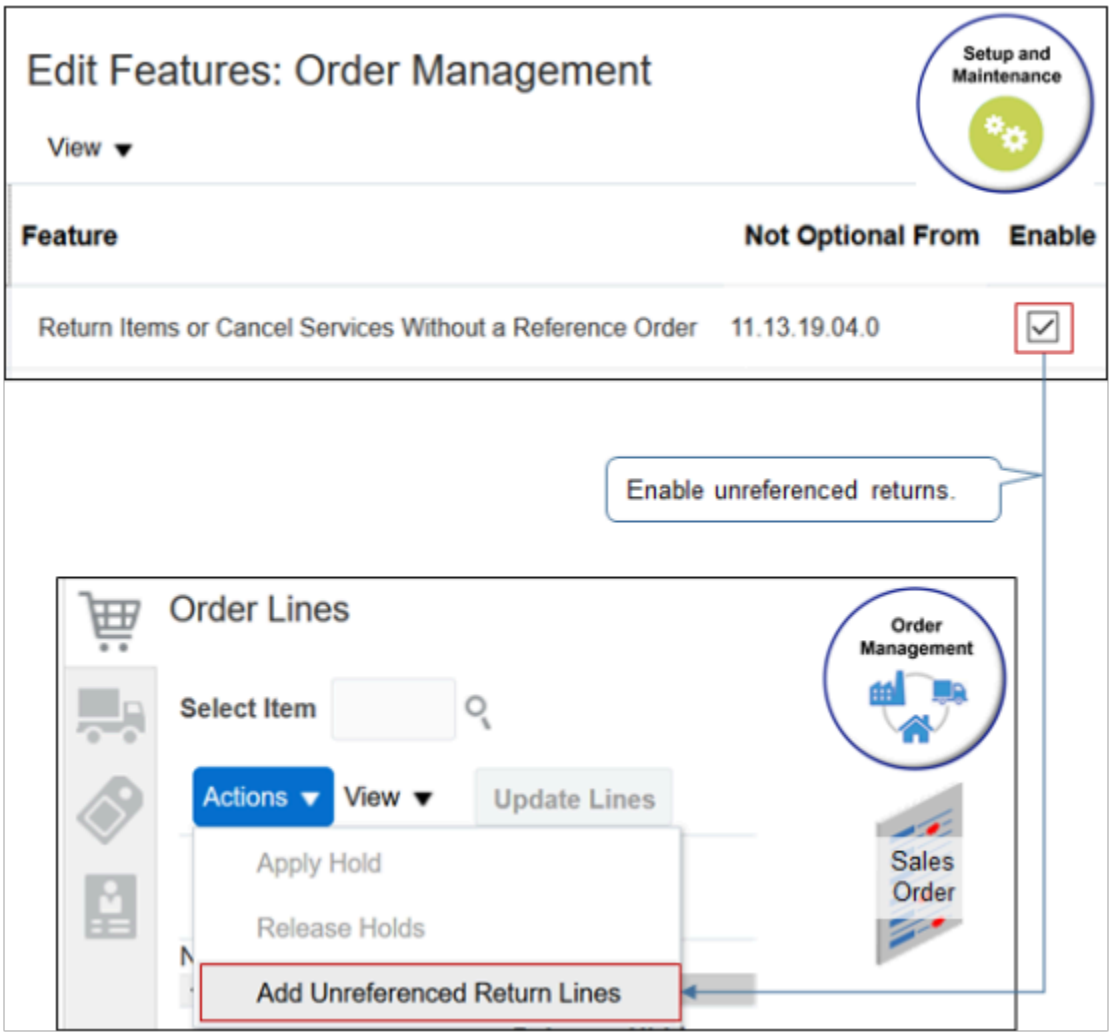
- Fill in attributes on the order header, then, in the Order Lines area, click Add Unreferenced Return Lines.
- Add your unreferenced return lines, then click Add to add them to the sales order.

Here's what Order Management does.

- Uses attribute values from the order header to set default values for attributes on return lines.
- Runs pretransformation rules to set the default values.
- Prices the return according to current system date.
- Sets Billing Frequency to One Time Billing and credits the entire amount as a single credit.

- Adds included item for each kit according to the product structure that you set up in the Product Information Management work area.

Set Up



If you're upgrading, then here's what it does, depending on whether you extended your pricing algorithms.

- **Haven't extended**. Promotes all algorithms to the latest version.
- **Have extended**. Reconciles algorithm extensions, then promotes all algorithms.

How Pricing Works

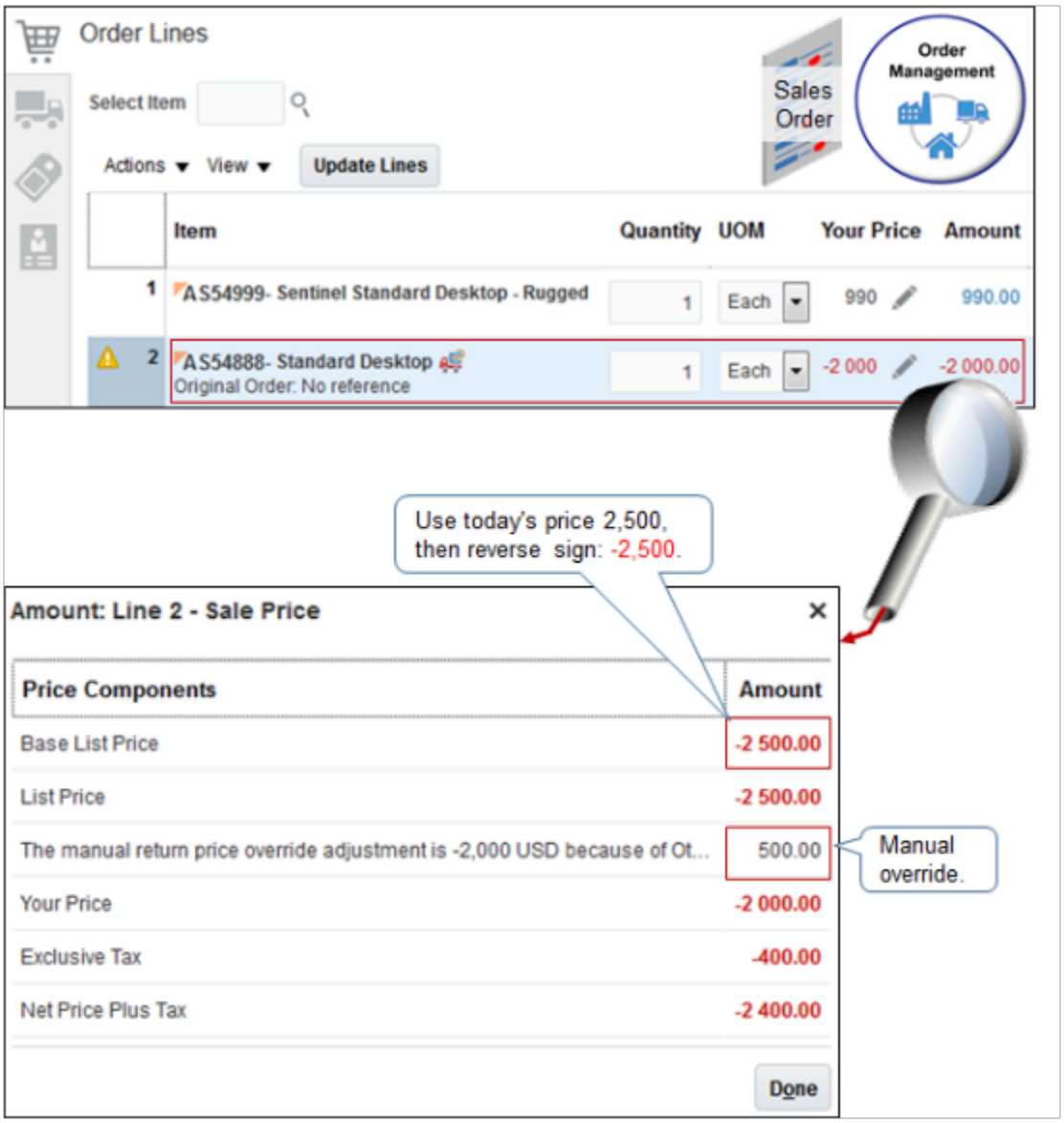
Pricing Administration comes predefined to calculate price for each unreferenced return.

- Determines whether the return line does or doesn't include a reference to the original sales order.
- Calculates price like it normally calculates a sales order line, then adds a negative sign.
- Calculates price according to the pricing set up that exists on the return date or cancel date. For example, it applies the discounts and charges that existed on the item as of the return date or cancel date.
- Calculates a one time charge or recurring charge.

- Calculates the pricing adjustment and discount.
- If Pricing can't determine the charge, then it applies a zero charge.
- Allows the Order Entry Specialist to manually adjust the price as an override on the return line.
- Prices the cancel for a coverage or subscription, including pricing only part of the coverage or subscription period.

You don't need to do any set up to implement this predefined behavior.

For example:



Note

- Pricing determines that the list price for the AS54888 item is $2500.00 according to the price that exists on the date and time when the user added the order line.
- Pricing applies the negative sign to get -2500.00.

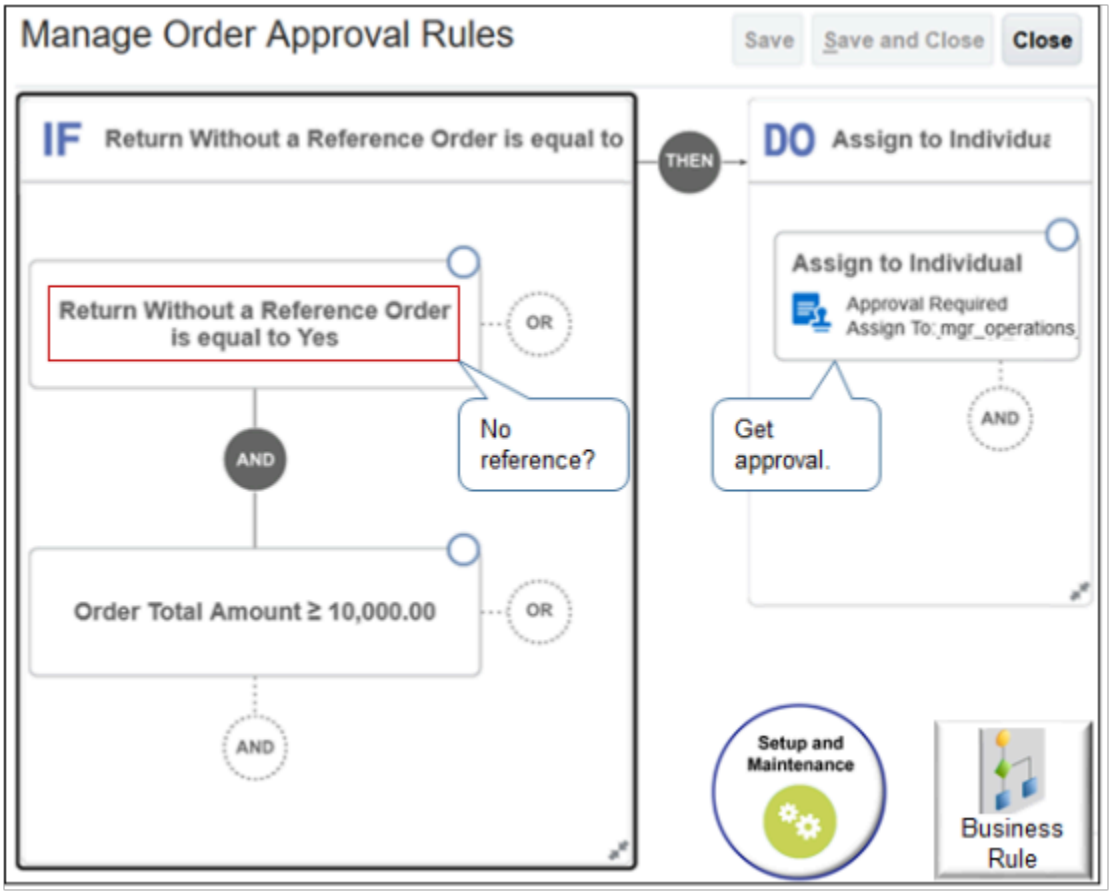
- The user overrides the price to -$2000.00. For example, assume your company policy requires that you reduce the refund by $500 when the customer returns the AS54888 without the original sales order.

You can modify the predefined behavior for an unreferenced return.

- You can modify your pricing process to calculate a return amount that's different than the price as of the return or cancel date, such as including a restocking fee.
- You can apply a manual price adjustment on the charge.
- The predefined behavior for a referenced return or an unreferenced return doesn't support pricing from a returns price list.

Create a Business Rule

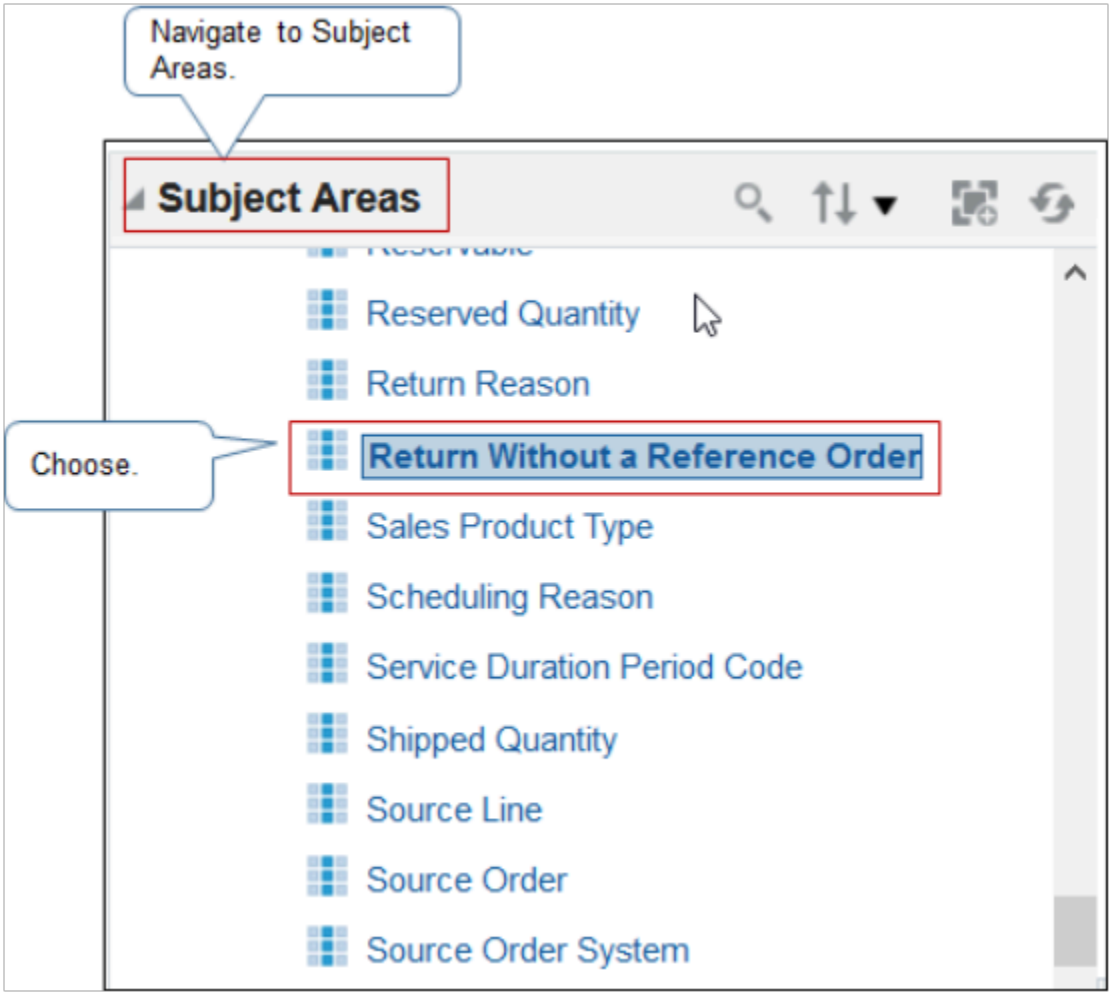
You can write a business rule, such as an approval rule, that processes an unreferenced return.



Here's the logic that this example implements.

```
If attribute Return Without a Reference Order equals yes, and if attribute Order Total Amount on sales order
 is equal to or greater than $10,000, then assign sales order to the mgr_operations position for approval.
```

Get a Report



In the Reports and Analytics work area, expand the Order Management, Fulfillment Lines Real Time subject area, expand Fulfillment Lines General Details, then notice the attributes you can use for a fulfillment line. To create a report that includes unreferenced return lines, add a filter.

- Return Without a Reference Order = Y

For details, see *Use Reports and Analytics with Order Management*.

Related Topics

- Constrain Return Orders
- Import Return Orders
- Opt Into Features in Order Management
- Return Items Without Original Sales Order

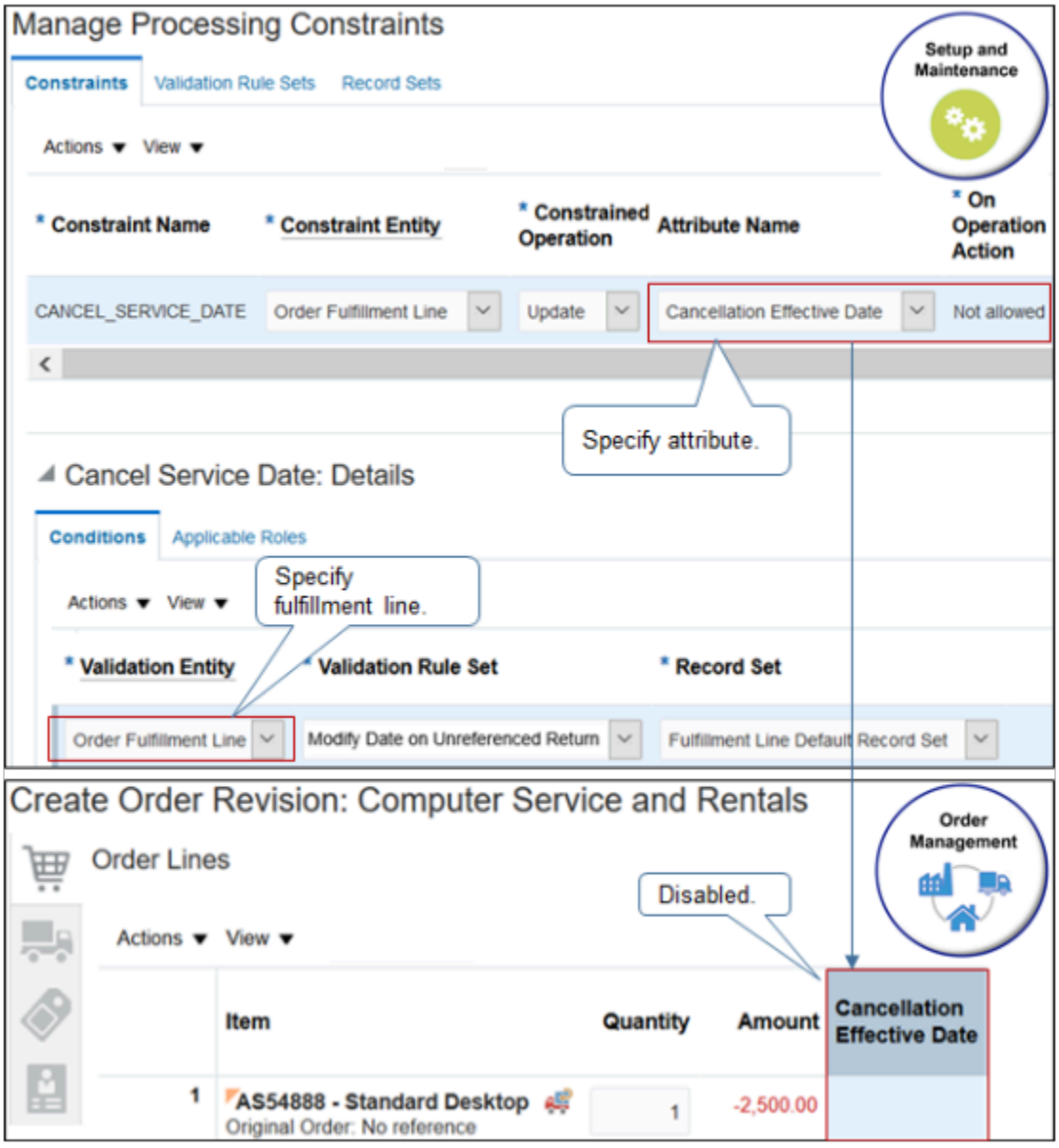
Constrain Return Orders

Set up a processing constraint that constrains changes the Order Entry Specialist makes to a return that doesn't reference the original sales order.

For example, Order Management uses the Cancellation Effective Date attribute to price a return that doesn't reference the original order. Here's your constraint.

- If the return doesn't reference the original sales order, then don't allow the user to modify Cancellation Effective Date.

Here's your set up.



For details, see *Manage Processing Constraints*.

Use an order management extension to implement a more specialized constraint. For example, if its been 15 days or more since Order Management sent the sales order to invoicing, then write an extension that prevents the user from creating a return material authorization. Use the On Save extension point. For details, see *Points Where You Can Run Order Management Extensions*.

This topic uses example values. You might need different values, depending on your business requirements.

Try it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Processing Constraints
2. Add the validation rule set.
    - On the Manage Processing Constraints page, click **Validation Rule Set**, then add a rule set.

Attribute	Value
Attribute	Value
Name	Modify Date on Unreferenced Return
Description	If the return doesn't reference the original sales order, then don't allow user to modify date.
Short Name	UNREF
Validation Type	Table
Entity	Order Fulfillment Line

    - In the Details area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Name	Cancellation Effective Date
Validation Operation	Is Not Null

- Click **Generate Packages**, notice the Confirmation dialog displays your request ID, then wait a few minutes for the request to finish.

  For example:

  ```
  The concurrent request to generate constraints validation packages was submitted. Request ID:
   10650.
  ```

3. Add the constraint.

   - Click **Constraints > Actions > Add Row**, then set the values.

Attribute	Value
Constraint Name	CANCEL_SERVICE_DATE
Display Name	Cancel Service Date
Constraint Entity	Order Fulfillment Line
Constrained Operation	Update
Attribute Name	Cancellation Effective Date
Enabled	Contains a check mark

   - In the Conditions area, add the condition.

Attribute	Value
Group Number	1
Validation Entity	Order Fulfillment Line
Validation Rule Set	Don't allow date change on unreferenced return
Scope	Any
Record Set	Fulfillment Line Default Record Set
Message	You must not modify Cancellation Effective Date when you return a sales order without the original order because Order Management uses this date to price the return.

Attribute	Value

- Click **Generate Packages** and wait a few minutes.

Test Your Set Up

1. Navigate to the Order Management work area and create a new sales order.

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Bill-to Account	1006

2. In the Order Lines area, click **Actions > Create Unreferenced Return Lines**, then set the values.

Attribute	Value
Item	Add a subscription that uses a fixed Service Duration Type. For details, see *Set Up Coverages for Sales Orders*. For this example, assume your implementation includes fixed subscription QP_SUBS_ITEM7-FIXED.
Return Quantity	1
UOM	Each
Return Type	Return for Credit and Return the Item
Return Location	Vision Operations

3. Click **Submit**.
4. If the order line status is Not Started, then wait a moment, and click **Refresh**. Repeat until status is Awaiting Receiving.
5. Click **Actions > Create Revision**.
6. On the order line, modify the value in Cancellation Effective Date, then click **Submit**.

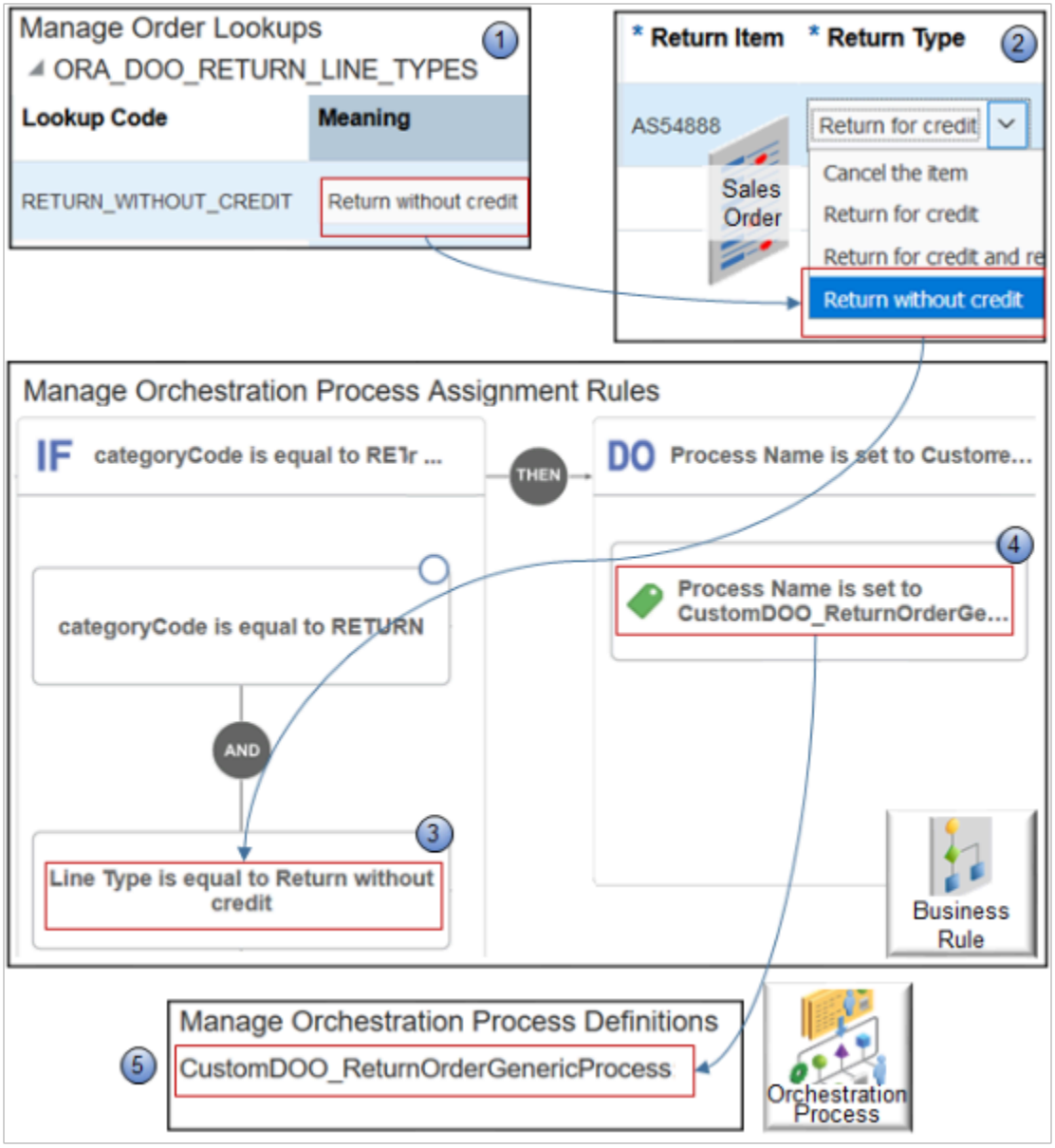
Related Topics

- Allow Users to Return Items Without the Original Sales Order
- Import Return Orders
- Overview of Creating Order Management Extensions
- Return Items Without Original Sales Order

Return Sales Orders Without Credit Memo

Set up Order Management so your users can return a sales order without creating a credit memo.

You might not want Order Management to create a credit memo, such as if you allow customers to return a competitor's order. Here's the flow you will set up.



Note

1. You create a lookup that allows your users to specify whether its a return without credit.
2. The user searches for and views a closed sales order, clicks Return in the Order Lines area, then uses the Return Items dialog to set Return Type.
3. You create an assignment rule.
    - If CategoryCode is equal to Return, and.
    - If Line Type is equal to Return Without Credit
4. You create an action in your rule.
    - Assign orchestration process CustomDOO_ReturnOrderGenericProcess.
5. You create CustomDOO_ReturnOrderGenericProcess. You copy the predefined ReturnOrderGenericProcess, then remove steps so it doesn't issue a credit memo.

Get more information.

- Get more examples. For details, see *Allow Users to Return Items Without the Original Sales Order*.
- Learn about credit memos. For details, see *Using Receivables Credit to Cash*.

Summary of the Steps

1. Set up the lookup.
2. Create the orchestration process.
3. Create the assignment rule.

Set up the Lookup

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups
2. On the Manage Order Lookups page, in the Search area, enter the value, then click **Search**.

Attribute	Value
Lookup Type	ORA_DOO_RETURN_LINE_TYPES

3. In the Lookup Codes area, click **Actions > New**, set the values, then click **Save and Close**.

Attribute	Value
Lookup Code	RETURN_WITHOUT_CREDIT
Meaning	Return without credit
Description	Return the item but don't create a credit memo.

Create the Orchestration Process

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions

2. On the Manage Orchestration Process Definitions page, search for the value.

Attribute	Value
Process Name	ReturnOrderGenericProcess

3. In the search results, click the **row** that includes ReturnOrderGenericProcess in the Name column, then click **Actions > Duplicate**.
4. On the Edit Orchestration Process Definition page, set the values, then click **Save**.

Attribute	Value
Process Name	CustomDOO_ReturnOrderGenericProcess
Process Display Name	Return Sales Order Without Creating Credit Memo Enter any text that's meaningful to you.

5. In the Process Details area, click **Status Conditions > Fulfillment Line Status Values**, then click **Edit Status Rule Set**.
6. On the Edit Status Rule Set page, delete any rows that contain these values in the Status Value column:
    - Awaiting Billing
    - Billed
    - Partially Received
7. Click **Save and Close**.
8. In the Process Details area, click **Orchestration Process Status Values**, then delete rows that contain these values in the Status Value column:
    - Awaiting Billing
    - Billed
    - Partially Received
9. In the Process Details area, click Step Definition, then delete the rows that contain these values in the Step Name column:
    - Create RMA Invoice
    - Wait for RMA Invoice
10. Click **Save**, then deploy the orchestration process. For details, see *Deploy Orchestration Processes*.

Create the Assignment Rule

Create and publish an assignment rule that assigns your orchestration process.

- Create the If statement.

- `If categoryCode is equal to Return, and Line Type is equal to Return without credit`
- Create the Then statement.

  `Process Name is set to CustomDOO_ReturnOrderGenericProcess`
- Publish and test your rule.

Learn how to create an assignment rule. For details, see *Overview of Using Business Rules With Order Management*.

Related Topics

- Overview of Using Business Rules With Order Management
- Deploy Orchestration Processes
- Overview of Orchestration Processes
- Manage Lookups in Order Management
- Return Items Without Original Sales Order

Don't Refund Lines That You Return to Your Customer

Create a rule that doesn't refund your customer when you return the item to your customer.

You can use these return flows with Order Management.

- Receive, Inspect, Put Away, Deliver. For example, you receive the item from your customer, you inspect it, it passes inspection, and you put it away in your warehouse.
- Receive, Inspect, Return to Customer. For example, you receive the item from your customer, you inspect it, it fails inspection because its damaged due to customer negligence, and you ship it back to your customer.

You can't use the Receive, Inspect, Reject, Put Away flow with Order Management. For example, assume you receive the item from your customer, it fails inspection because its damaged due to customer negligence, and you reject it. You can't put the rejected item away in your warehouse.

Assume you create a return with a quantity of 5 for the AS54888 desktop computer on order line 1. During inspection, you find that all 5 of the computers are damaged due to customer negligence. You can't put away items that don't pass inspection into inventory. Instead, you return the item to your customer and set the delivered quantity to zero.

You also don't want to provide a credit for the quantity that you will return to your customer. Here's how you can modify your orchestration process so it doesn't provide that credit.

1. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Orchestration Process Definitions
2. Search for, then open the orchestration process that you use to process returns.

3. Locate the step that creates the invoice. It will typically have these values.

Attribute	Value
Step Name	Create Invoice
Step Type	Service
Task Type	Invoice
Task	Invoice
Service	Create Billing Lines

4. On the step that you just located, in the Line Selection Criteria column, click **Click for Rule**.
5. Create this rule:

```
Root:
DooSeededOrchestrationRules.DOOHeader
IF
 Header is a DooSeededOrchestrationRules.DOOHeader and
 Fline is a Header/childFLines
 And "Y"equals ignore case Fline.invoiceEnabledFlag
 And "Y"equals ignore case Fline.invoiceableItemFlag
 And the following test is not true
 "TO" equals ignore case Header.sourceDocumentTypeCode
 And the following test is not true
 "INCLUDED" equals ignore case Fline.itemSubTypeCode
 And the following test is true
 "ORDER" equals ignore caseFline.categoryCode or
 the following test is true
 "RETURN" equals ignore case Fline.categoryCode and
 null isn't Fline.rmaDeliveredQty and
 Fline.rmaDeliveredQty more than BigDecimal.ZERO
THEN
```

```
assert new DooSeededOrchestrationRules.Result( resultObjKey:Fline.fulfillLineId )
```

For example:

For details, see *Set Up Orchestration Processes*.

6. Release and deploy your orchestration process. For details, see *Deploy Orchestration Processes*.

9 Configure-to-Order

Overview

Overview of Configure-to-Order

Use configure-to-order to allow your customer to choose options for each component of a configured item.

For example, order a laptop computer with a 1 terabyte hard drive. The computer is an example of a configured item. The hard drive is an example of a configure option. 1 terabyte and 500 megabytes are each values your customer can choose for the option. A configured item can include more than one configure option. A laptop computer can include other options, such as memory, the display, color of the chassis, and so on.

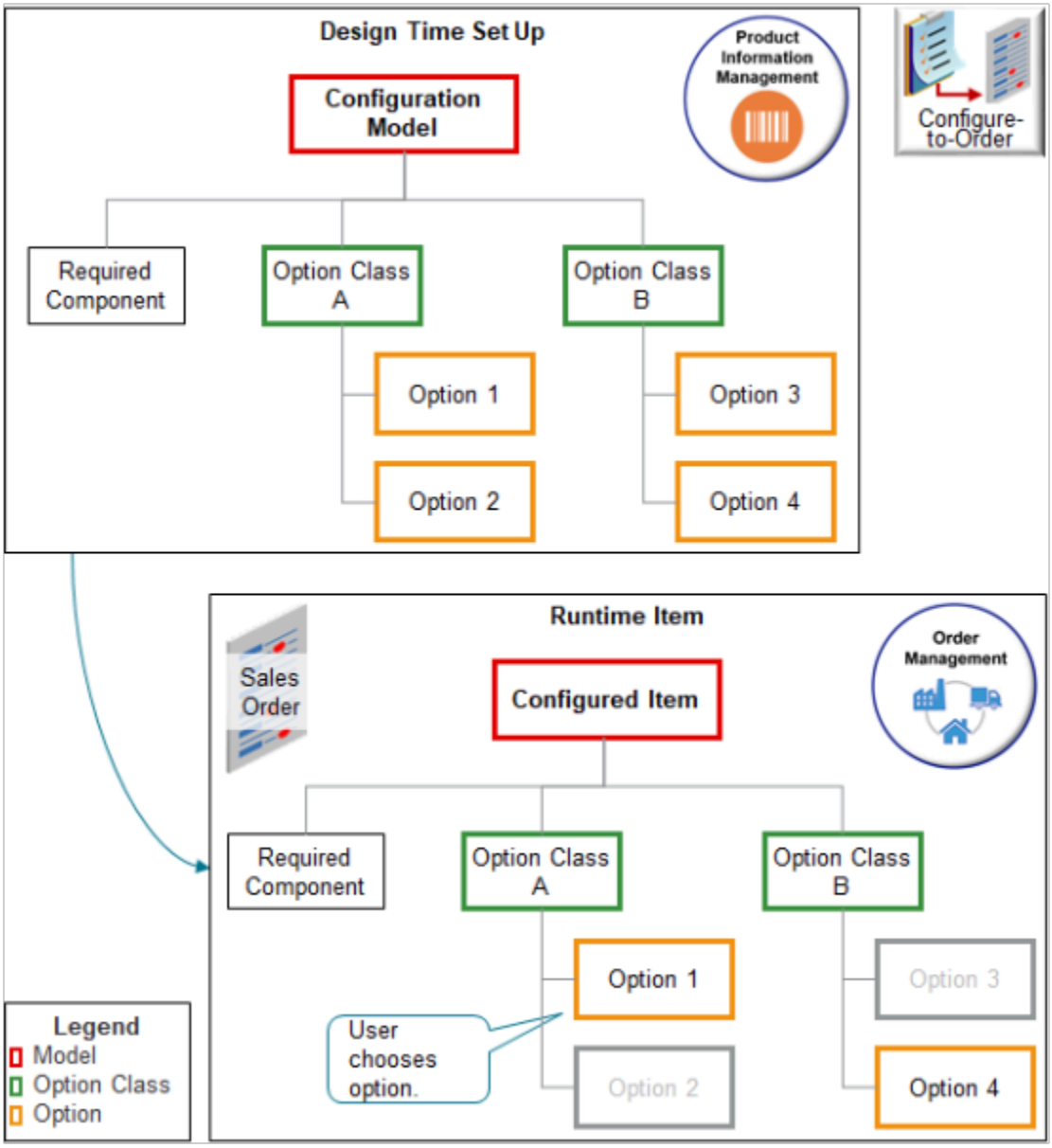
Configure-to-order is the process of ordering and fulfilling a configured item.

Why Should I Use Configure-to-Order?

- Its expensive to build and stock supply for all the different possible combinations of options, store it in inventory, then wait for your customers to order. Instead, use just-in-time manufacturing and other manufacturing processes, such as postponement, to build the item on demand, when and where you customer orders it.
- Some options rarely sell, but when they do sell, they meet an important customer requirement.
- Some items cost a lot of money and are expensive to maintain as on-hand inventory.
- Some items are physically large and not practical to stock in every possible combination.
- Modeling each configured item separately improves handling and helps the warehouse and shop floor to identify and manage the item.
- Improve visibility for the item and on-hand quantities in inventory, promising, and planning.

What's a Configuration Model?

Here's a generic structure of a configuration model.



Here are some important concepts.

Concept	Description
Configuration model	A structure that defines the options your user can choose for each component. You can also specify that a component is required. You define the model during set up. You don't order or build the model at runtime.
Option class	An object you use to organize configure options.
Configure option	Child of an object class. The hard drive, monitor, and mouse are each an example of a configure option. Your user can choose the value for a configure option from a list of available options.

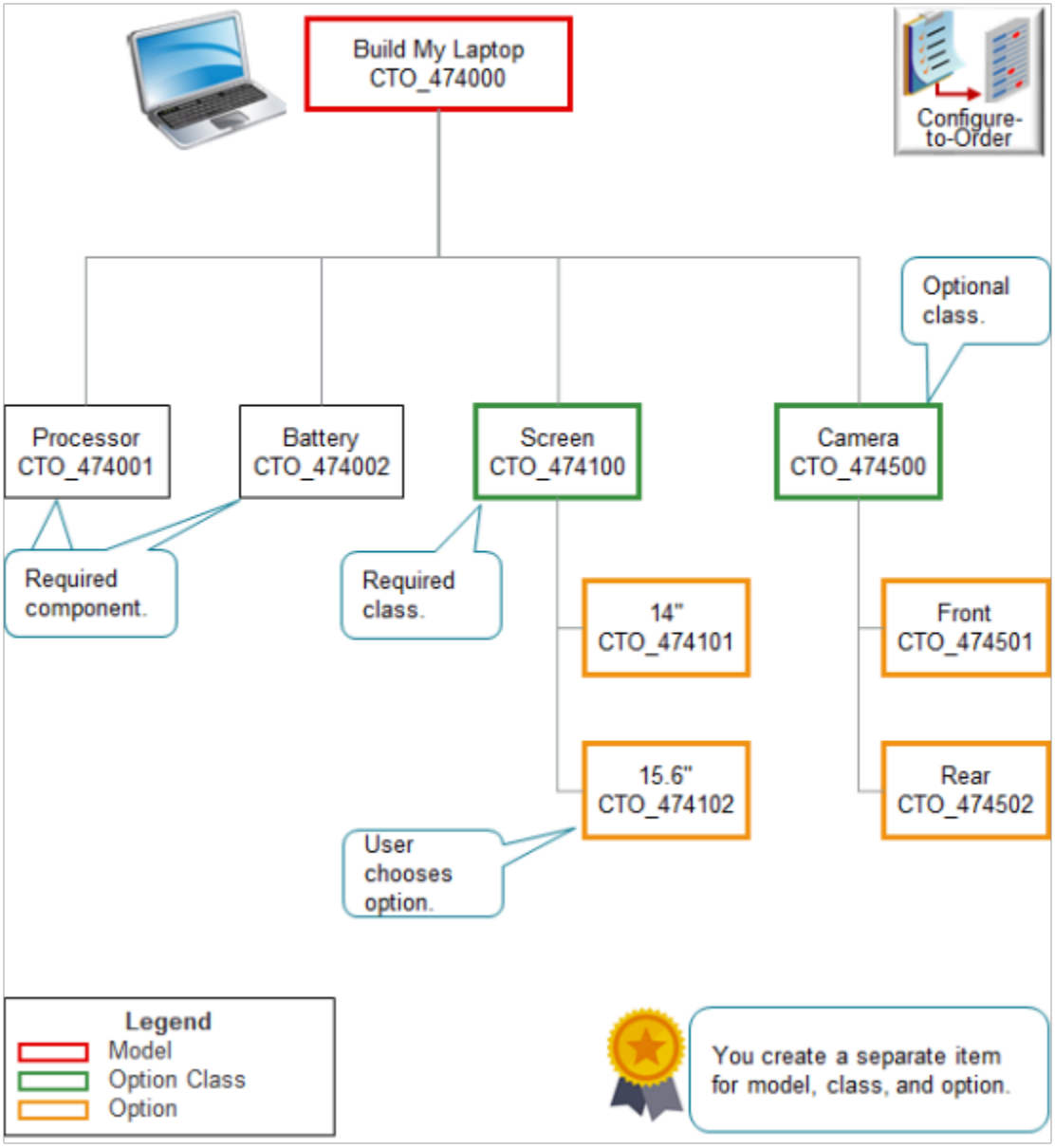
Concept	Description
Configured item	An item that includes one or more configure options that your user chooses. A desktop computer where you choose the hard drive, monitor, and mouse is an example of a configured item. A configured item is the result of the choices that your user makes at runtime when configuring the item.
Required component	Component in the model that the user can't choose. For example, you usually can't choose the cooling fan when you order a laptop computer.

Note

- You create the model, classes, options, and structure in the Product Information Management work area at design time.
- A user chooses options in the Order Management work area in a sales order at runtime.

Example

Here's an example of a configuration model.



Note

- The model, each option class, and each option are separate items that you create in the Product Information Management work area. For example, CTO_474000 is the item name for the model, CTO_474100 is the name of the screen option class, and CTO_474101 is the name of the 14" screen option.
- The processor and battery are required components in this model. You can't order a laptop without them.
- The screen option class is required. You can't order a laptop without a screen, but you can order a 14" screen or 15.6" screen.
- The camera option class is optional. You can order a laptop without a camera. If you add a camera, then you can add a front camera, rear camera, or front and rear camera.
- Your user chooses the options at runtime.

You use the Product Information Management work area to create the structure. For example:

- Create an item for each screen option.
- Create an item for the screen option class, then add the screens to the class.
- Create the model, then add the option class to the model.

Types of Configurations

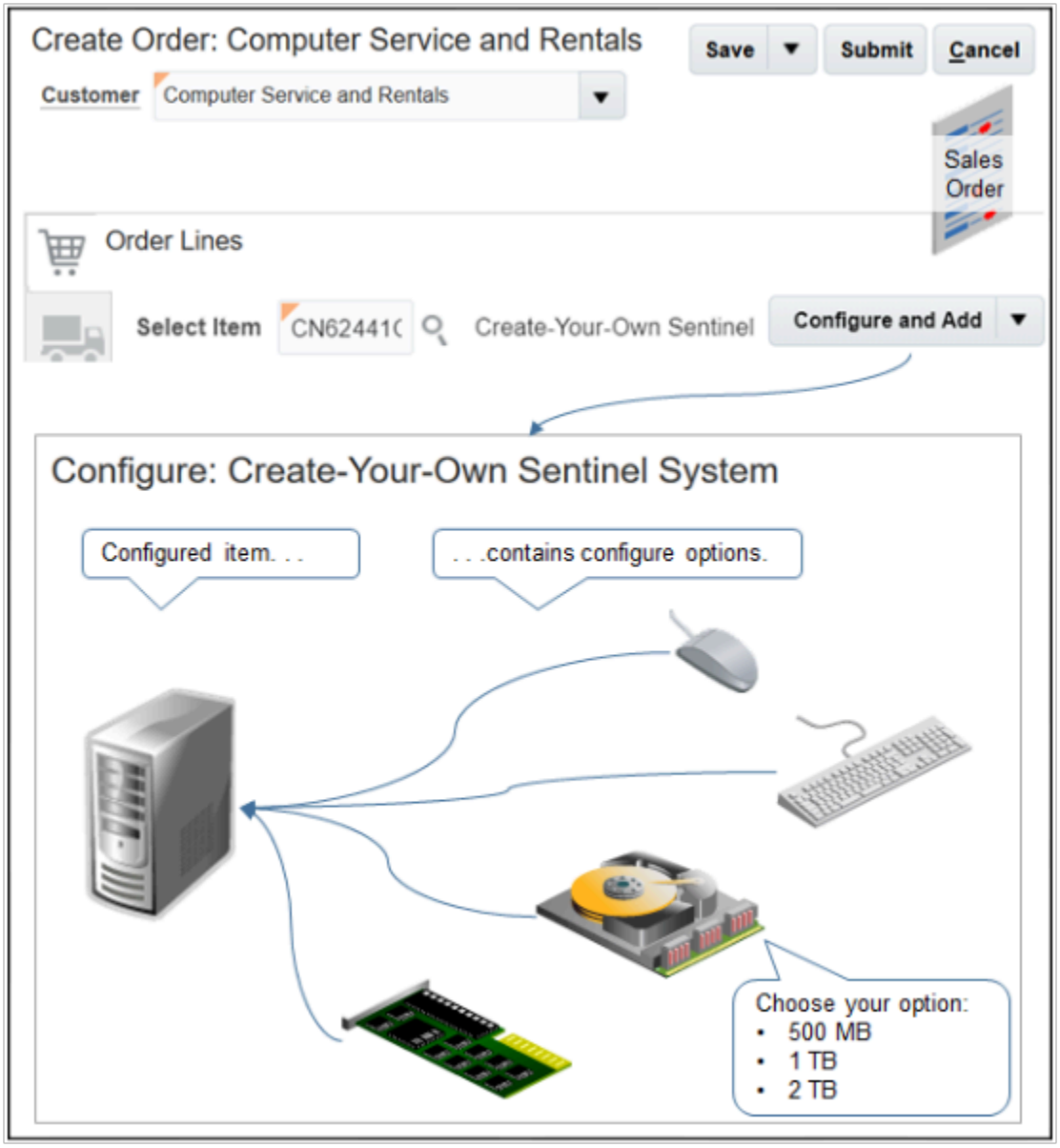
Configuration	Description
Assemble-to-order (ATO)	Item that isn't yet manufactured and isn't ready to ship. You make it or procure it according to the options that your user sets. • A warehouse usually doesn't stock every possible set of options your user might set. Instead, the warehouse receives the work order, then. • Uses a drop-ship flow or back-to-back flow to purchase the configured item from a supplier. • Uses a back-to-back flow to manufacture the item according to the work order.
Pick-to-order (PTO)	Item where you already manufactured the components, or you must make or purchase them. • Can include a back-to-back flow. • Can include drop ship. • Can include back-to-back and drop ship, but you must ship to your customer in one package.
Hybrid	A pick-to-order item that contains at least one assemble-to-order component.

Examples of Using Configure-to-Order

Examine some examples that use configure-to-order.

Simple Example

Here's an example that includes a few configure options.



You go to the Order Management work area, create a sales order, search for a configuration model on the catalog line, then click Configure and Add. Use the Configure page to set configure options. For details, see *Add Configured Items to Order Lines*.

Complex Example

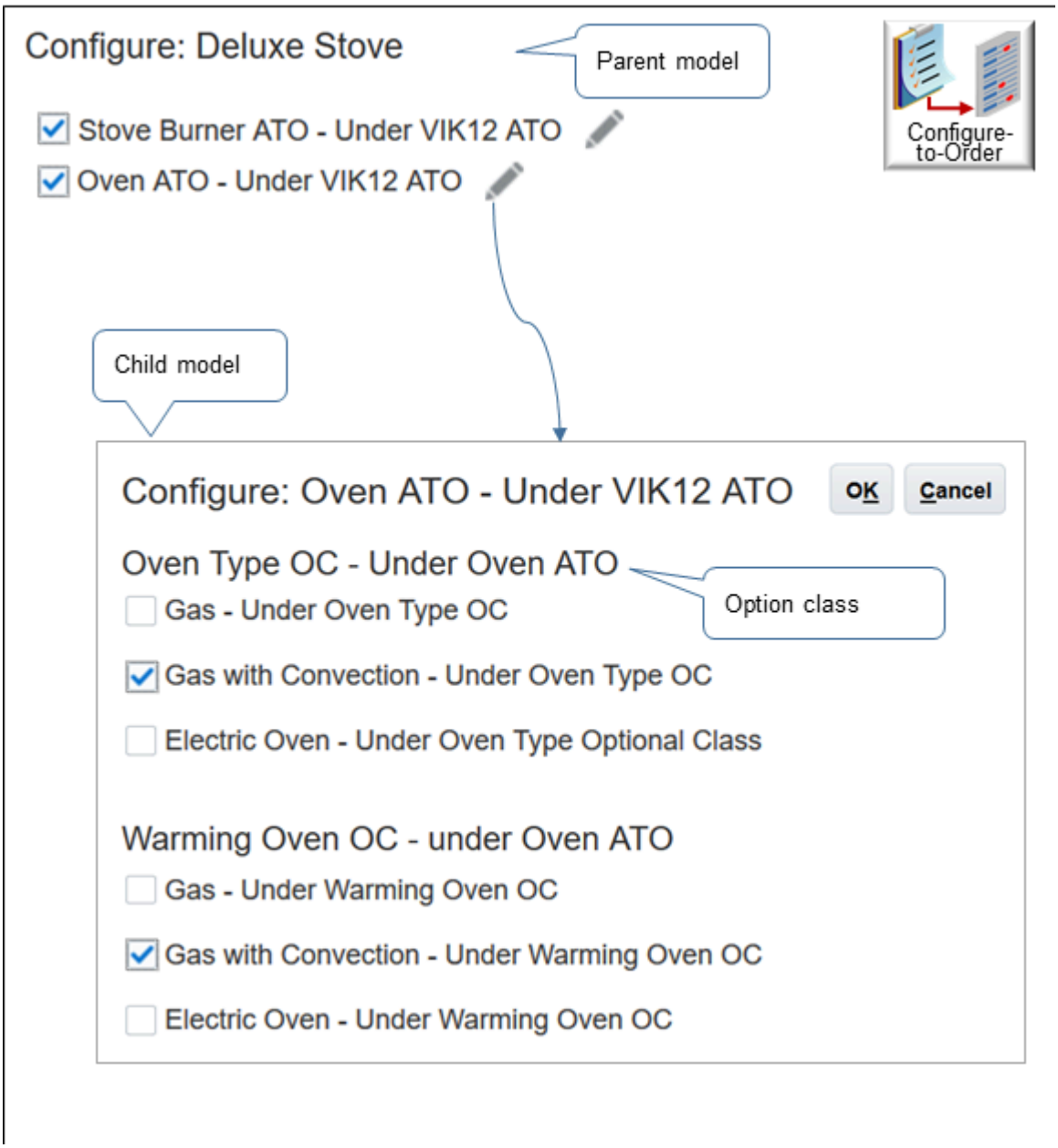
Add complexity to your model. Assume you sell a kitchen stove top with oven. You need to provide a variety of configure options, such as gas, electric, convection, commercial, residential, type of finish, and so on. For this sales order, your customer needs large commercial gas burners for the stove top, a convection gas main oven, a convection gas warning oven, a griddle with commercial burners, large knobs, and stainless steel finish.

Note

- This configuration includes a parent model. The parent references child models.
- Deluxe Stove is the parent model.

- Stove Burner ATO and Oven ATO are each a child model.
- Burner Knobs is an option class.
- Large Knobs and Small Knobs are each a configure option in the Burner Knobs class.
- Finish is an option class.
- Midnight Black Finish, Stainless Steel Finish, and Winter White Finish are each a configure option in the Finish class.
- In this example, the user has chosen options.
    - Stove Burner ATO
    - Oven ATO
    - Large Knobs
    - Stainless Steel Finish
- Click the pencil to configure a child model.

This model includes two different types of ovens. The main oven and a warming oven. The user can choose the type of oven and the type of warmer.

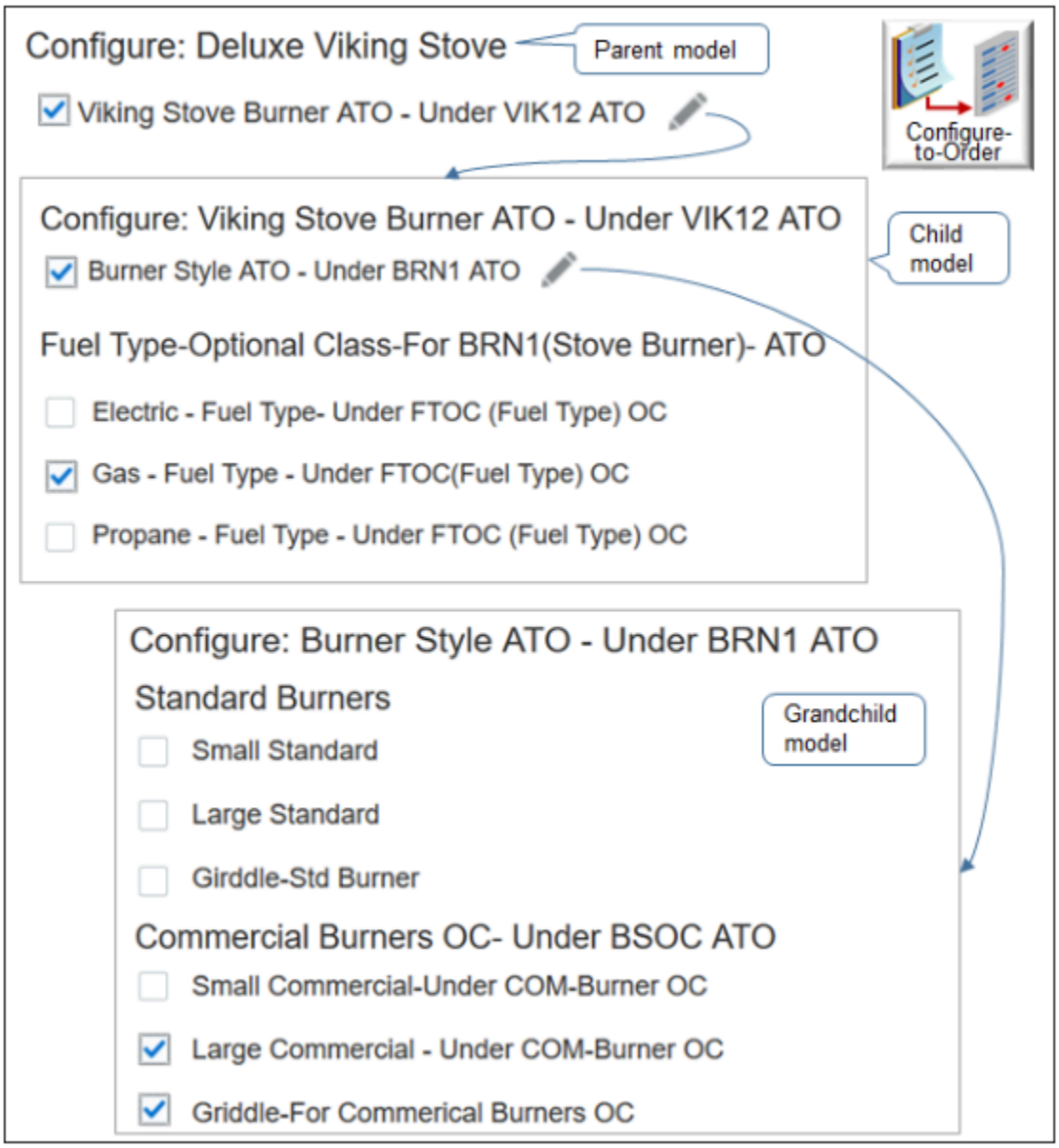


Note

- The Oven ATO model is a child of the parent Deluxe Stove model.

  Note that VIK12 is an abbreviation for Deluxe Stove. You can specify it.
- Oven Type is an option class in the Oven ATO model.
- Gas, Gas with Convection, and Electric Oven are each an option in the Oven Type class.
- Warming Oven is an option class in the Oven ATO model.
- Gas, Gas with Convection, and Electric Oven are each an option in the Warming Oven class.

Continue to add options to your model.



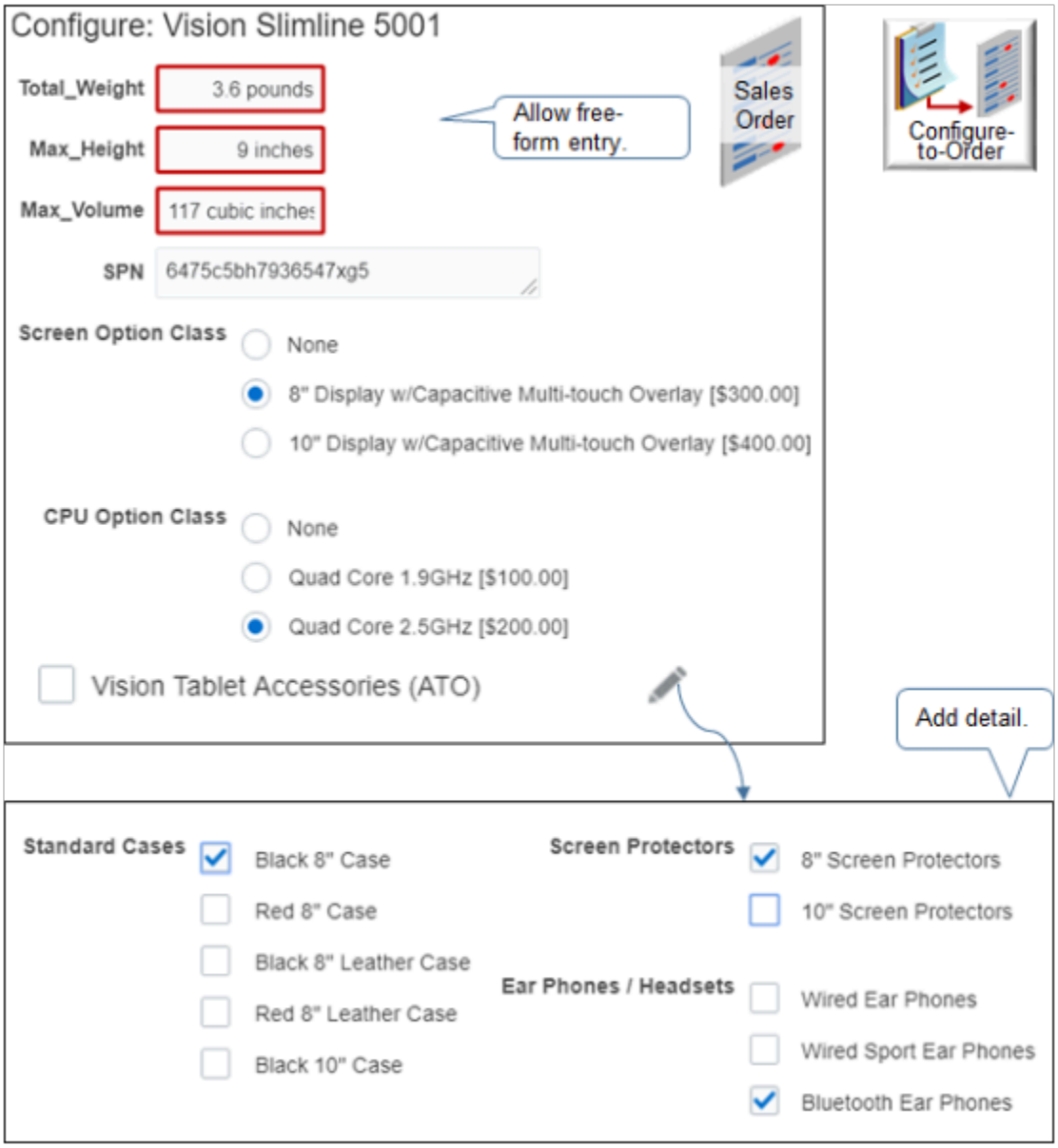
Note

- Burner Style ATO is a child model of the Stove Burner ATO model and a grandchild of the parent Deluxe Stove model.
- In this example, the user can choose two options for the Commercial Burners class.
    - The Large Commercial option specifies to use large commercial burners for the stove top.
    - The Griddle option specifies to include a griddle burner for the griddle part of the stove top. Commercial stove tops often include more than one burn area. One area includes an open flame and another area includes a flat, steel griddle. This model illustrates how you can set up your configuration to meet specialized customer requirements.
    - You can continue to add child, grandchild, great grandchild models and so on to meet your specific hierarchical requirements.

Learn about this example. For details, see *Overview of Using Web Services with Configure-to-Order*.

More Examples

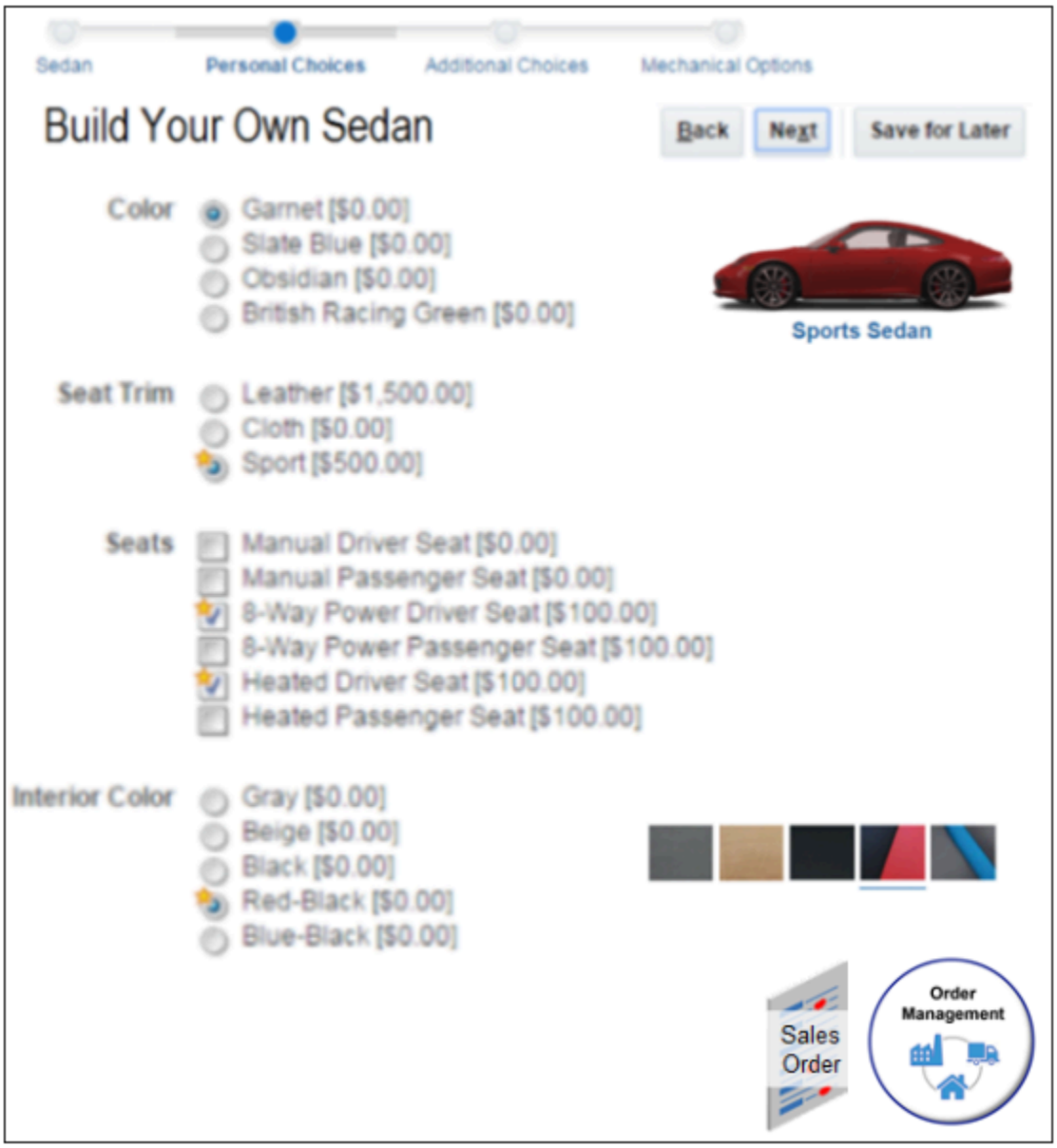
Here are some more examples.



For example:

- Use a transaction attribute to add content at run-time, such as the weight of the item.
- Add detail, such as display a second dialog that includes choices for accessories.

Here's an example where you choose options for a car, such as color, seat trim, seats, and so on.



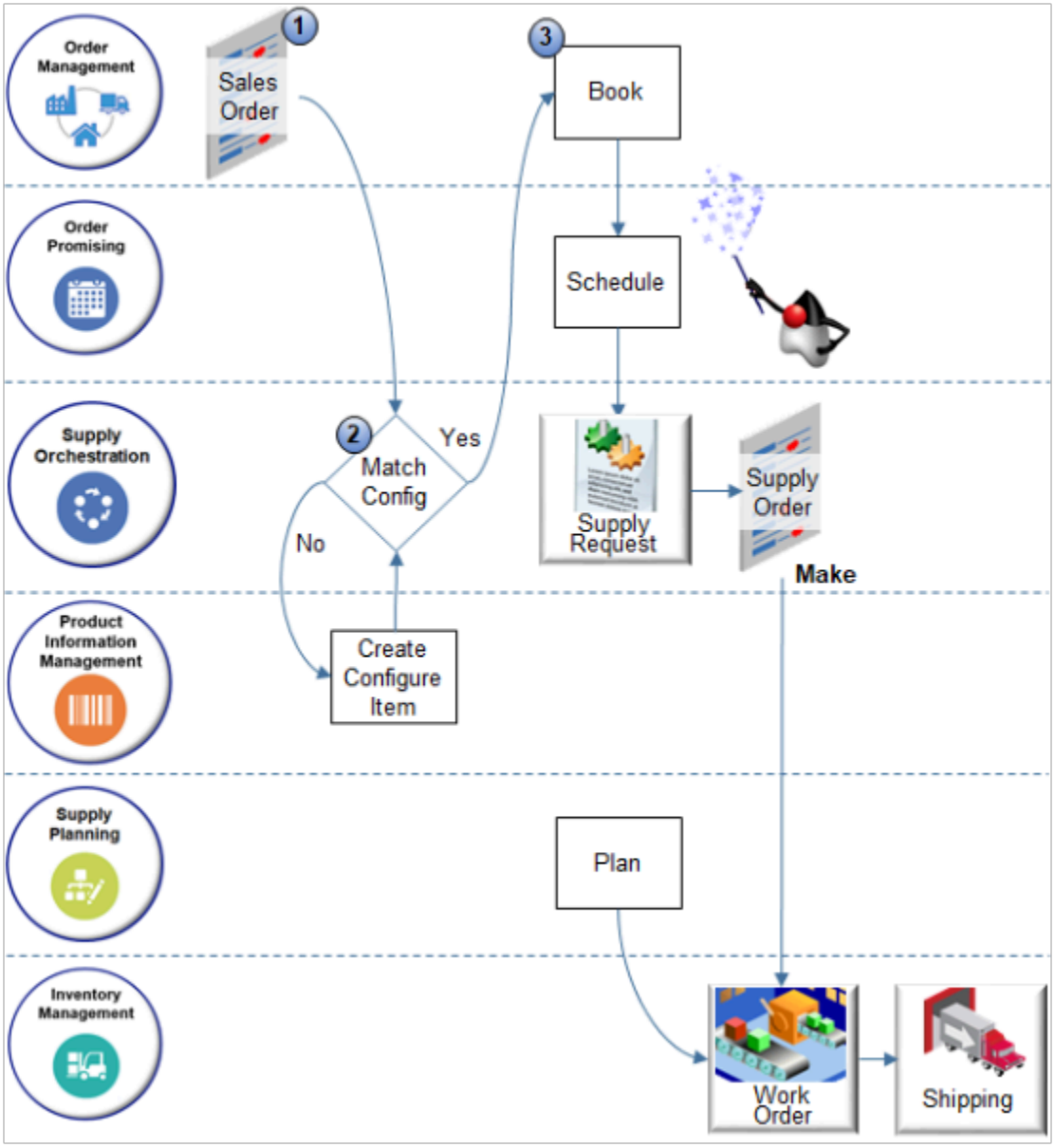
To get all the setup details, see the content that starts in *Guidelines for Setting Up Configuration Models*.

Related Topics

- Overview of Using Web Services with Configure-to-Order
- Add Configured Items to Order Lines

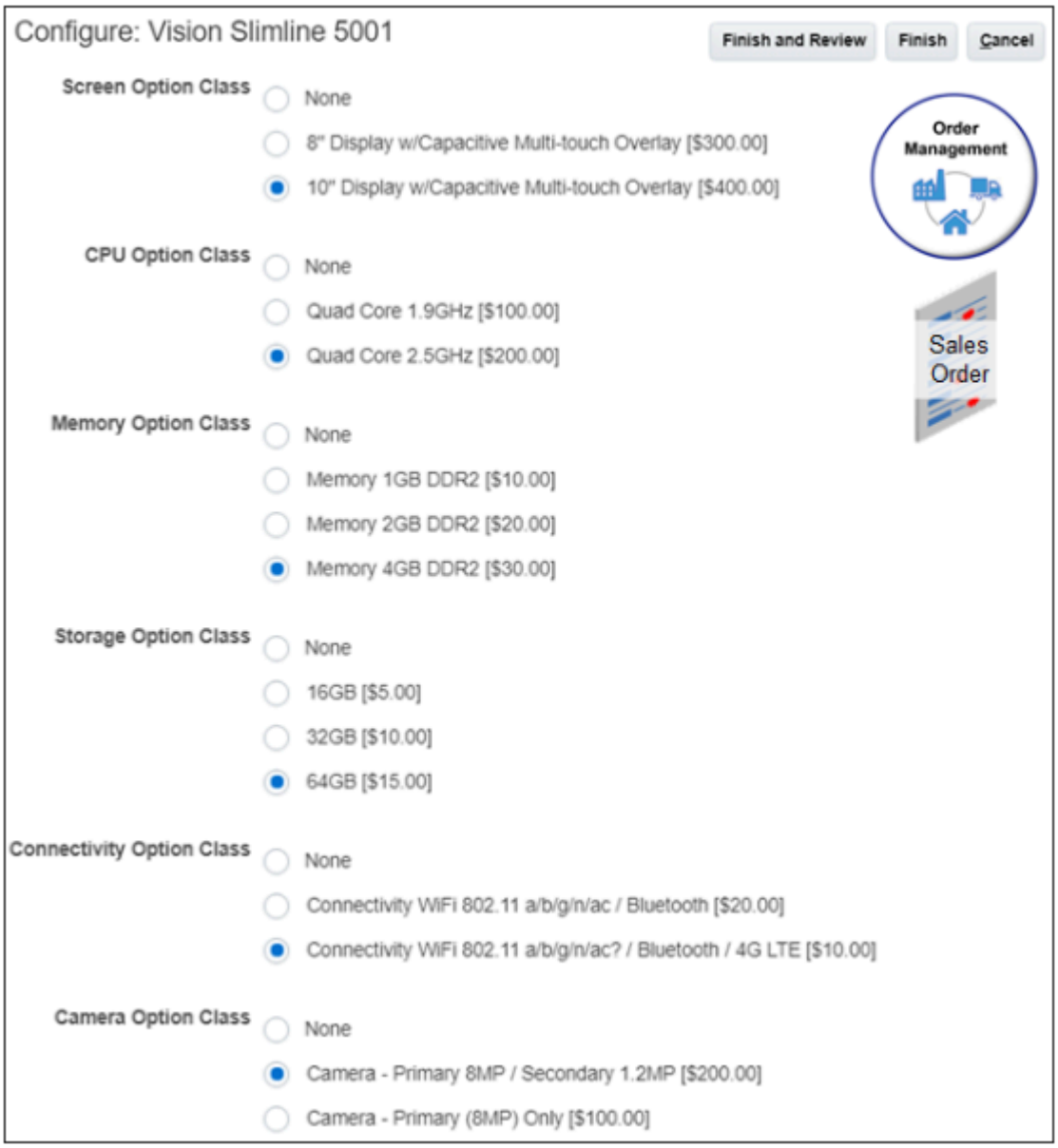
How Configure-to-Order Works

Here's an example of how Supply Chain Orchestration fulfills a configured item in a make flow.


Order Management
Sales Order
1
3
Book
Order Promising
Schedule
Supply Orchestration
2
Match Config
Yes
No
Supply Request
Supply Order
Make
Product Information Management
Create Configure Item
Supply Planning
Plan
Inventory Management
Work Order
Shipping

Note

1. Your user creates a sales order in the Order Management work area, searches the catalog line for the zCZ_AT6751010- Vision Slimline 5001 item, clicks Configure and Add, uses the Configure dialog to set configure options, clicks Finish, then submits the order.



2. Order Management sends a request to Supply Chain Orchestration. The request includes details about the configured item, including options that the user chooses.

   If its an assemble-to-order request, or a pick-to-order request with at least one assemble-to-order component, then Supply Chain Orchestration examines the request to determine whether it already processed the configured item during a prior request.

If Supply Chain Orchestration	Description
Hasn't already processed the configured item during a prior request.	Supply Chain Orchestration sends a request to Product Information Management to create a new one, and Product Information Management creates it. Product Information Management stores the configuration in a single record with a unique name. Order Management stores the record identifier in the Configuration Item attribute on the order line. For example: o Assume the user adds the Vision Slimline 5001 to an order line. It includes configure options for the screen and CPU. The user sets the screen to 10" and CPU to 2.5GHz. Supply Chain Orchestration searches it records and finds that no other order line has ever requested this configuration, so it sends a request to create it. o Some time later, the user adds another Slimline and sets the screen to 10" and CPU to 2.5GHz. Supply Orchestration finds the exact same configuration and uses it. o Some time later, the user adds another order line that contains the Slimline, but this time sets the screen to 8" and CPU to 2.5GHz. Supply Chain Orchestration doesn't find the exact same configuration, so it sends a request to create it. Supply Chain Orchestration sends a request to create a new configuration even if only one option is different from configurations it already created. This feature works the same across users, order lines, and orders.
Already processed the configured item with the same configure options in a previous request.	Supply Chain Orchestration sends the Configuration Item attribute to Order Management.

If Supply Chain Orchestration	Description

Here's how Product Information Management stores the zCZ_AT6751010.

Note

- A search for zCZ_AT6751010 returns the model and each instance of the model.
- zCZ_AT6751010 is the configuration model you create. Product Information Management doesn't update it.

- Supply Chain Orchestration sent a request to Product Information Management to create the other two instances when your user ordered them in Order Management.
  - zCZ_AT6751010*1088 is a unique instance of zCZ_AT6751010. It contains specific values that your user chooses, such as a 10" screen and 2.5GHz CPU.
  - zCZ_AT6751010*1089 is another unique instance of zCZ_AT6751010. It contains an 8" screen and 2.5GHz CPU.

  Product Information Management stores the item as a nonconfigured item with no structure. It doesn't store the structure of the configured item instance. It does this to avoid storing duplicate structural elements in tables. Supply Chain Orchestration stores the choices your user makes for each configure option and the structure in the match repository that Supply Chain Orchestration uses. If you enable transaction attributes on your item, then Supply Chain Orchestration also stores them in the match repository.

3. The flow continues. It uses the same steps that a make flow uses for an item that isn't configured. For details, see *How Supply Chain Orchestration Works*.

Points to Consider

- Configure-to-order primarily supports the make flow and the buy flow. It also supports the transfer flow and drop ship flow. For example, if your customer returns a configured item, then its available for transfer or ship from stock.
- Supply Chain Orchestration sends a request to Product Information Management to create the configured item for assemble-to-order when the user clicks Submit in the Order Management work area.
- The configure-to-order flow creates the configured item in Product Information Management on demand, when you need it. If the configuration already exists, then the flow reuses it instead of creating a new record.

  Inventory creates the physical item just-in-time, only when you need it, then ships it. Inventory doesn't stock the item before you need it. Inventory creates another instance of the physical item for each shipment.
- Configure-to-order also supports drop ship where you source and ship from a supplier. Supply Chain Orchestration doesn't manage supply for drop ship, but it does manage creating the configured item in a drop ship flow.
- To support the make flow, configure-to-order uses the work definition to create a work order at runtime. The model can be valid in Product Information Management but the flow can still fail in manufacturing. For example, if the work order in manufacturing fails to assign even one option or option class to an operation step, then the work definition isn't valid, Promising won't provide a recommendation, and the flow will fail.
- Use the View Configured Item Structure page or a web service to get details about each configure option. For details, see *View Structures of Configured Items*.
- You can't disable matching because it makes sure your flow is efficient.
  - You can add a transactional item attribute for configuration matching that implements specificity in your configuration. You can add it to an item class at any level of your model.
  - The Match Configuration attribute on the Specifications tab on the Edit Item page in Product Information Management applies to Oracle E-Business Suite. It has no effect on Oracle Cloud applications.
  - At runtime, you can't cancel a configure option in a pick-to-order item or kit when the shipment is out of proportion. Assume configured item A includes configure option B and configure option C. 1 unit of item A includes 1 unit of option B, and 1 unit of option C, so the proportion is 1-1-1 (1 unit of A equals 1 unit of B and 1 unit of C). Assume you order 10 units of item A, so the ordered proportion is now 10-10-10. Assume you ship 8 units of B and 2 units of C. The proportion in the shipment is 10-8-2, and its out of proportion. If the shipment is out of proportion, then you can't cancel the remaining quantity that hasn't shipped.

Child Components

Configure-to-order supports each item or assemble-to-order model as a child component in a pick-to-order model. The back-to-back flow can stock each component separately, in different locations.

- It isn't necessary to source all child components in a pick-to-order model from the same business-to-business flow.
- You can ship some children in a typical Order Management shipment as long as you ship them together with the business-to-business items.
- If you use the same date, then you can ship all children in a business-to-business flow or none in the business-to-business flow.
- You can drop ship all children from the same supplier.

Main Setup

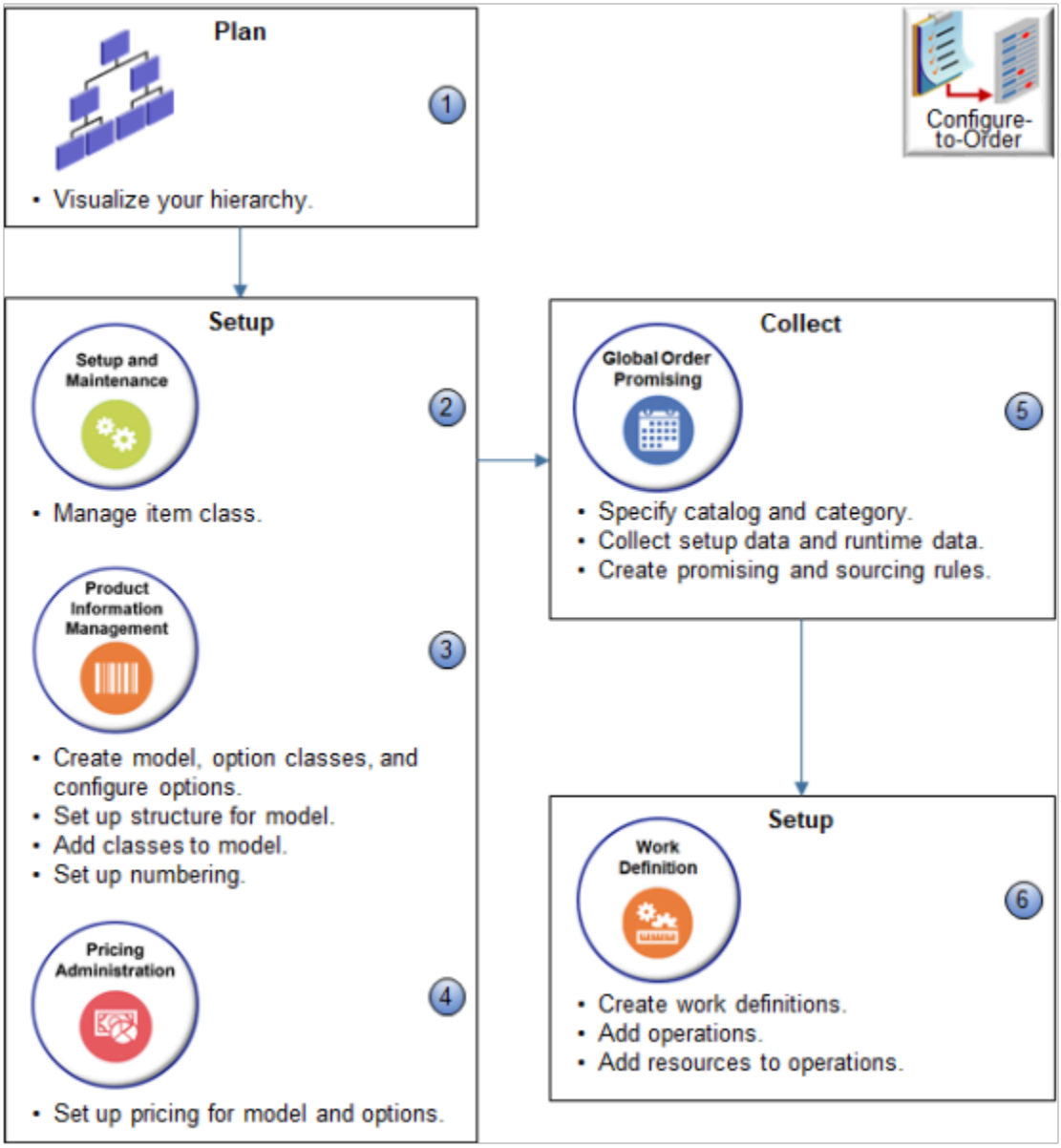
Overview

Guidelines for Setting Up Configuration Models

Follow guidelines to help your setup go smoothly.

Use a Proven Setup Sequence

You use various work areas to set up your configuration model. Here's an example sequence for the make flow.



Note

1. Use a graphics program or pen and paper to visualize your hierarchy.
2. Use the Setup and Maintenance work area to manage the item class.
3. Use the Product Information Management work area.
    - Create the model, option classes, and configure options.
    - Set up a structure for your model.
    - Add classes to your model.
    - Set up numbering.
4. Use the Pricing Administration work area to set up pricing for your model and configure options.
5. Use the Global Order Promising work area.
    - Specify catalog and category.

- Collect setup data and runtime data.
- Create promising and sourcing rules.

6. Use the Work Definition work area.
   - Create work definitions.
   - Add operations.
   - Add resources to operations.

Organize Your Hierarchy

Your hierarchy can include parents and children, but there are limits about what you can include in assemble-to-order (ATO) items, pick-to-order (PTO) items, and kits. Here's what's allowed.

Hierarchy	Allowed?
ATO child in an ATO parent	Yes
ATO child in a PTO parent	Yes
PTO child in a PTO parent	Yes
PTO child in an ATO parent	No
ATO child or PTO child in a kit parent	No
Kit child in a kit parent, ATO parent, or PTO parent	No

Assume you sell these items in various configurations:

- Kitchen Appliances
- Stove
- Stove Burners
- Espresso Station
- Milk Frother
- Coffee Grinder
- Air Fryer

Here are some example hierarchies.

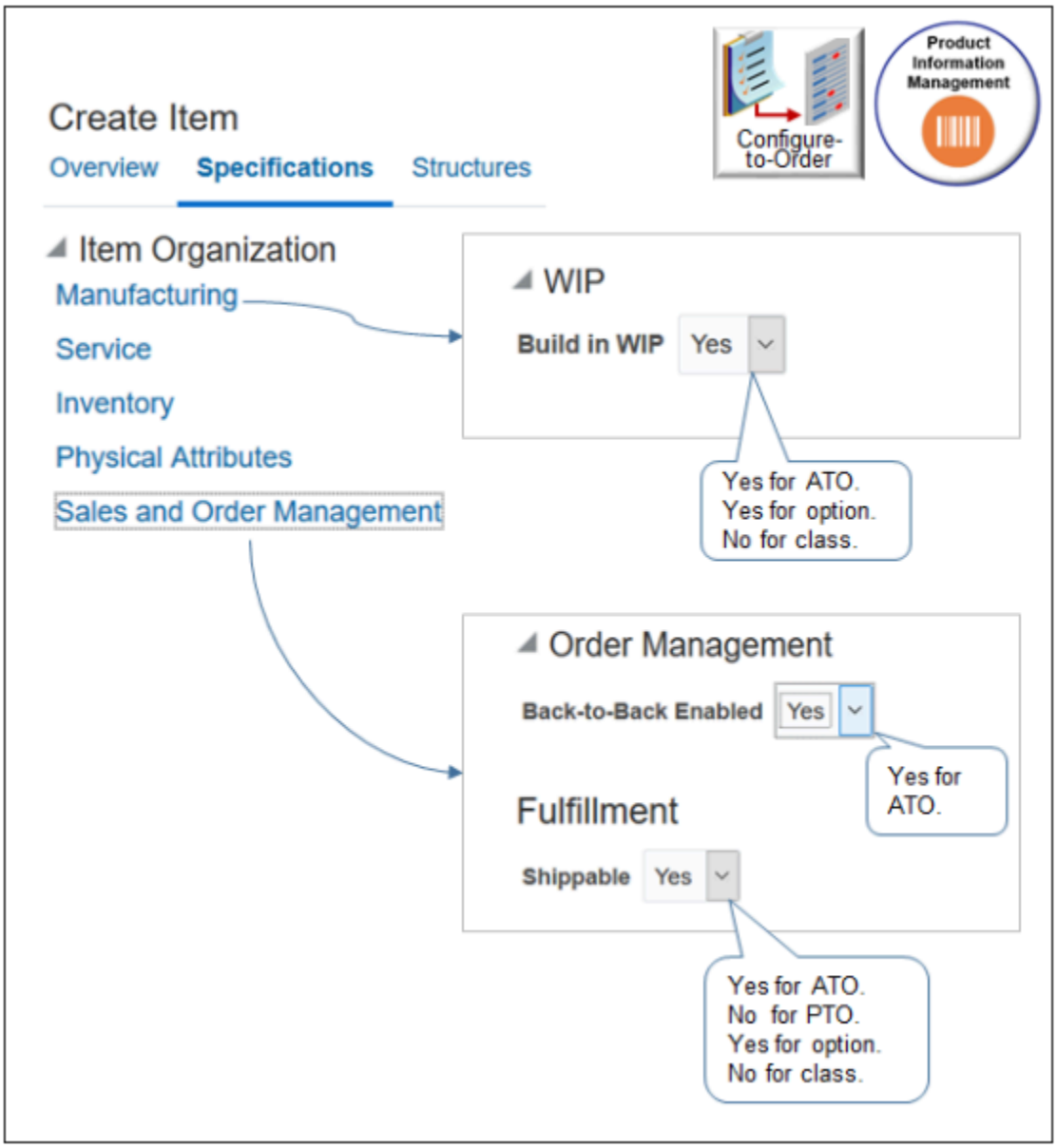
Allowed	Not Allowed
ATO Stove ATO Stove Burners	ATO Stove PTO Stove Burners
PTO Espresso Station ATO Milk Frother	Kitchen Appliances Kit Air Fryer Kit
PTO Espresso Station	ATO Espresso Station

Allowed	Not Allowed
PTO Coffee Grinder	Milk Frother Kit
ATO Kitchen Appliances ATO Stove ATO Stove Burners ATO Espresso Station ATO Milk Frother ATO Coffee Grinder	ATO Kitchen Appliances ATO Stove ATO Stove Burners PTO Espresso Station ATO Milk Frother ATO Coffee Grinder You can't have a PTO child in an ATO parent, so you can't have PTO Espresso Station as a child of ATO Kitchen Appliances. If you need to include the Espresso Station as a child of Kitchen Appliances, then change Kitchen Appliances from ATO to PTO.

More

- You can include a configure option in your hierarchy that isn't part of an option class.
- You can set up a hybrid configuration model. For example, add a child assemble-to-order model to a parent pick-to-order model or assemble-to-order model. The configurator views each child as a component of the parent, and it creates all child models at the same time.
- Reuse your option classes. For example, create an option class for hard drives, then reuse it for different laptop computer models. Or sell it as an individual component to your customer who needs to expand their drive capacity or replace a failed drive.

Set Up Your Item



Go to the Product Information Management work area, then:

- Click **Tasks > Create Item** to create the model, each class, and each option.
- Click **Specifications > Manufacturing**.
    - Set Build In WIP to Yes for an assemble-to-order model and each option. It instructs manufacturing to build the item.
    - Set Shippable to Yes for an assemble-to-order model and each of its options. Set it to No for a pick-to-order model and each of its options. Pick-to-order is already built, for example, with a return or transfer order.
    - Set Shippable and Build In WIP to No for any option class. You don't build or ship the class. You build and ship the option in the class.
- Click **Specifications > Sales and Order Management**.

- Set Back to Back Enabled to Yes for an assemble-to-order model for all Inventory organizations.
- Set Shippable to Yes for an assemble-to-order model. Set it to No for a pick-to-order model and each of its option classes and options.

Set other options.

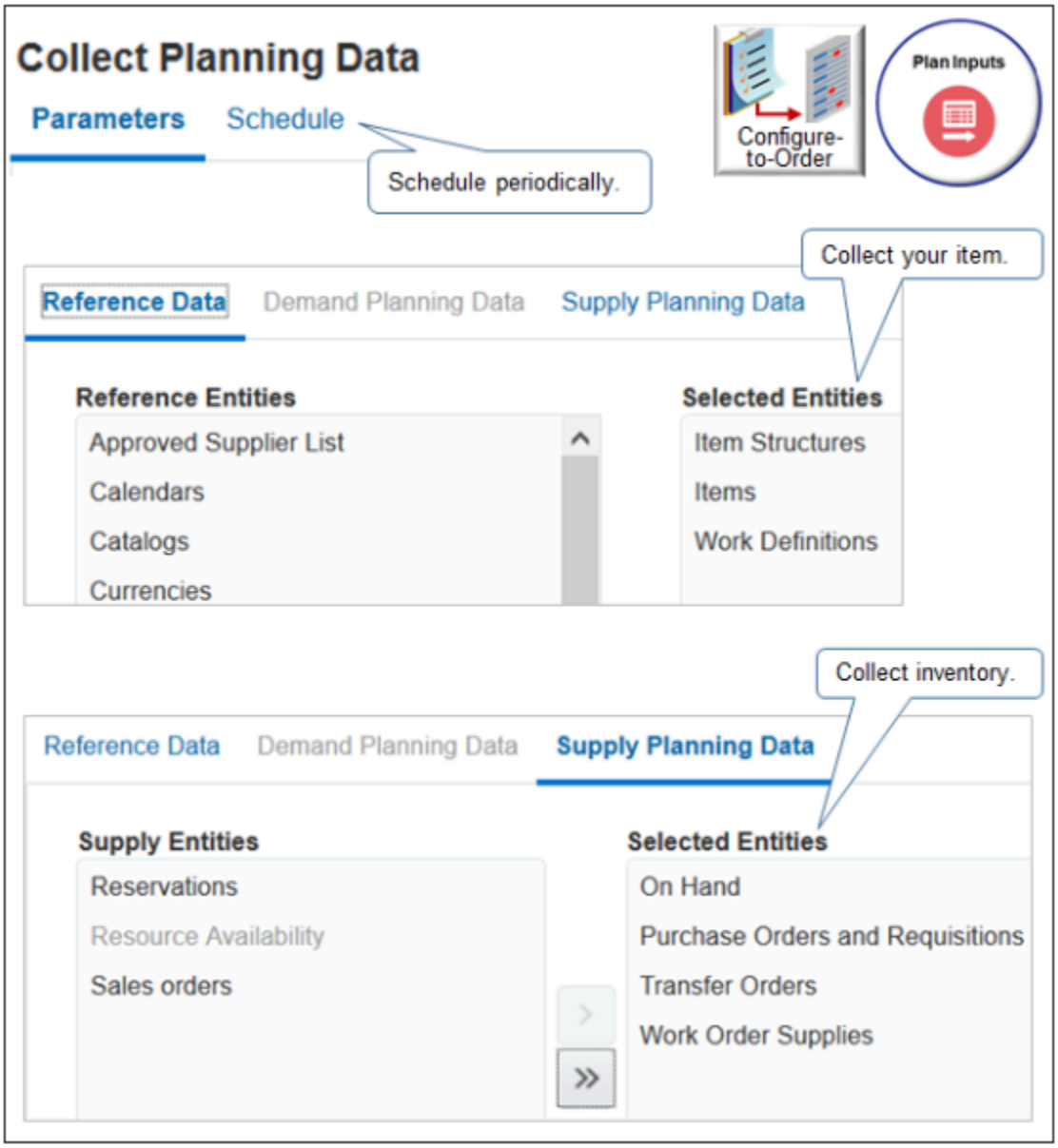
- Set the Optional attribute to Yes for each component of an assemble-to-order or pick-to-order item when you create the item in Product Information Management. Setting it to Yes means the user can set a value for it, or not set any value.

  For example, if you set Optional to Yes for the Hard Drive component, and if the user doesn't choose an option for the hard drive, such as 1TB or 2TB, then configure-to-order uses the default value that you set for the hard drive.
- If you use a transaction attribute, then, to create a new configured item at run time, set the scope to Configuration Matching in the item class for your transaction attribute. You must set the scope for each transaction attribute that's unique in your assemble-to-order model. If you don't do this, then Order Management will save the value, but Supply Chain Orchestration won't save it or send it to manufacturing.

Set Up Global Order Promising

Here are some setups you can do.

- Promise an assemble-to-order item, pick-to-order item, or a hybrid that includes an assemble-to-order item inside a pick-to-order item.
- Consider lead times for the model, option class, and option when determining the lead time for the configured item. You can do supply chain availability but not lead-time promising when you use an available-to-promise rule in a business-to-business flow. You can set some values, but don't mix rule types.
- Exclude sourcing on each option and option class.
- Create a supply recommendation at runtime for each configured item in a back-to-back flow.
- Promise each configured item in a drop ship flow.
- Make sure Supply Planning is available for each component. Planning plans each component, including child models.



Note

- Go to the Plan Inputs work area, then click **Tasks > Collect Planning Data**.
- Collect your configured item on a schedule, such as at the end of each day, so the flow can include them in the on-hand quantity. For example, the return orders you receive throughout the day might increase on-hand quantity.
- Collect items, item structures, and work definitions for your new model. Promising won't provide a make recommendation if you don't collect the work definition.
- Collect on-hand inventory for each configure option. If even one component isn't in stock, then Promising won't display availability for your model.
- You must collect each new configured item.
- Refresh the repository after you collect.

Avoid the Not Available message at runtime.

- Create an available-to-promise (ATP) rule to get a recommendation from Promising during the back-to-back flow.
- Explicitly reference the model and each component in your available-to-promise rule. If you don't explicitly reference them, then include them in a category.

Consider Resource Capacity

Consider resource capacity and availability for each configured item that you manufacture.

If	Then
• You want to promise according to actual lead time. • And you must do an operation that depends on a configure option that resides below the option class. • And you want promising to consider resource capacity and availability when it promises a configured item that you manufacture.	You must specify the operation assignment for the item at the first level component of the model. You do this when you set up the work definition for your assemble-to-order model. The flow ignores assignments you set at lower levels.

Note

- You can use an available-to-promise rule to promise according to fixed and variable lead times for an attribute on the option.
- Oracle Manufacturing Cloud supports assignments at any level in the model.

Use the Configurator Models Work Area

Use the Configurator Models work area to extend your configuration model. For example:

- Set default choices or values.
- Automatically select a configure option according to another choice your user makes when configuring the item.
- Prevent your user from selecting options that result in a configured item that won't work.
- Control the number of instances your flow creates for a model at runtime.
- Calculate and set the values for numeric configure options.

If you use the Configurator Models work area, then follow these guidelines.

- If you must display a transaction attribute at runtime, then import your model into the Configurator Models work area.
- If you revise a model, update the item class, add it to an unreleased workspace, then release the workspace into the Configurator Models work area.

For details, see *Create and Maintain Configurator Models*.

Import Configurator Models

If you use the Configurator Models work area, and if you import a configured item at runtime through.

- **File-base data import**. Use the same template you use when you import a sales order that includes only one order line.
- **Web service**. Use the same web service you use when you import a sales order that includes only one order line.
- Make sure your import includes child lines for the configure options and that it establishes a relationship between the child and parent configured item.

I Use My Own Application to Set Up the Configuration Model

Assume you use your own application to set up a configuration model and to create the bill of materials that Oracle Applications use to build the model, and you use that list to import the order lines that contain your configured item and its components. If at some later point you import a revision for the model, then you must make sure the sequence that you use in the revision is identical to the sequence you used in the original import.

Assume you import a configuration model for a desktop computer, it has 3 components, and you import this sequence and hierarchy.

```
Desktop Computer
4K Monitor
HD Monitor
```

You also have a rule that says depending on the value of the Desktop Computer, if user doesn't select 4K Monitor, then add the HD Monitor to the configured item.

Some time later, you import a revision for Desktop Computer. If your revision doesn't import Desktop Computer, 4K Monitor, and HD Monitor in the same sequence that you used in the original import, then the configuration might fail. For example, if your revision imports 4K Monitor before it imports Desktop Computer, then the rule might fail because the rule logic depends on the value of Desktop Computer before it can proceed to make the decision about HD Monitor.

I Use Oracle Configurator to Set Up the Configuration Model

If your application integrates with Oracle Configurator and you used it to set up the configuration model, then don't import the entire list of components from the model when your send your revision. Instead, import only the configuration header and revision number, and the import will communicate with Oracle Configurator to recreate the component list.

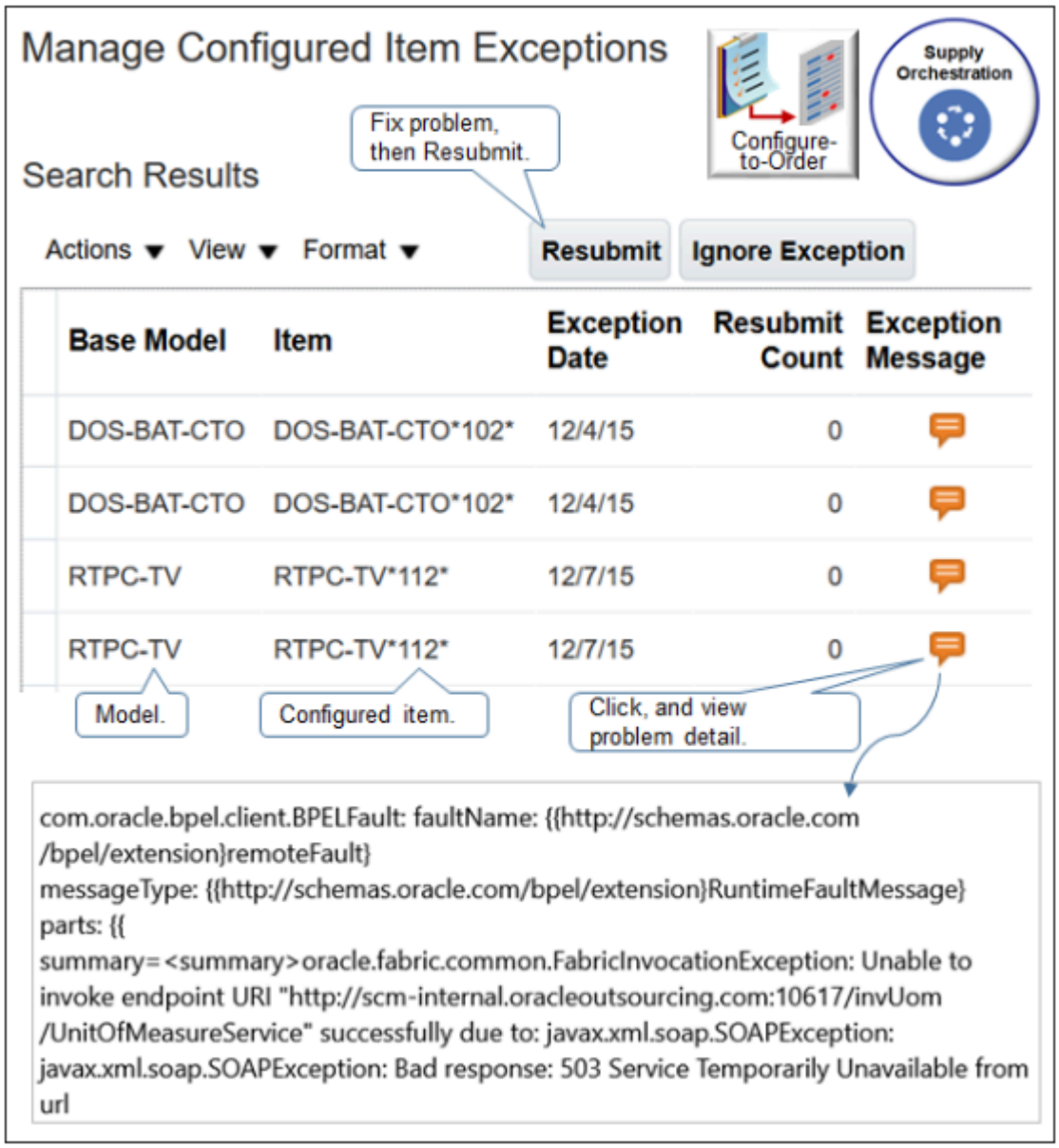
Troubleshoot

You set up a configuration model in two phases.

1. Create the model, options, classes, and hierarchy in Product Information Management.

2. Send details about the model to your downstream systems.

   Use the Manage Configured Item Exceptions page to examine most errors that occur in this step. Correct the error and resubmit, or ignore it.



For details, see *Troubleshoot Problems with Configure-to-Order*.

Other Guidelines

Use two browser applications during set up. Opening a second browser application allows you to toggle between work areas without having to close a work area and then open another work area. For example, you can keep the Product Information Management work area open in browser x, open the Work Definition work area in browser y, then toggle between x and y as necessary if you find you need to modify your set up in Product Information Management.

Related Topics

- Troubleshoot Problems with Configure-to-Order
- Overview of Phantom Explosion in Work Orders

Overview of Setting Up Configuration Models

Set up a configuration model, including the model hierarchy, item class, creating the model, pricing, and so on.

Summary of the Setup

Do the work described in these topics.

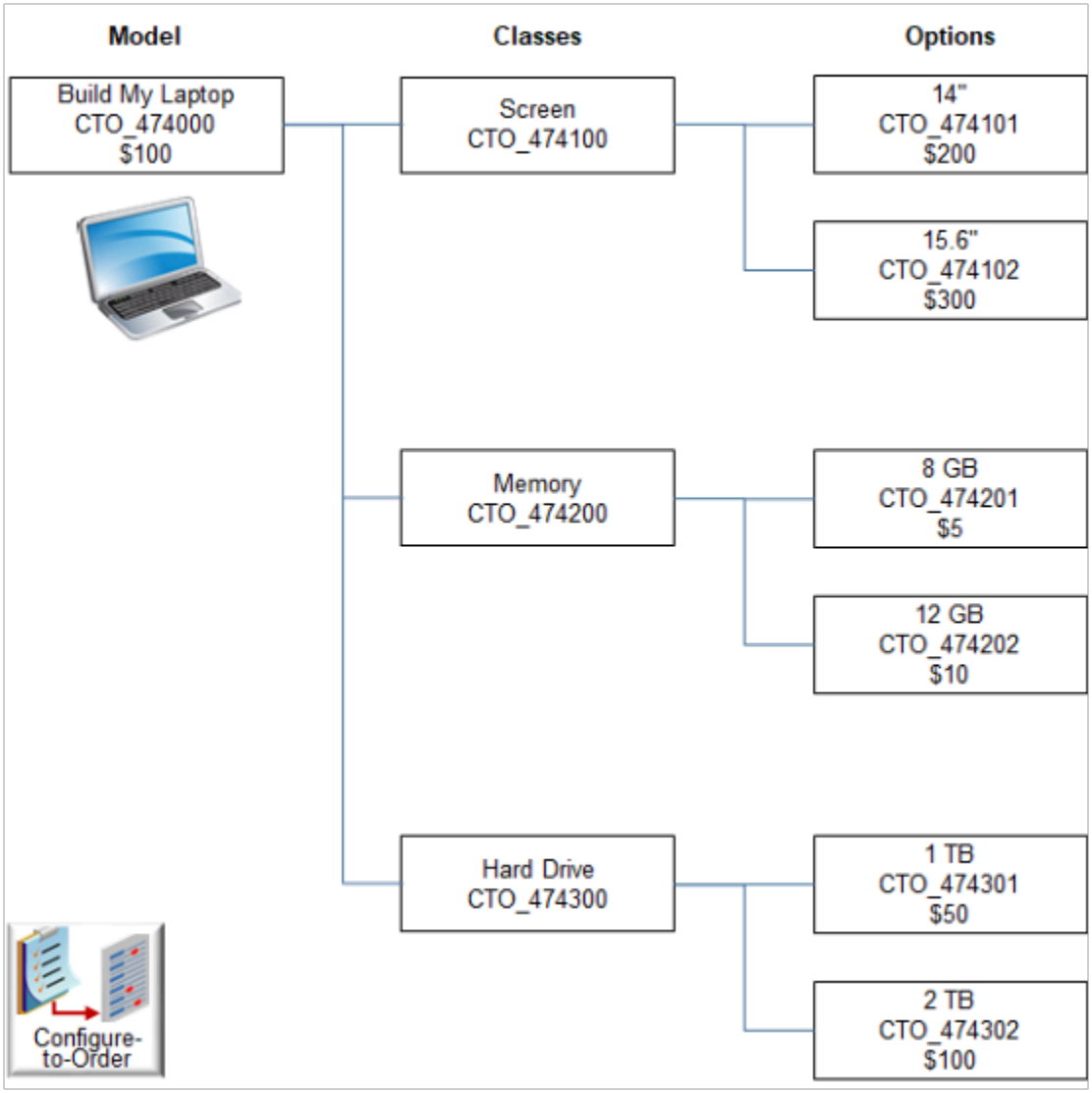
1. *Visualize the Hierarchy for Your Configuration Model*
2. *Manage Your Item Class*
3. *Create Your Configuration Model*
4. *Set Up Numbers for Your Configuration Model*
5. *Set Up Pricing for Your Configuration Model*
6. *Specify Catalog and Category for Your Configuration Model*
7. *Collect Planning Data for Your Configuration Model*
8. *Create Promising Rules for Your Configuration Model*
9. *Create Sourcing Rules for Your Configuration Model*

Details

Visualize the Hierarchy for Your Configuration Model

Use a graphics program or pen and paper to visualize the hierarchy.

Assume you must set up configuration model CTO_474000, Build My Laptop, in a make flow that uses assemble-to-order.



Note

- Come up with a meaningful name for your model, such as Build My Laptop.
- Identify the classes in your model that you will allow your customer to choose. For example, you can often choose from different screen sizes, memory, and hard drives when you buy a laptop, but you usually can't specify the voltage for the hard drive.
- Identify the options you will provide in each class, such as a 14" screen and a 15.6" screen.
- Design a number nomenclature that uniquely identifies the model, each class, and each option. For example, CTO_474000 identifies the model, where CTO means configure-to-order, and 474000 identifies the model. Each class and each option use CTO_474xxx to identify that they're part of CTO_474000. For example:
    - The Screen class uses CTO_474100 to identify that its part of CTO_474000.
    - The 14" option uses CTO_474101 to identify that its part of the CTO_474100.
- Determine the price you plan to charge for the model and each option.

Related Topics

- Item Classes

Manage Your Item Class

Use a predefined class or create your own for your configuration model.

Create your own class so you can manage your model independently of other models, such as how to number each unique configuration, specifying who can configure the model, specifying transaction attributes to store data that the flow creates at runtime, and so on.

For now, you will do the minimum setup you need to get started when you create your own class.

1. Go to the Setup and Maintenance work area, click **Tasks > Search**, search for, then open Manage Item Classes.
2. On the Manage Item Classes page, in the search results, click **Actions > Create**.
3. In the Create Item Class dialog, set the values, then click **Save and Add Details**.

Attribute	Value
Item Class	My Laptop
Internal Name	My_Laptop
Description	Class to use for the Build My Laptop model.

4. On the Edit Item Class page, verify the options, then click **Save and Close**.

Attribute	Value
Enabled	Contains a check mark
Item Creation Allowed	Contains a check mark

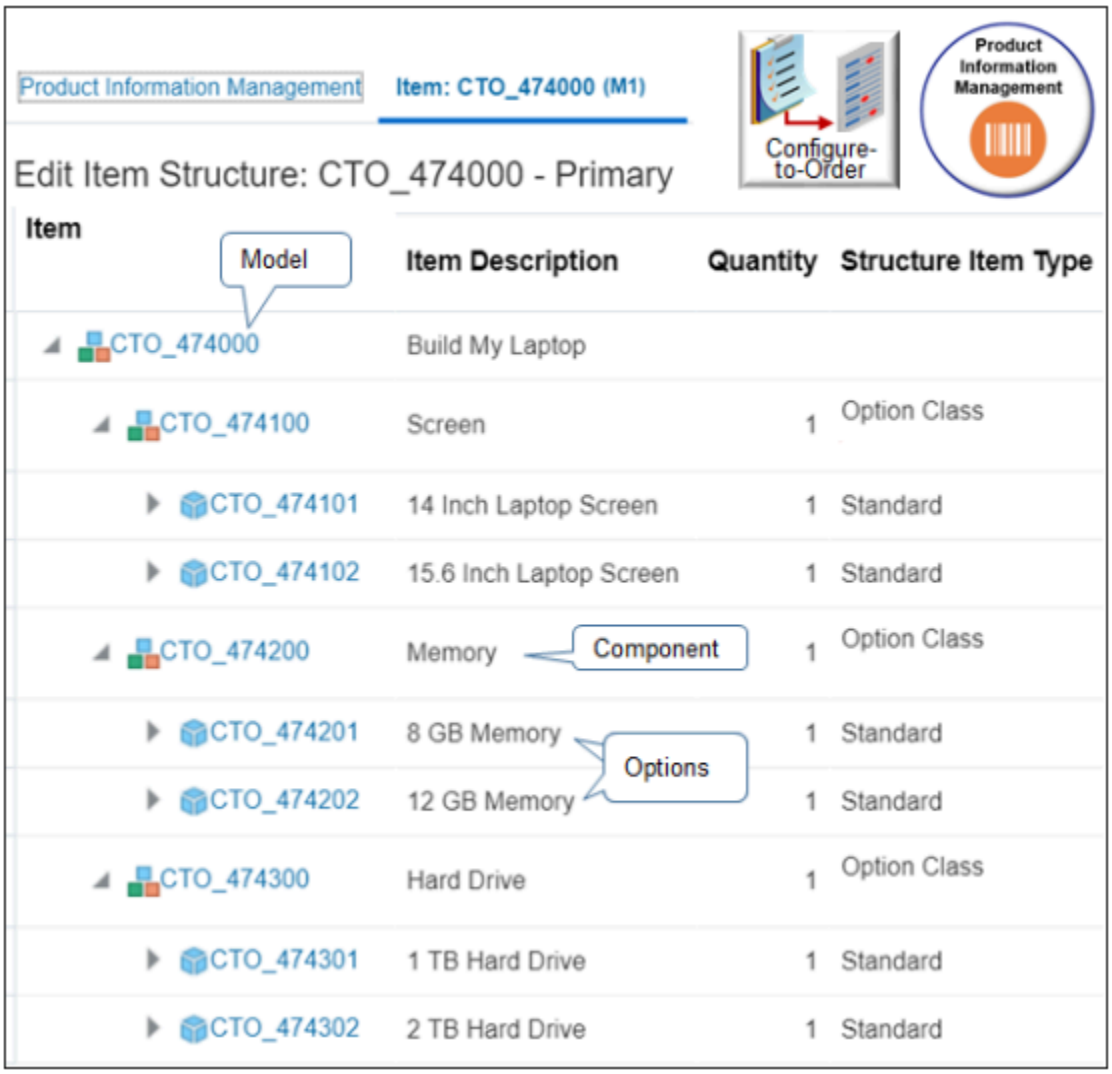
Related Topics

- Item Classes

Create Your Configuration Model

Use the Product Information Management work area to create your configuration model.

Here's an example model you will create.



Summary of the Set Up

1. Create classes and configure options.
2. Add options to your option classes.
3. Create the model.
4. Add classes to your model.
5. Associate your model with an inventory organization.
6. Verify your set up.

Create Classes and Configure Options

You start at the bottom of the hierarchy. You create classes and configure options, then add them to the model.

1. Go to the Product Information Management work area.
2. On the Product Information Management page, click **Tasks > Create Item**.
3. In the Create Item dialog, set values, then click **OK**.

Attribute	Value
Organization	V1

Attribute	Value
	V1 is an abbreviation for Vision Operations.
Create New	Selected
Number of Items	9 This is a nifty feature you can use to add more than one item to the same class.
Item Class	My Laptop This is the class you created earlier.
Template	Finished Goods As an alternative, use the ATO Item template for configure options and the ATO Option Class template for option classes. If you use them, verify each value that the template sets.

4. On the Create Multiple Items page, don't click Save. Instead, set the values.

Item	Description	User Item Type
CTO_474101	14 Inch Laptop Screen	Finished Good
CTO_474102	15.6 Inch Laptop Screen	Finished Good
CTO_474201	8 GB Memory	Finished Good
CTO_474202	12 GB Memory	Finished Good
CTO_474301	1 TB Hard Drive	Finished Good
CTO_474302	2 TB Hard Drive	Finished Good
CTO_474100	Screen	ATO Option Class
CTO_474200	Memory	ATO Option Class

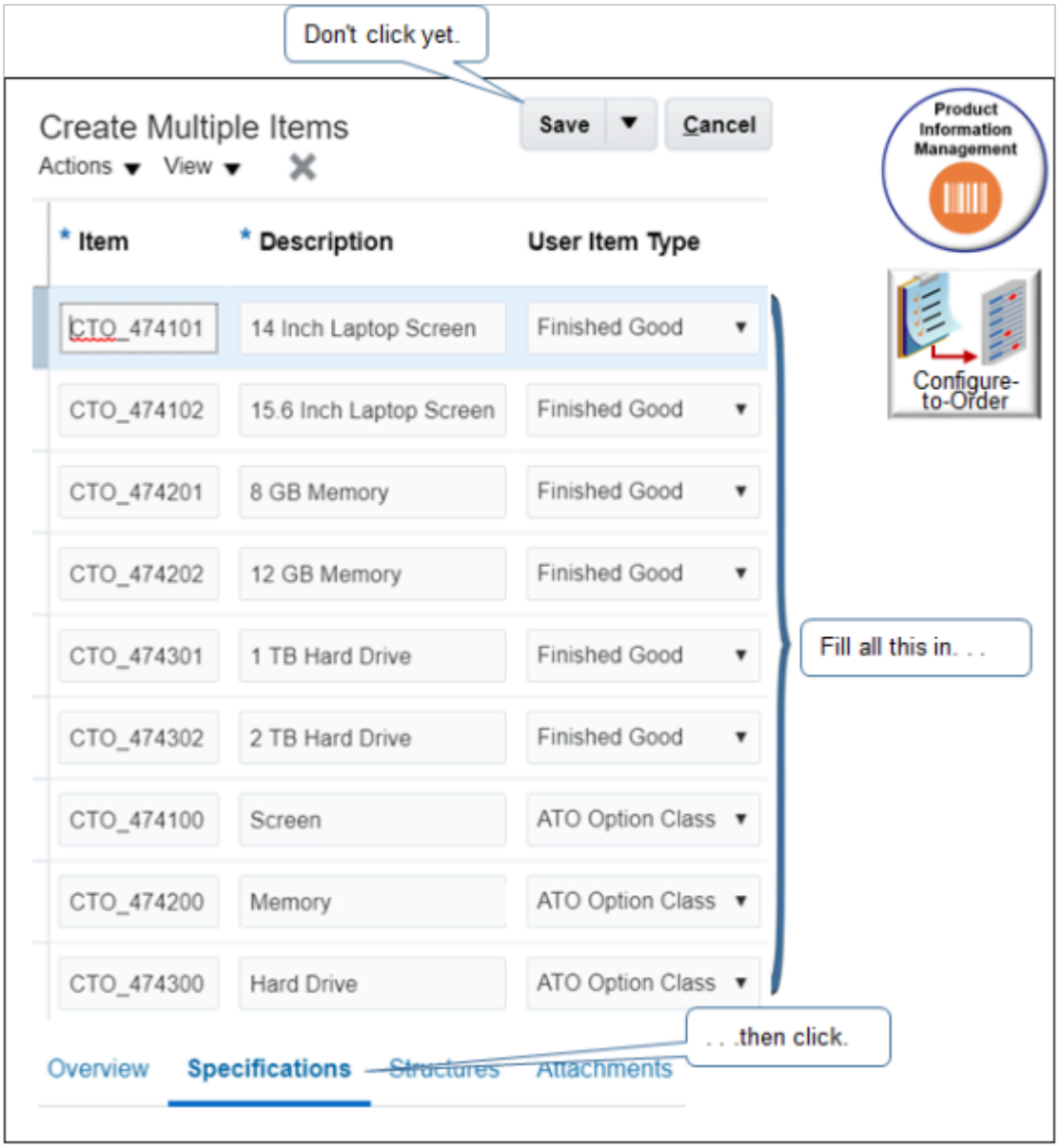
Item	Description	User Item Type
CTO_474300	Hard Drive	ATO Option Class

Leave other attributes at their default values.

Attribute	Value
Item Status	Active
Lifecycle Phase	Design
Pack Type	Leave empty.

Attribute	Value

Here's what your set up should look like.



5. Set the values for the first item in the list.
    - Click the **row** that contains CTO_474101 in the Name column, click **Specifications > Sales and Order Management**, then set the value.

Attribute	Value
Order Management Indivisible	Yes Specify whether your user can use a decimal quantity for the item. If Yes, then your user can use only a whole number for the quantity, such as 1, 2, or 3, and can't use a decimal, such as 1.4. In this example, each item is indivisible. For example, you can't order a fraction of a screen, memory, or hard drive.

    - Click **Associations**, click **Actions > Select and Add**, search for an inventory organization, click **Apply**, then click **Done**.

Attribute	Value
Organization	M1 You must specify an inventory organization. If you don't, you can't create a work definition for your item.
Organization Name	Seattle Manufacturing In this example, assume you already set up Seattle Manufacturing as an inventory organization.

6. Repeat step 5 for each row.
7. Set the values for the option classes.

    You use the option class to group options but you don't actually ship the class. You do ship the option. So, you set some values differently for the option class than you do for the option.

    - Set the values for the screen. Click the **row** that contains CTO_474100 in the Name column, click **Specifications**, click **Manufacturing** under Item Organization, then set the values.

Attribute	Value
Structure Item Type	Option Class
Pick Components	No

Attribute	Value
Assemble to Order	Yes
Build in WIP	No
Supply Type	Phantom

The class is a phantom. You don't pick, assemble, or build it. You do pick, assemble, and build the option.

- Click **Sales and Order Management**, then set the value.

Attribute	Value
Shippable	No You don't ship the class. You ship the option.

- Repeat this step for the memory option class, CTO_474200.
- Repeat this step for the hard drive option class, CTO_474300.

8. Accept all other default values, then click **Save > Save and Close**.

Add Options to Your Option Classes

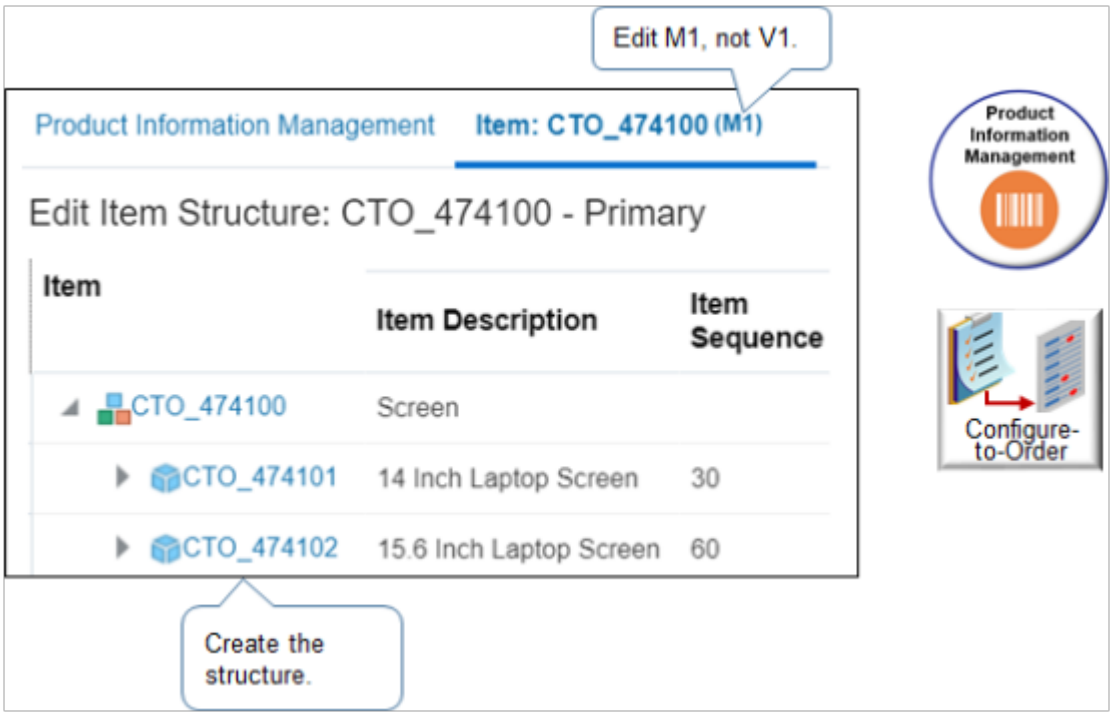
1. Search for and open the screen class for editing.
   - Click **Tasks > Manage Items**.
   - Search for the screen class.

Attribute	Value
Name	CTO_474100

Attribute	Value

- In the search results, click the M1 version.

> **Note:** Product Information Management creates a new version of each item when you associate the item with an inventory organization, then appends the name of the organization, such as M1, to the name of the item. The search results display the V1 and M1 versions in separate rows, but doesn't display V1 or M1. It displays only the name. Make sure you click the M1 version. Its usually the second row.



2. Add the structure.
   - On the Edit Item page, click **Structures**, then click **Actions > Create from Common**.

     Don't use **Actions > Create**. If you do, you will be creating a separate structure. Instead, click Create from Common to use the structure you created for the V1 version.
   - In the Create New Item Structure from Common dialog, set the values, then click **OK**.

Attribute	Value
Organization Name	Vision Operations
Item	CTO_474100
Structure Name	Primary

Attribute	Value
Common Structure Levels	First Level

- In the Item Structures list, in the Name column, click **Primary**.
- On the Edit Item Structure page, click **View**, then add a check mark to **Component Order Management**.

3. Add an option.

- On the Edit Item Structure page, click **Actions > Select and Add**.
- In the Select and Add dialog, search for the item.

Attribute	Value
Item	CTO_474101

- In the search results, click the **row**, then click **Edit and Add**.
- In the Edit Multiple Components dialog, set the values, then click **OK**.

Attribute	Value
Optional	Set to No for the option classes. Set to Yes for the configure options. Note - An option class can be required or optional. For example, a laptop computer can't function without a screen, so the screen class isn't optional. - If you set Optional to No for the option class, and if you set Optional to Yes for at least one option in the class, then the Order Management work area will display the class on the Configure page at runtime. This allows your user to choose the option. - If you set a configure option to No, then Order Management won't display it at runtime, your user can't set the option, and the purchase order and shipping documents won't include it. If its a manufactured item, then the work order will still include it because its part of the item structure.
Minimum Quantity	1 Specify the minimum quantity the user can set for this option. A laptop computer must have one and only one screen, so set the minimum quantity to 1 and the maximum quantity to 1.
Maximum Quantity	1
Instantiability	Don't modify for this example.

Attribute	Value
	Specify whether to create a separate flow for each instance of the item on the order line. If the quantity of the component is greater than one, and if Instantiability is Yes, then use a separate instance for each assemble-to-order flow. For example, assume you add an order line for the CTO_474000 with a quantity of 12, and Instantiability is. - Enabled. Create 12 separate flows and set the quantity on each flow to 1. Enable Instantiability when each item requires a slight variation that the flow must fulfill. For example, the item is a personal computer that includes two hard drives, and you must serialize each drive. For another example, the item is a cell phone and you must provision a unique telephone number for each phone. - Not enabled. Create one flow and set the quantity on the flow to 12.

4. Repeat step 3 for the CTO_474102 option.
5. Click **Done**, then click **Save > Save and Close**.
6. Repeat these steps but create the memory hierarchy.

```
CTO_474200
 CTO_474201
 CTO_474202
```

7. Repeat these steps but create the hard drive hierarchy.

```
CTO_474300
 CTO_474301
 CTO_474302
```

Create the Model

Create the CTO_474000 model so it works in an assemble-to-order flow.

1. On the Product Information Management page, click **Tasks > Create Item**.
2. In the Create Item dialog, set values, then click **OK**.

Attribute	Value
Organization	V1
Create New	Selected
Number of Items	1
Item Class	My Laptop
Template	ATO Model Move ATO Model from the available list to the selected list.

Attribute	Value
	Make sure the selected list contains only ATO Model.

3. On the Create Item page, set the values, then click **Save**.

Attribute	Value
Name	CTO_474000
Description	Build My Laptop
Item Status	Active
Lifecycle Phase	Design
User Item	ATO Model
Pack Type	Leave empty

4. Click **Specifications**, then set the values. Use these values for an assemble-to-order flow.

Attribute	Value
Structure Item Type	Model
Autocreated Configuration	No
Pick Components	No
Assemble to Order	Yes
Build in WIP	Yes Set it to Yes for an assemble-to-order item that you will make.

Attribute	Value

If you were setting up a pick-to-order flow, you would use different values.

Attribute	Value
Pick Components	Yes
Assemble to Order	No
Build in WIP	No

5. Click **Sales and Order Management**, then set the values.

Attribute	Value
Customer Ordered Customer Orders Enabled Order Management Transaction Enabled	Yes If you set these attributes to No, the flow will fail in Order Management.
Back-to-Back Enabled	Yes This attribute allows Supply Orchestration to procure the item in a back-to-back flow, such as buy or make. o Set to Yes for an assemble-to-order item. o Set to No for pick-to-order. You can enable the configure options for back-to-back in a pick-to-order flow, but not the model.
Shippable	Yes Note o Set to Yes for an assemble-to-order model. Order Management views the model as the item. The configured item is an attribute on the model. So you must make the model is shippable. o Set to No for a pick-to-order model. Each item in the pick-to-order model is shippable, but the pick-to-order model only represents the collection of items to ship. It isn't shippable.
Ship Model Complete	No Note

Attribute	Value
	◦ Set to No for assemble-to-order because an assemble-to-order model is a single item. ◦ Set to Yes for pick-to-order because Order Management only supports shipping all pick-to-order items together from the same warehouse.
Invoiced	Yes If you set this attribute to No, then the flow won't send any details to Accounts Receivable.

6. Click **Planning**, then set the values.

Attribute	Value
Planning Method attribute in the MPS MRP Planning area	MPS Planning If you set it to MPS Planning or MRP Planning, then the model, option classes, options, and components display in Planning Central. For details, see *Item MRP and MPS Planning Specifications*.
Forecast Control	Consume Then Explode Use this value if you intend to forecast the model. For details, see *Forecast an Assemble-to-Order Item*.

7. Click **Purchasing**, set the values, then click **Save**.

Attribute	Value
Purchasable	Yes Set to Yes for the model, each option class, and each option.
List Price	100 Purchasing uses this value as the default purchase price in the purchase order. It applies only when you create a blanket purchase agreement with your supplier. It has nothing to do with set ups you make in the Pricing Administration work area. Set a value for the model and each option, but not the option classes. For details, see *Create Purchase Orders for Configured Items*.

8. Add a structure.

- Click **Structures**, then click **Actions > Create**.
- In the Create New Structure dialog, set the value, then click **OK**.

Attribute	Value
Name	Primary

- Click **Save**.

Add Classes to Your Model

1. Add the CTO_474100 class.
   - On the Edit Item page, on the Structures tab, in the Item Structures list, in the Name column, click **Primary**.
   - On the Edit Item Structure page, click **Actions > Select and Add**.
   - In the Select and Add dialog, search for the value.

Attribute	Value
Item	CTO_474100

   - Click the **row** that contains CTO_474100 in the search results, then click **OK**.
   - On the Edit Item Structure page, click the **row** that contains CTO_474000 in the Item column.
2. Repeat step 2 to add the memory class, CTO_474200.
3. Repeat step 2 to add the hard drive class, CTO_474300.
4. Click **Done**, then click **Save**.
5. Add a picture of your model.
   - On the Edit Item page, in the large empty area under the page name, under the text No items to display, next to the text None, click the **plus sign**.
   - In the Attachments dialog, click **Browse** to locate a file that contains an image of your model, click **OK**, then click **Save**.

Associate Your Model with an Inventory Organization

1. Add the association.
    - On the Edit Item page, click **Associations**.
    - Click **Actions > Select and Add**.
    - In the Select and Add dialog, search for an inventory organization, then click **Apply > Done**.

Attribute	Value
Organization	M1
Organization Name	Seattle Manufacturing Make sure the organization you choose as an inventory organization. If it isn't, then it won't display when you create the work definition. For details, see *Inventory Organizations*.

    - Click **Save > Save and Close**.
2. Create the structure in the M1 version from the structure you created in the V1 version.
    - On the Manage Items page, open the CTO_474000 (M1) version of your model for editing.

      Its typically the second row in the results.
    - On the Edit Item: CTO_474000y (M1) page, click **Structures**, then click **Actions > Create from Common**.

      You add the structure from the V1 version you created earlier. In this example, the V1 version is the master organization. This way, you use a single structure for all classes and options in the configured item, which helps manage and maintain the item. For example, if you update the structure in V1, then Product Information Management updates the structures that reference V1.

      Don't use **Actions > Create**. If you do, you will be creating a separate structure for the M1 version.

      Learn about master organizations. For details, see *Item Organizations*.
    - In the Create New Item Structure from Common dialog, set the values, then click **OK**.

Attribute	Value
Organization Name	Vision Operations
Item	CTO_474000
Structure Name	Primary
Common Structure Levels	All Levels

Attribute	Value

- In the Information dialog, click **OK**.
- Click **Save > Save and Close**.

Verify Your Set Up

1. Sign into Order Management and create a sales order.

Attribute	Value
Customer	Computer Service and Rentals

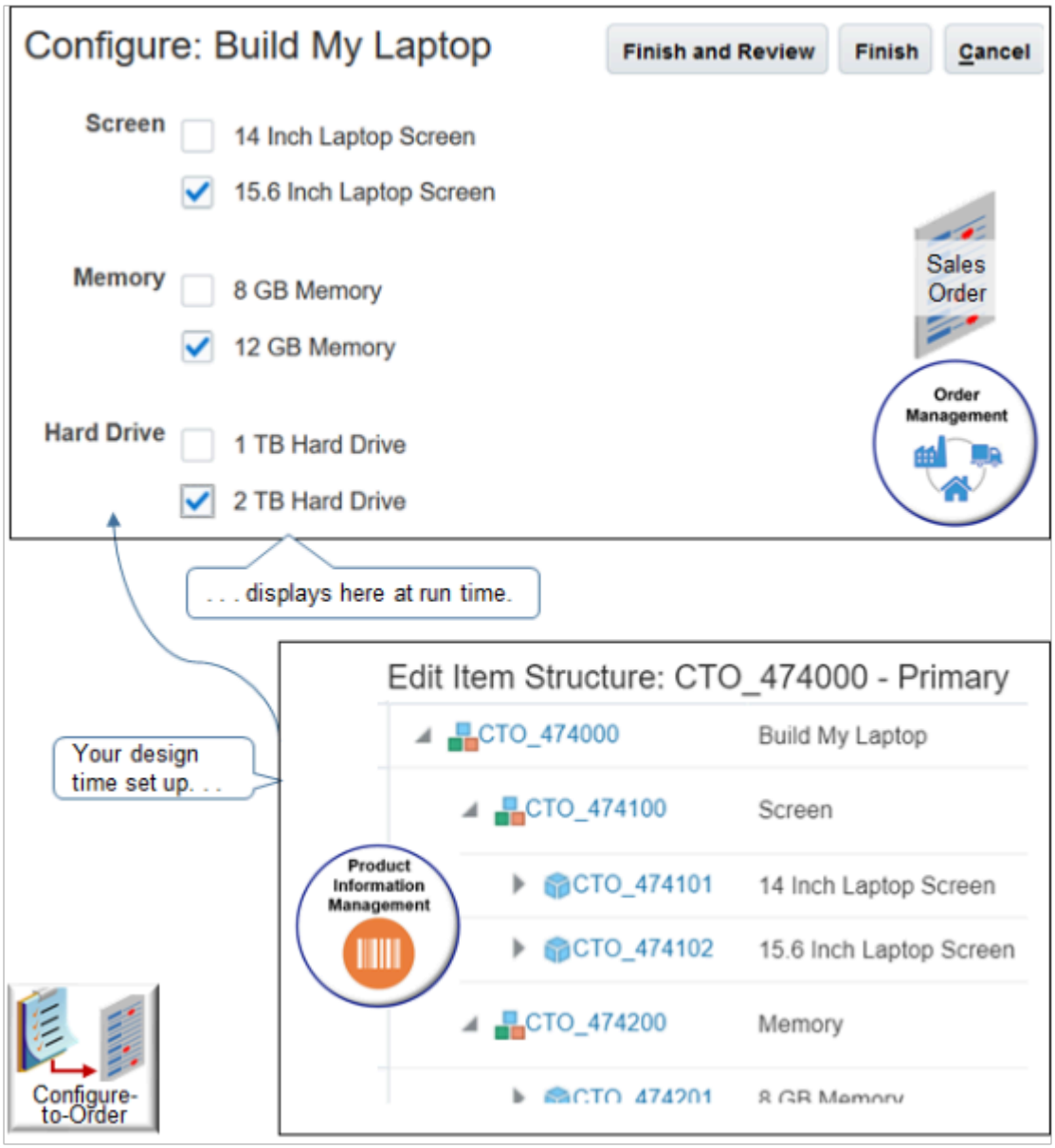
2. On the catalog line, enter the value.

Attribute	Value
Item	CTO_474000

3. Wait for the search results to display, then verify that the catalog line displays the Configure and Add button.

   Configure and Add displays only for a configured item.

4. Click **Configure and Add**, then verify the Configure page.



Note

- Verify that the hierarchy on the Configure page reflects the hierarchy you set up in Product Information Management.
- Verify that you can choose one, and only one configure option in each class.
- Notice there's no pricing. You set it up in a different topic. For details, see *Set Up Pricing for Your Configuration Model*.

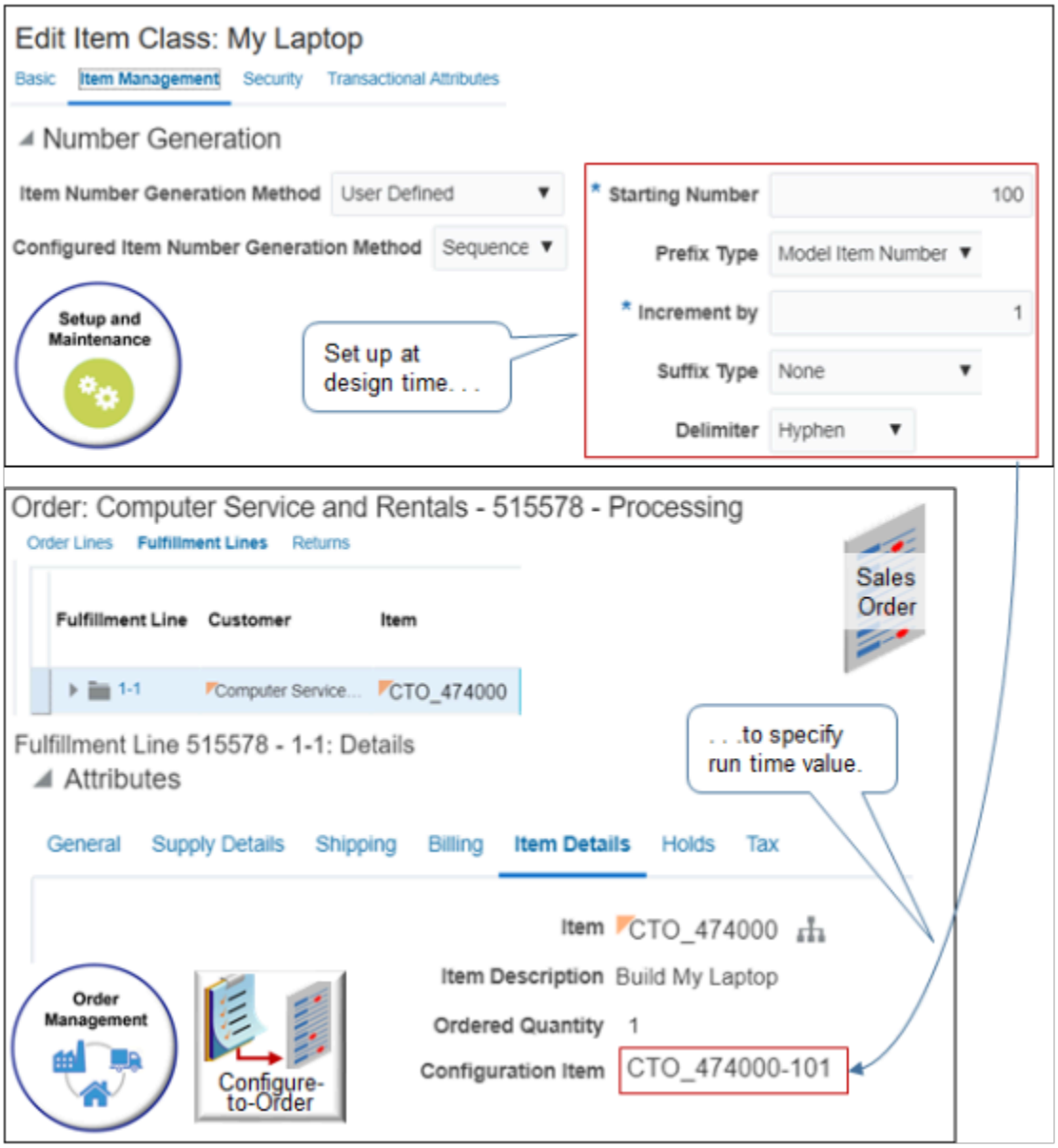
Related Topics

- Set Up Pricing for Your Configuration Model
- Inventory Organizations
- Item MRP and MPS Planning Specifications
- Item Organizations

Set Up Numbers for Your Configuration Model

Set up the item class so that the item name for each configured item is unique. A unique name can be useful to help track and manage each item.

Here's what your setup looks like after you complete the procedure.



Note

- You use the Edit Item Class page in the Setup and Maintenance work area to specify the number at design time.
- Order Management displays the number in the Configuration Item attribute on the sales order at runtime.

Let's say you need to add a suffix to the item name for each unique configuration of item CTO_474000. You need to start the suffix with the number 100, increment it by 1 each time a user orders the item, and use a hyphen (-) to separate the suffix from the name. For example, here's the name that the flow creates the first time the user adds the item.

```
CTO_474000-100
```

where

- CTO_474000 is the name of the configured item
- - (the dash) is the delimiter
- 100 is the starting number of your sequence

Here's the name for the next time a user adds the item.

```
CTO_474000-101
```

The number is unique across order lines, sales orders, and users.

Summary of the Setup

1. Get the name of the class.
2. Modify the class.
3. Verify your setup.

Get the Name of the Class

1. In the Product Information Management work area, click **Tasks > Manage Items**.
2. On the Manage Items page, search for CTO_474000, then open it for editing.

   This topic assumes you already created the CTO_474000. For details, see *Create Your Configuration Model*.
3. On the Edit Item page, note the value in the Item Class attribute.

   Assume the class for the CTO_474000 is named My Laptop.

Modify the Class

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Product Management
   - Functional Area: Items
   - Task: Manage Item Classes
2. On the Manage Item Classes page, in the search results list, search for, then open the My Laptop class for editing.
3. On the Edit Item Class page, click **Item Management**, then set the values.

Attribute	Value
Item Number Generation Method	User Defined

Attribute	Value
	This means you, the administrator, defines numbering. Not the end-user who creates the sales order.
Configured Item Number Generation Method	Sequence

4. In the Details area, set the values.

Attribute	Value
Starting Number	100
Prefix Type	Model Item Number Specify what displays before the starting number.
Increment By	1
Suffix Type	None Specify what displays after the starting number.
Delimiter	Hyphen Specify the character that separates the starting number from the prefix or the suffix.

5. Click **Save and Close**.

Verify Your Setup

1. Go to the Order Management work area and create a sales order.
2. Add the CTO_474000 item to order line 1.
3. Add the CTO_474000 item again, but this time add it to order line 2. Set the options for CTO_474000 on order line 2 different that the options you set for it on line 1.
4. Click **Submit**.
5. Click **Actions > Switch to Fulfillment View**.
6. On the Order page, click **Fulfillment Lines**.
7. Click the **row** that contains 1-1 in the Fulfillment Line column.
8. In the Attributes area, click **Item Details**, then notice that the value of the Configuration Item attribute is CTO_474000-100.
9. Click the **row** that contains 2-1 in the Fulfillment Line column.
10. Notice that the value of the Configuration Item attribute is CTO_474000-101.

You can see the number throughout the configure-to-order flow:

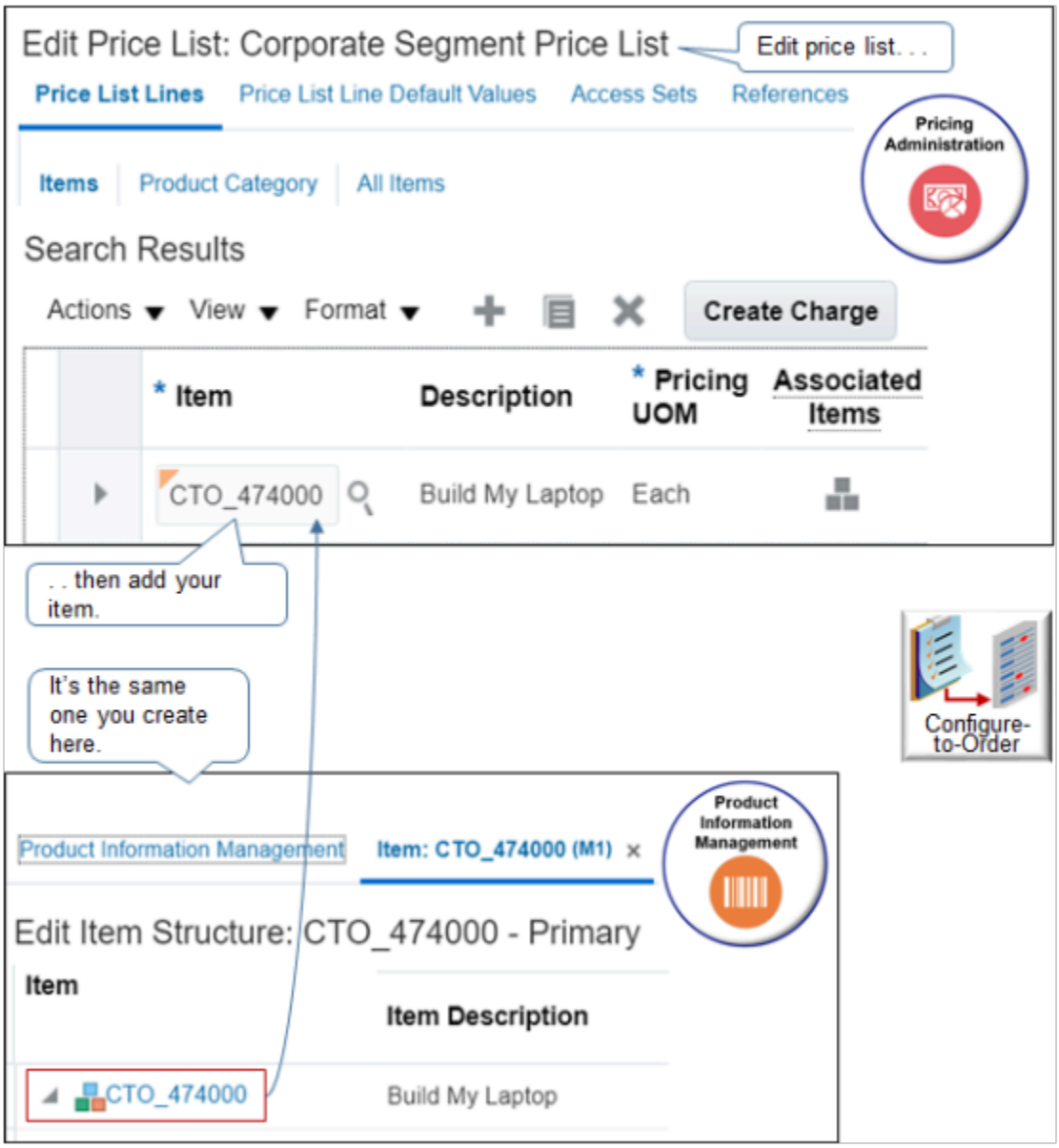
- In fulfillment views and the Availability Options area in the Order Management work area.
- In the Item column of the Supply Lines area of the Supply Order Details page in the in the Supply Orchestration work area.
- In the Item column of the Review Dispatch List in the Inventory Management work area.

Set Up Pricing for Your Configuration Model

Use the Pricing Administration work area to set up pricing for your configuration model.

Suppose you must add your new model to the Corporate Segment Price List, and you already set up Pricing so it uses this price list for your Computer Service and Rentals customer.

You use the Pricing Administration work area to set up pricing for the configuration model that you create in the Product Information Management work area.



Summary of the Setup

1. Manage your price list.
2. Verify your set up.

This topic describes how to set up a configuration model that has a number of items and charges. For guidelines and a more simple example, see *Set Up Pricing for Configuration Models*.

This topic uses example values. You might need different values, depending on your business requirements.

Manage Your Price List

1. Sign in with pricing administrator privileges.
2. Go to the Pricing Administration work area.
3. Click **Tasks > Manage Price Lists**. Learn about price lists. For details, see *Manage Price Lists*.
4. On the Manage Price Lists page, search for Corporate Segment Price List, then open it for editing.

5. Add the charge for each item.
    - On the Edit Price List page, in the Search Results area, click **Actions > Add Row**, then search for the item.

Attribute	Value
Item	CTO_474000

    - Wait for the search results to display the description, then set **Pricing UOM** to Each.
    - Click **Create Charge**, scroll down, set the values, then click **Save**.

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	100
Start Date	Choose the current date.

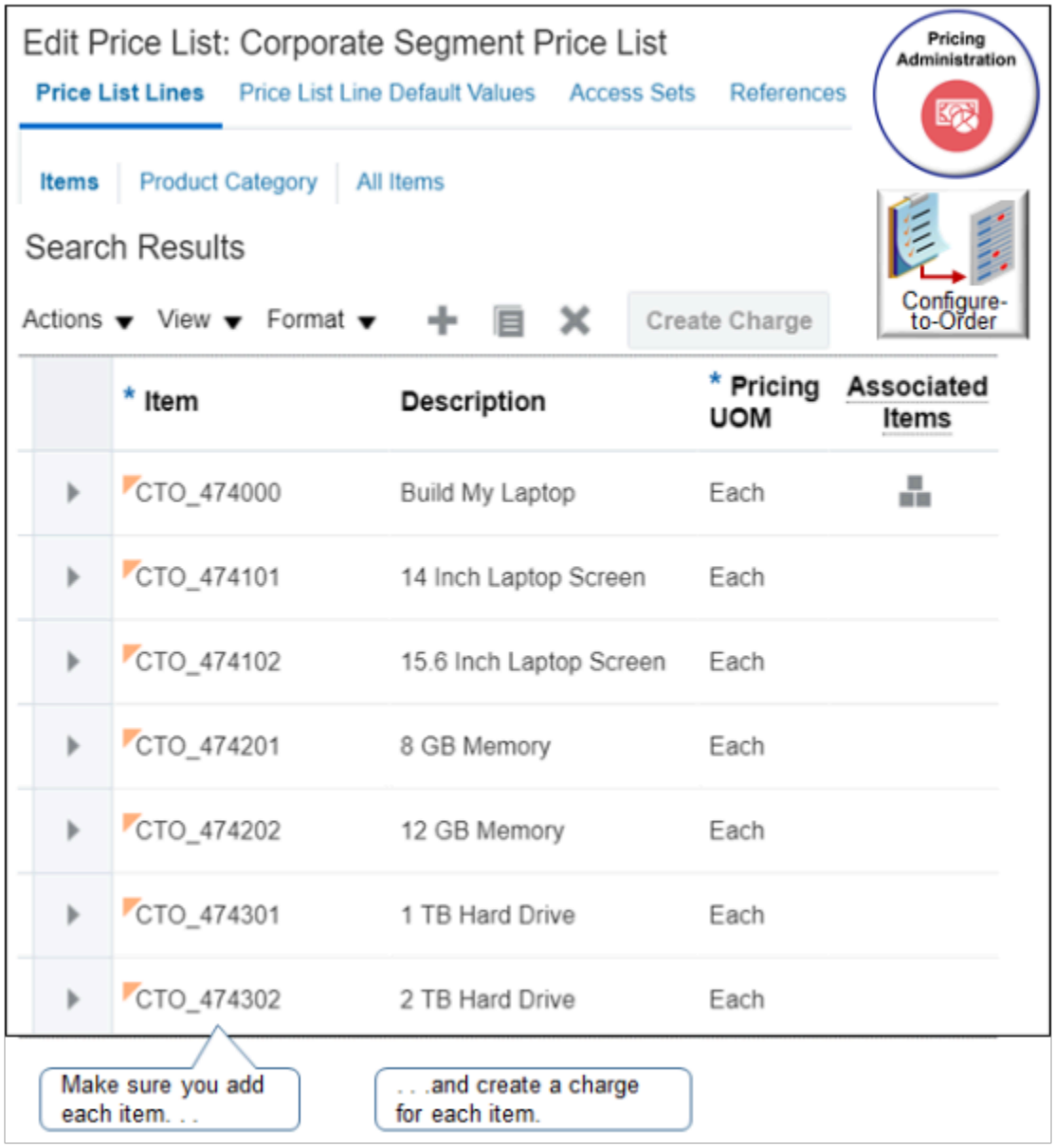
    - Repeat this step for each charge.

      Here's the pricing you set up when you visualized the hierarchy. For details, see *Visualize the Hierarchy for Your Configuration Model*. Use the same values for each attribute except for Base Price.

Item	Description	Base Price
CTO_474100	Build My Laptop	100
CTO_474101	14 Inch Laptop Screen	200
CTO_474102	15.6 Inch Laptop Screen	300
CTO_474201	8 GB Memory	5
CTO_474202	12 GB Memory	10
CTO_474301	1 TB Hard Drive	50

Item	Description	Base Price
CTO_474302	2 TB Hard Drive	100

Your set up should look like this.

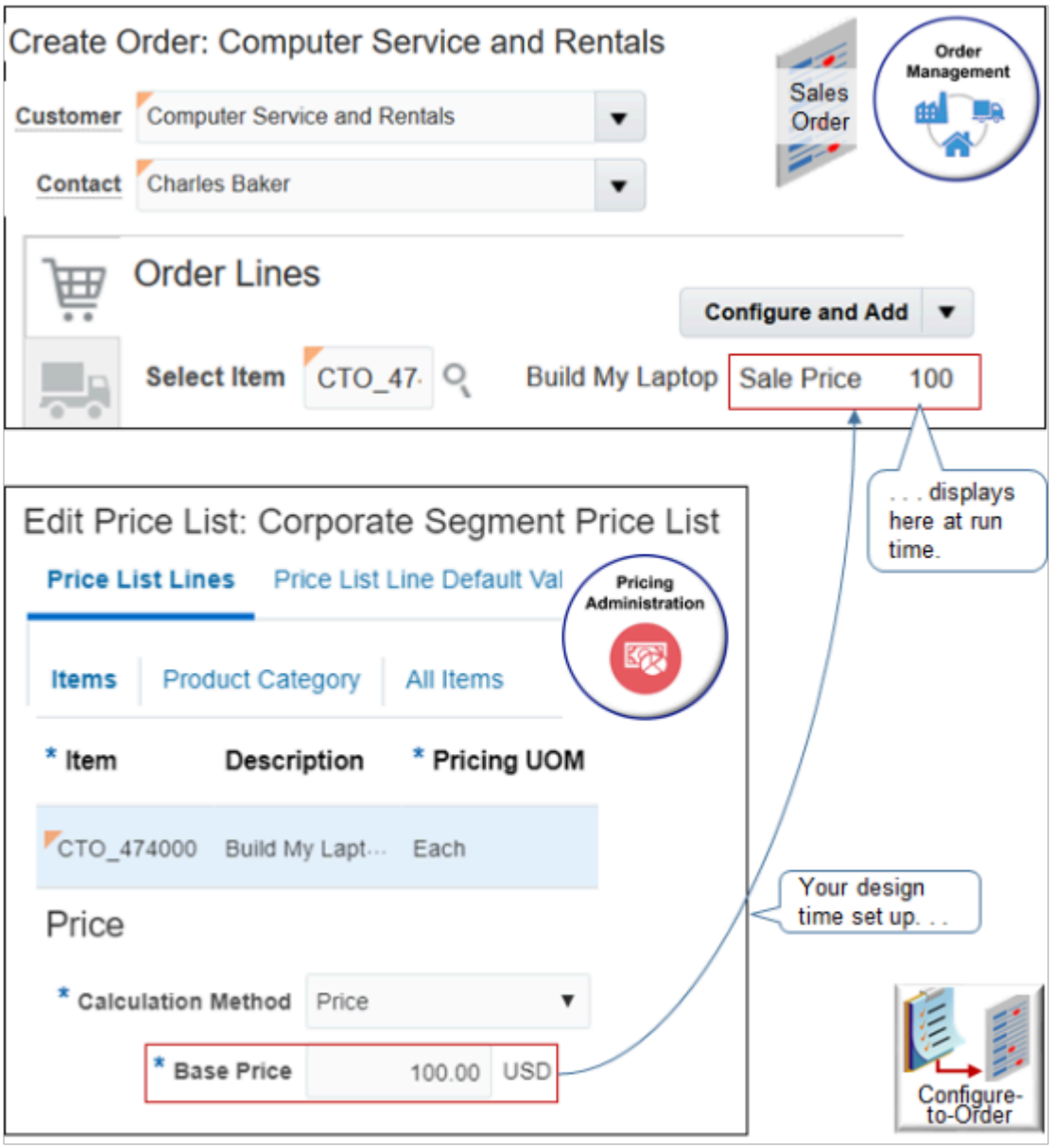


6. Save your changes.

Verify Your Set Up

1. Use another browser to sign into Order Management, create a sales order, then search for CTO_474000 on the catalog line.

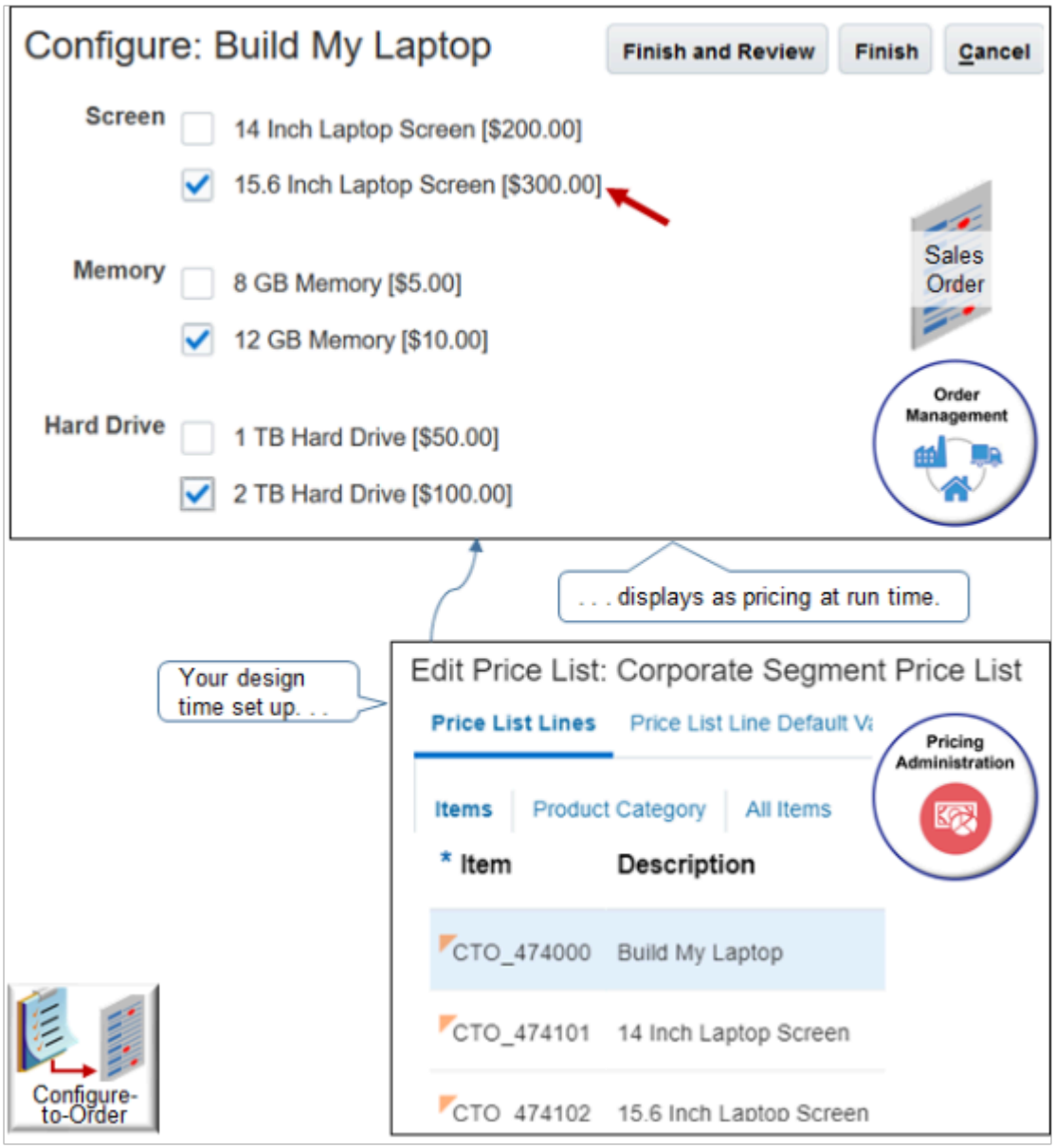
   Notice that your design time set up on the Edit Price List page in Pricing Administration displays on the order line of the sales order in Order Management at runtime.



2. Verify that the catalog line in Order Management displays the base price you set for the CTO_474000 in Pricing Administration, which is $100.

3. Click **Configure and Add**, then verify that each item contains the pricing you set up in Pricing Administration.

   Notice that your design time set up on the Edit Price List page in Pricing Administration displays on the Configure page of the sales order in Order Management at runtime.



   For example, verify that the price for the 15.6" screen is $300.

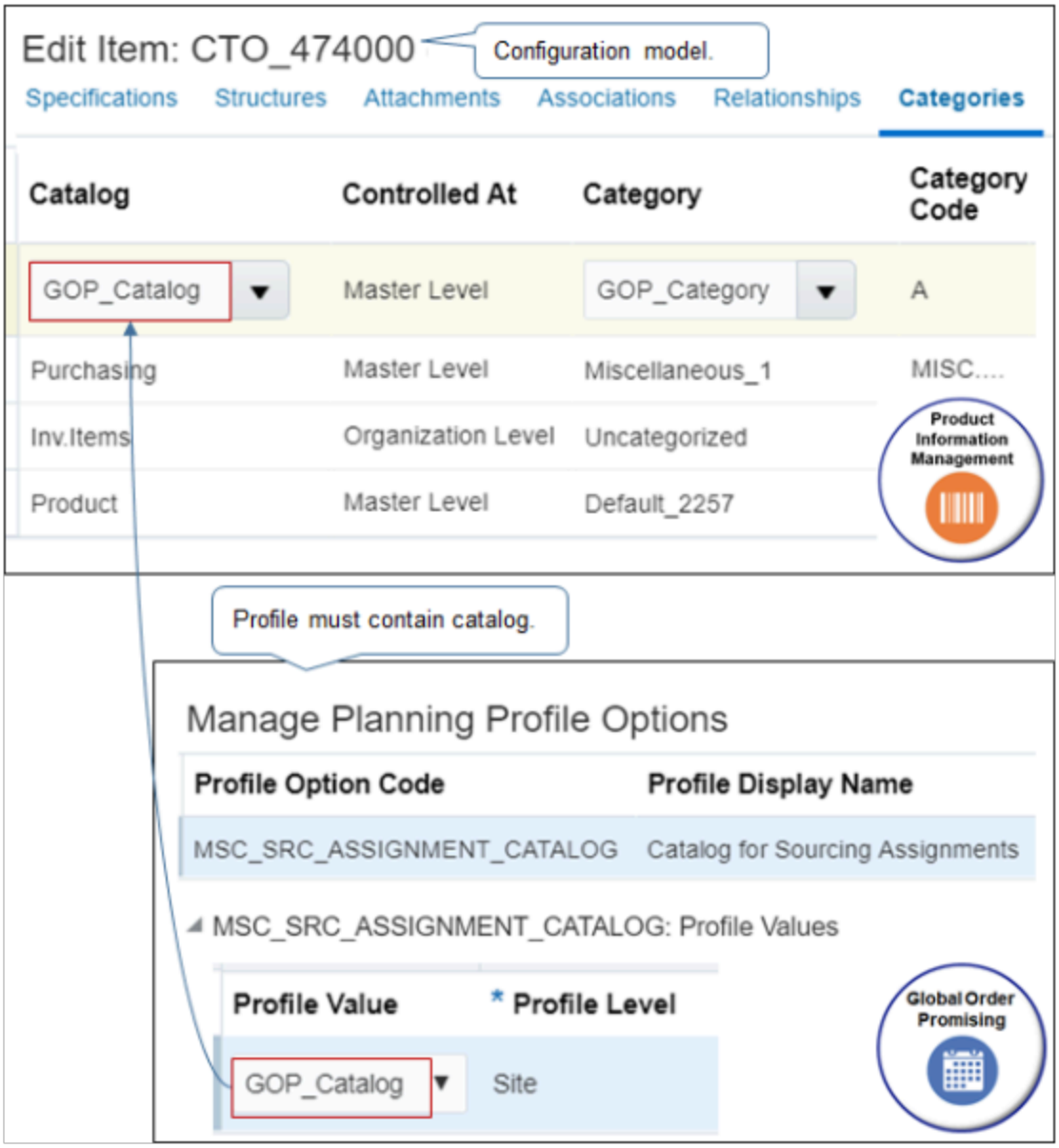
4. Click **Finish and Review**, verify the summary displays the correct pricing for your choices and calculates the correct total, then click **OK**.
5. Click **Submit** to send your sales order to order fulfillment.

Related Topics

- Overview of Setting Up Configuration Models
- Visualize the Hierarchy for Your Configuration Model
- Manage Price Lists
- Set Up Pricing for Configuration Models

Specify Catalog and Category for Your Configuration Model

A catalog is a collection of categories you use to classify your model and organize it in a hierarchy. Product Information Management associates each item it creates with the categories you set for the model.



Global Order Promising uses the category when it runs the available-to-promise rule for your model. The category helps Promising to recognize the configured item.

- You must assign at least one catalog that the Catalog for Sourcing Assignments profile in Promising contains. Supply Chain Orchestration sends a request to Product Information Management to create the configured item, then copies the catalog to the item when Product Information Management creates the item.
- In this example, Global Order Promising uses the predefined GOP_Category (Global Order Promising) to assign the configured item to the rule.
- Make sure you assign each component in your model to an available-to-promise rule or associate it with a catalog that you assign to an available-to-promise rule.
- If you don't set up the catalog and category correctly, the flow will usually create the configured item, but other downstream processing might fail, such as scheduling. For details, see *Troubleshoot Problems with Configure-to-Order*.
- The Product Information Management work area might automatically add catalogs and categories to your model depending on how you set up the work area. It might still be necessary to add a catalog to support your configure-to-order flow.

For details, see *Relationship Between Categories and Catalogs*.

Try it.

1. Identify the category you must use.
    - Go to the Global Order Promising work area.
    - Click **Tasks > Manage Planning Profile Options**.
    - On the Manage Planning Profile Options page, search for the value.

Attribute	Value
Profile Option Code	MSC_SRC_ASSIGNMENT_CATALOG

    - In the search results, in the Profile Values area, note the value.

Attribute	Value
Profile Value	GOP_Catalog This attribute identifies the catalog that Promising uses when assigning an assignment set. You specify this value in the Catalog attribute on your model in Product Information Management. If you change the Profile Value, then you must specify your value on the model. For example, if you set Profile Value to my_value, then you must set the Catalog attribute on your model to my_value. GOP_Catalog is an example. You can add a catalog in the Profile Values list and use it instead of GOP_Catalog.

2. Add the category to your item.
    - In the Product Information Management work area, click **Tasks > Manage Items**.
    - Search for, then open your model, such as CTO_474000 for editing.
    - On the Edit Item page, click **Categories**.
    - Click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Catalog	GOP_Catalog
Category	GOP_Category

3. Collect data and refresh the server. For details, see *Collect Planning Data for Your Configuration Model*.

   Here are the catalogs that the flow copies to the configured item.
    - Inventory
    - Purchasing
    - Planning
    - Cost
    - Order Entry
    - Product Line Accounting
    - Asset Management
    - Distributed Order Orchestration
    - Order Capture
    - Pricing
    - Configurator
    - Supply Chain Financial Flow Orchestration

Related Topics

- Relationship Between Categories and Catalogs

Collect Planning Data for Your Configuration Model

Collect planning data at various points of your set up process, and also after you finish set up.

Collect Setup Data

Collect setup data each time you.

- Create or modify a model.
- Modify the item, item structure, catalog, or work definition.

- *Collect Planning Data for Order Management*
- Set up a new item in the Product Information Management work area.

Collect data. For details, see *Collect Planning Data for Order Management*.

Attribute	Value
Items Item Structures Work Definitions	You must include these entities for each model.

Collect Runtime Data

Periodically collect data for each new configured item your users add when they create a sales order. Collection gets the on-hand quantity for return orders and canceled order. The quantity affects planning. Specify attributes when you do the Supply Planning Data task.

Attribute	Value
On Hand	Promising uses this entity to get the inventory that's in stock for the model. If all components in the configured item aren't in stock, then Promising won't find availability for the model. For example, assume the schedule requires you to build the item within five days to meet the delivery date. Promising examines the build schedule for each component, the current backlog of purchase orders for component x, and determines component x isn't available for your order until the sixth day.
Purchase Orders and Requisitions Transfer Orders Work Order Supplies	Add these entities so Global Order Promising can get availability across the entire supply chain.

Don't use templates that the dialog displays, such as Dynamic Data for Supply Planning. The templates don't apply for configure-to-order.

Make sure you refresh the server.

Related Topics

- Overview of Collecting Promising Data for Order Management

Create Promising Rules for Your Configuration Model

Use an available-to-promise (ATP) rule and sourcing rule to promise your configured item in different ways.

At runtime, Order Promising determines the lead time for the model and components, depending on the option classes and options your customer chooses. Here's how Promising does it.

- Examines various paths in the item structure and determines the longest path.

- Considers fixed and variable lead times for items across the structure. The lead time for the model influences order promising behavior. For example, if a sales order requests a model but no supply exists, then Promising makes sure it promises the sales order only on or after the model lead time.

Set up Order Promising to promise a pick-to-order model.

- Make sure you set the Ship Model Complete attribute to Yes on the model in Product Information Management.
- Make sure all components will be available on a specific date in the specified warehouse so Promising can promise the model for the date.
- If a delay occurs in receiving a component in the warehouse at some later time, then a person in shipping might decide to ship only the parts of the configured item that are available. You might need to do more processing to ship the remaining part of the configured item that doesn't involve promising.

This section describes how to set up rules for a configured item. For details, see *Assignments and Promising Rules*.

Assume you must create an available-to-promise rule for the CTO_474000 model. For details, see *Create Your Configuration Model*.

Try it.

1. Go to the Global Order Promising work area, then click **Tasks > Manage ATP Rules**.
2. On the Manage ATP Rules page, click **Actions > Create**, then set the values.

Attribute	Value
Name	ATP rule for Build My Laptop
Promising Mode	Choose a value. - **Supply Chain Availability Search**. Consider supply for the configuration components in your model when promising availability. Choose this value for most configured items. You must use this value for a back-to-back flow. - **Lead Time Based**. Use this value when the supply chain for your item is reliable and predictable. - **Infinite Availability Based**. Use when your item is always available.
Search Components and Resources	Enable it. Allow Promising to consider components and resources when promising an assemble-to-order model. If you source the item through a back-to-back flow, then you must enable this option. Enabling produces the planning recommendation that the flow needs to create the supply order.
User Defined Fence in Days	Specify the point in time when the flow can consider that the configured item is always available. To make sure processing for an advance order is timely, set your time fence to a lead time that exceeds your typical lead time.

For this example, enable all attributes in the Supply Types area and Demand Types area. Global Order Promising will consider each type you chose when it analyzes your supply chain.

In your actual deployment, disable attributes you know aren't viable.

3. Click **ATP Rule Assignment**, click **Actions > Add Row**, then set the values. Add a separate row for the model, for each option class, and each configure option.

Assignment Basis	Assigned-to Item
Item	CTO_474000
Item	CTO_474101
Item	CTO_474102
Item	CTO_474201
Item	CTO_474202
Item	CTO_474301
Item	CTO_474302
Item	CTO_474100
Item	CTO_474200
Item	CTO_474300

Note

- You must assign a rule to the model, each option class, and each configure option in your model at some hierarchical level.
- You typically assign at the Item level.
- You typically use the same rule for model, classes, and options. You can use different rules, but using the same rule simplifies set up, maintenance, and troubleshooting.

4. Click **Actions > Add Row**, then set the values.

Assignment Basis	Assigned-to Item Category
Category	GOP_Category

Note

- This step assigns your rule at the Category level so the same rule can recognize each instance of your configured item that Product Information Management creates at runtime, such as CTO_474000-100 and CTO_474000-101.
- Supply Orchestration associates the configured item that it creates at runtime with the same category you set for the configuration model in Product Information Management at design time. Downstream processes also use the same catalog at runtime.

Related Topics

- Create Your Configuration Model
- Overview of Setting Up Configuration Models
- Set Up Promising Rules and Sourcing Rules for Order Management
- Assignments and Promising Rules

Create Sourcing Rules for Your Configuration Model

Create a sourcing rule to specify where and under what circumstances to make your model and its components available to your customer.

Assume you need a sourcing rule that sources the Build My Laptop model from Seattle Manufacturing. Seattle Manufacturing stocks the required components and the configure options. Their job is to assemble the components according to the choices that your user makes on the sales order.

Try it.

1. Go to the Global Order Promising work area, then click **Tasks > Manage Sourcing Rules**.
2. On the Manage Sourcing Rules page, Click **Actions > Create**, then set the values.

Attribute	Value
Name	Sourcing rule for Build My Laptop
Organization Assignment Type	Local Set to: - Global when you must specify where to fulfill and ship the sales orders. You don't specify an organization to create supply. Instead, you specify a transfer or buy source. Use buy for a drop ship supplier. - Local when you must specify how to create supply and the organization that creates it.
Organization	Seattle Manufacturing

Attribute	Value

3. In the Sourcing Rule Effective Dates area, click **Actions > Add Row**, then set the start date.
4. In the Effective Start Date area, Click **Actions > Add Row**, then set the values.

Attribute	Value
Type	Make At Set to Make At or Buy From for an assemble-to-order model, such as the Build My Laptop model. You typically make an assemble-to-order item instead of keeping it in inventory. So, you usually don't use Transfer From for the model. Use Transfer From for a made-to-stock item, such as a component. For example, you don't stock the laptop, but you probably would stock screens for the laptop. Global Order Promising enables Make At only when you set the assignment type to Local.
Supplier Supplier Site Supplier Site Source System	If you create a global rule for a Buy From sourcing type, then you must specify values for these attributes. If your supply chain includes more than one supplier, then create a separate sourcing rule for each supplier. Add a separate row for each supplier.
Allocation Percent	100 In this example, you add only one row, so specify 100%. If you add more than one row, then you can allocate demand across sources. For example, if you add a row for Vision Manufacturing and set allocation to 70%, add another row for Vision Distribution and set allocation to 30%, then Promising will use Vision Manufacturing to promise 70% of the orders.
Rank	1 If you add more than one row, then you can specify the rank order to use for sources. For example, if you add a row for Vision Manufacturing and set Rank to 1, add another row for Vision Distribution and set Rank to 2, then Promising will use Vision Manufacturing to promise the order first. If Promising determines that Vision Manufacturing can't fulfill the order, then Promising will consider Vision Distribution.
Shipping Method	Global Order Promising disables Shipping Method for a local rule. You can't edit it. Leave it empty for a global rule.

Attribute	Value
	If you set a value for a global rule, you might get an error. `The value provided for the Shipping Method attribute is invalid.` Don't set shipping method in this context because it specifies where supply originates, not how to ship it to the customer.
Exclude for Options and Option Classes	Exclude options and option classes when promising a sales order. Exclude them for a Make At or a Buy From sourcing type. For example, exclude an item from planning when you know your source can't make it because it includes toxic chemicals that the source isn't authorized to handle, or your company limits production to only one specific site. For another example, assume you know Seattle Manufacturing created a large oversupply of the CTO_474100 screen option class from a prior marketing campaign. You already know supply is available. To improve planning performance, you decide to exclude it from planning.

5. Click **Save > Save and Close**.
6. Assign your sourcing rule.
    - Click **Tasks > Manage Assignment Sets**.
    - On the Manage Assignment Sets page, click **Actions > Create**.
    - On the Create Assignment Set page, set the values.

Attribute	Value
Name	Assignment Set for Sourcing Rules
Catalog	GOP_Catalog Use the same catalog that you use for your model in the Product Information Management work area and in your available-to-promise rule.

7. Click **Actions > Add Row**, then set the values.

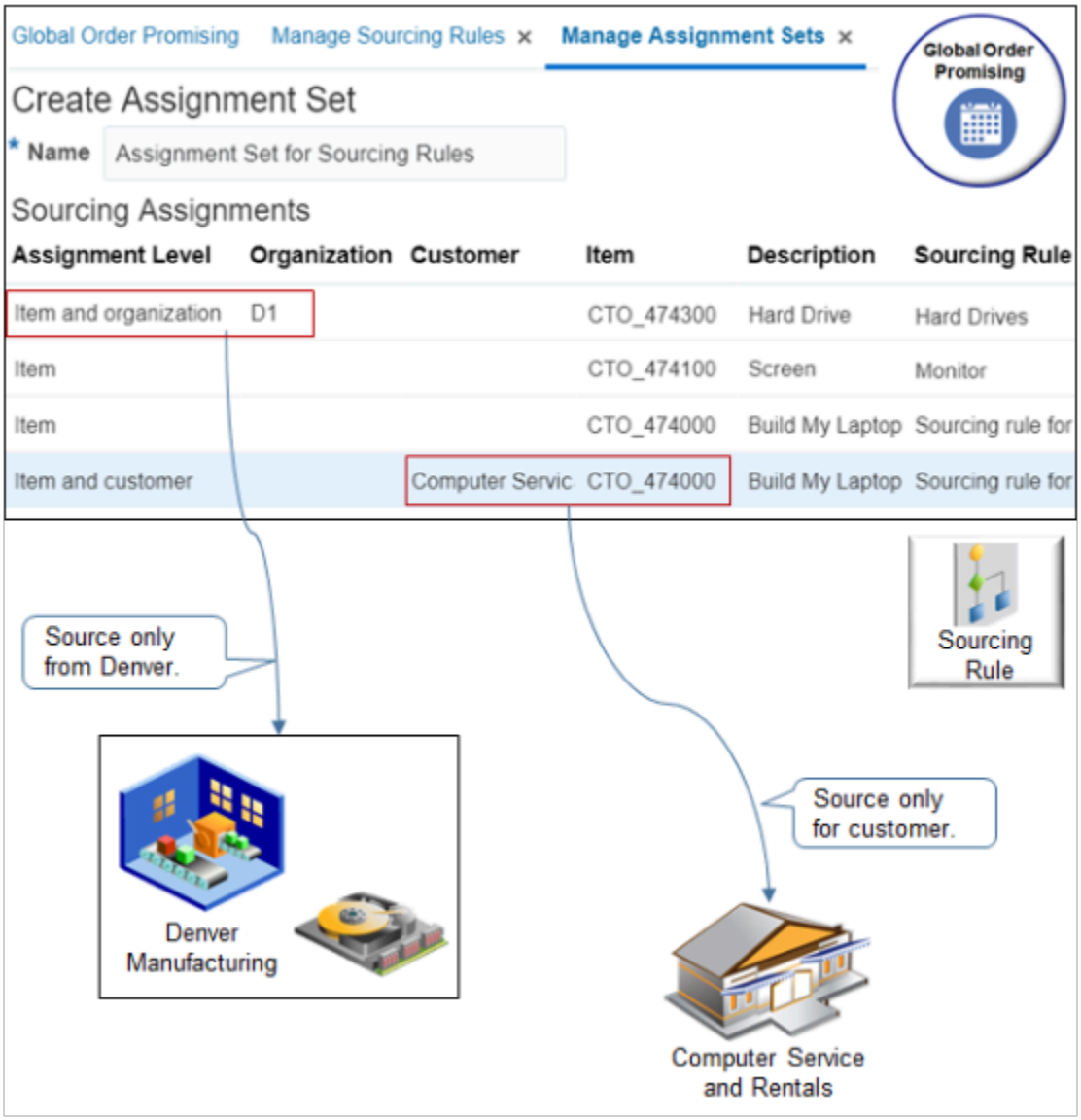
Attribute	Value
Assignment Level	Item Note - Promising fulfills your sales order only from the source that you assign to the assignment set.

Attribute	Value
	◦ Assign at least one sourcing rule at the global level so Global Order Promising can use it to identify a ship-from location. If you don't, then your users must specify the warehouse in every sales order. ◦ You must use the same ship-from and source in a configure-to-order make flow.
Item	CTO_474000 Assign your model. In this example, CTO_474000 is the Build My Laptop model.
Sourcing Type	Sourcing Rule
Sourcing Rule or Bill of Distribution	Sourcing rule for Build My Laptop

For details, see *Sourcing Rule*.

Add More Assignments

You can add more than one sourcing rule and assign them differently. Global Order Promising offers you a wide range of choices.



For example, assume you create three different sourcing rules and assign them differently according to item, organization, and customer.

1. Assign the CTO_474000 Build My Laptop item, which is the model, to Sourcing rule for Build My Laptop.
2. Assign the CTO_474000 to Sourcing rule for Build My Laptop only if the customer is Computer Service and Rentals.
3. Assign the CTO_474300 Hard Drive, which is an option class, to the Hard Drives sourcing rule, which is a rule you set up that's optimized to source hard drives.
4. Assign the CTO_474100 Screen, which is an option class, to the Monitor sourcing rule, which is a rule you set up that's optimized to source monitors. Source the CTO_474100 only from D1 Denver Manufacturing when running the sourcing rule.

If you add more than one assignment, then Global Order Promising considers the sourcing rule you set for the most detailed assignment level first. Here's a list of assignment levels that you can use, listed from least detailed to most

detailed. For example, Region is more detailed than Global. If you add a global assignment and a regional assignment, then planning runs the regional assignment first, then the global assignment.

- Global
- Region
- Demand class
- Customer
- Customer and customer site
- Category
- Category and region
- Item
- Category and demand class
- Category and customer
- Category and customer site
- Item and region
- Item and demand class
- Item and customer
- Item and customer site

For details, see *Set Your Assignment Levels* and

Consider Your Sourcing Hierarchy and Assignment Set Hierarchy.

Guidelines

Make sure you:

- Assign an ATP rule to each option, option class, required item, and each item in the model's structure.
- Collect the model's structure and all of the structure's child items.

Troubleshoot Problems with Configure-to-Order

Troubleshoot problems that happen during setup or at runtime in a configure-to-order flow.

Problems That Happen During Setup

Trouble	Shoot
I can't find an item when I create a work definition. I search for my item in the Create Work Definition dialog, but the search can't find it, or parts of my structure are missing in the Work Definition work area. For example, I create an applicability rule, but the rule editor doesn't display a configure option, or I manually add rule text and encounter an error. Assume you set up a hierarchy in Product Information Management.	The Work Definition work area only recognizes items that you associate with an inventory organization. Make sure you add an inventory organization on the Associations tab in Product Information Management for the model, each option class, and each configure option. For details, see these topics. • Create Your Configuration Model • Inventory Organizations

Trouble	Shoot

• CTO_474000 is the model (computer)
  ○ CTO_474300 is an option class (hard drive)
  - CTO_474301 is a configure option (1TB hard drive)
You associate CTO_474000 and CTO_474300 with an inventory organization, but don't associate CTO_474301 with an inventory organization.
 	
You then manually add an applicability rule.
 	
ITEM='CTO_474000'.'CTO_474300'.'CTO_474301'
 	
But when you click Validate, you get an error.
 	
`The rule contains an invalid item CTO_474301`	

I encounter an error when I use the Create Work Definition dialog. `For Assemble to Order model CTO_474000, Primary item structure is not available.` The structure isn't associated with an inventory organization.	Use the Associations tab in Product Information Management to add your inventory organization, open the new version that Product Information Management creates, then use the Structures tab to add your structure to the new version. If you add the structure first, Product Information Management creates a new version of your item, appends (M1) to it, but doesn't copy the structure from the V1 version to the M1 version, and you must add the structure again to M1.
I encounter an error when I create an applicability rule. `Attribute RuleText in RuleEO is required.` I create an applicability rule, validate it, and close the rule editor. I then click Save to save the work definition, but encounter an error.	This error sometimes occurs if you navigate away from the rule editor and come back to it, or add and remove rule text several times. Click Cancel on the Edit Work Definition Details page, reopen the work definition, then add your rule.
The Product Information Management work area doesn't delete my items. I create items in the Product Information Management work area. During testing, I find I need to revise some and delete others. I search for my item on the Manage Delete Groups, click **Actions > Delete**, add it to a delete group, but it persists in the work area.	Try this. 1. Click **Tasks > Manage Delete Groups**. 2. On the Manage Delete Groups page, search for and open your delete group. 3. Select your item, click **Actions > Delete**, then click **Submit**. 4. Wait a few minutes for the delete to take effect. For details, see *Group Deletions of Items, Structures, New Item Requests, and Change Orders*.
The Manage ATP Rules page doesn't contain my item.	Collect and refresh. For details, see *Collect Planning Data for Your Configuration Model*.

Trouble	Shoot
I add a catalog to my model. I search the Manage ATP Rules page for my item but the search doesn't return anything.	
I receive a warning when I use the Plan Inputs page to collect data. `You requested net change collections, but a prior targeted collection was not performed for one or more of the selected entities. Do you want to continue?`	In the Collect Planning Data dialog, change the Collection Type to Automatic Selection to let the server choose the data to collect. You can use Net Change the next time you collect data.
Global Order Promising keeps giving available-to-promise recommendations. I test my make or buy setup, but Promising keeps giving available-to-promise recommendations.	This problem typically occurs in a test environment where your flow isn't finished yet or Promising isn't refreshed. Promising identifies on-hand quantities for the item and uses them to source supply until you exhaust the quantities. Examine and correct your inventory quantities, run collections, then run the *Refresh and Start the Order Promising Server* scheduled process.
I add an extensible flexfield to the parent of my configuration model. I also add an extensible flexfield to each child. However, if the user creates a revision, then Order Management copies the value from the flexfield on the parent into the flexfields on the children, replacing whatever data was in the flexfields on the children. We don't want this overwrite to happen.	Write a processing constraint on each extensible flexfield so the constraint prevents Order Management from updating the child flexfields during a revision.

Problems That Happen During Import

Manual Price Adjustment

I import a manual price adjustment on the child line of a model, but the adjustment doesn't show up in the Order Management work area, and I can't use the work area to change the adjustment.

Try this:

- Apply a manual price adjustment only on the model's root line.
- Don't apply a manual price adjustment on a child line.
- Locate the manualPriceAdjustments entity on the root line of your import payload, then set the ChargeRollupFlag attribute to `true` in that entity.

Import Can't Find Part of a Configured Item

You might encounter an error where your import can't find part of a configured item, such as a configuration node. For example, you encounter an error.

```
Cannot find a matching configuration node for item 12345
```

A configuration node is part of a configured item. A configure option is an example of a configuration node. This error might happen even if the node exists in the structure of a configured item that includes an instance type of Optional Single Instance. The order import process does a search that validates and creates the structure that it includes in the sales order even if a node, such as Option Classes, is missing in the order import data that defines the structure.

However, if the root of the configured item contains a reference model that you set up as Optional Single Instance, or if it's part of an option class, and if the order import data doesn't include the absolute path to the node, then the search won't find the node.

To avoid this problem, use the Product Information Management work area to modify the instance type of the structure from Optional Single Instance to Required Single Instance. Your modification won't affect functional behavior. You must also set the Use Configurator for Order Import Validation parameter to Yes. For details, see *Manage Order Management Parameters*.

Assume the order import data includes M1, M1.M2, M1.M2.SI2, and the Product Information Management includes this structure for a configured item:

```
M1
|_M2 (Optional Single Instance)
 |_OC1
 |_SI2
```

The order import search won't find the complete structure, it will create an error during order import, and it will add an entry in the Order Import log that's similar to this entry.

```
The order import process failed
 for source order source_order_identifier for the
 following reason: Cannot find a matching configuration node for item item_number on order line
 number order_line_number.
```

where

- *source_order_identifier* identifies the source order
- *item_number* identifies the item
- *order_line_number* identifies the order line

This entry indicates that the search couldn't find a matching configure option in the Optional Single Instance node.

For another example, assume the order import data includes M1, M1.M2, M1.M2.SI1, and the Product Information Management contains this structure.

```
M1
|_OC1
 |_M2 (Optional Single Instance)
 |_SI1
```

The order import search will fail in the same way it failed in the first example.

Explicitly Define Each Node

If the same option item occurs more than one time in your model's hierarchy, then you must explicitly define each node.

Assume you have this hierarchy:

```
Stove
 Oven
 10K BTU Gas Burner
 20K BTU Gas Burner
 Stovetop
 10K BTU Gas Burner
```

```
20K BTU Gas Burner
```

You have the same 10K BTU Gas Burner option in two different option classes, once in the Oven option class and again in the Stovetop option class. It's important that you explicitly describe each of these nodes in your hierarchy even though the item itself is the same in both classes.

Assume you use the value 10029041 to identify the 10K BTU Gas Burner:

Node	Code
Stove	<ns1:RootParentLineReference>1</ns1:RootParentLineReference>
Oven	<ns1:ParentLineReference>1</ns1:ParentLineReference>
10K BTU Gas Burner	<ns1:ProductNumber>10029041</ns1:ProductNumber>
Stovetop	<ns1:ParentLineReference>2</ns1:ParentLineReference>
10K BTU Gas Burner	<ns1:ProductNumber>10029041</ns1:ProductNumber>

Here's what your payload should look like:

```
<ns1:Line>
 <ns1:SourceTransactionLineIdentifier>6</ns1:SourceTransactionLineIdentifier>
 <ns1:ProductNumber>10029041</ns1:ProductNumber>
 <ns1:ProductDescription/>10K BTU Gas Burner</ns1:ProductDescription>
 <ns1:OrderedQuantity>1</ns1:OrderedQuantity>
 <ns1:OrderedUOMCode>EA</ns1:OrderedUOMCode>
 <ns1:ParentLineReference>1</ns1:ParentLineReference>
 <ns1:RootParentLineReference>1</ns1:RootParentLineReference>
<ns1:Line>
<ns1:Line>
 <ns1:SourceTransactionLineIdentifier>30</ns1:SourceTransactionLineIdentifier>
 <ns1:ProductNumber>10029041</ns1:ProductNumber>
 <ns1:ProductDescription/>10K BTU Gas Burner</ns1:ProductDescription>
 <ns1:OrderedQuantity>1</ns1:OrderedQuantity>
 <ns1:OrderedUOMCode>EA</ns1:OrderedUOMCode>
 <ns1:ParentLineReference>2</ns1:ParentLineReference>
 <ns1:RootParentLineReference>1</ns1:RootParentLineReference>
<ns1:Line>
```

Note

- ParentLineReference identifies each option class.
- RootParentLineReference identifies the parent of the option class. That parent is the Stove, which is the top node in the model.
- This example includes only a small part of the payload. For details about the entire payload that you actually need, see *Structure Your Configured Item's Payload*.

Here's an example that would fail:

```
<ns1:Line>
 <ns1:SourceTransactionLineIdentifier>6</ns1:SourceTransactionLineIdentifier>
 <ns1:ProductNumber>10029041</ns1:ProductNumber>
 <ns1:ProductDescription/>10K BTU Gas Burner</ns1:ProductDescription>
 <ns1:OrderedQuantity>1</ns1:OrderedQuantity>
 <ns1:OrderedUOMCode>EA</ns1:OrderedUOMCode>
 <ns1:ParentLineReference>1</ns1:ParentLineReference>
 <ns1:RootParentLineReference>1</ns1:RootParentLineReference>
<ns1:Line>
```

```
<ns1:Line>
 <ns1:SourceTransactionLineIdentifier>30</ns1:SourceTransactionLineIdentifier>
 <ns1:ProductNumber>10029041</ns1:ProductNumber>
 <ns1:ProductDescription/>10K BTU Gas Burner</ns1:ProductDescription>
 <ns1:OrderedQuantity>1</ns1:OrderedQuantity>
 <ns1:OrderedUOMCode>EA</ns1:OrderedUOMCode>
 <ns1:ParentLineReference>1</ns1:ParentLineReference>
 <ns1:RootParentLineReference>1</ns1:RootParentLineReference>
<ns1:Line>
```

This code might result in an error that's similar to:

```
The order import process failed for source order CSQ-2043-1~42542342 for the following reason: The item 10029041 cannot be selected more than once.
```

This example fails because it doesn't explicitly identify each node in the hierarchy. It uses the same `ParentLineReference>1` to identify each option class, so Order Management can't accurately explode the hierarchy. For details about exploding, see *Forecast an Assemble-to-Order Item*.

Problems That Happen at runtime

Trouble	Shoot
The Configuration page in the Order Management work area doesn't display the configure options. I set up my item in Product Information Management. I create a sales order in Order Management, search for the item on the catalog line, click **Configure and Add**, but the configuration page displays only the name of the item. It doesn't display any components or configure options.	Make sure you set the Optional attribute to Yes for each class that contains an option, and for each option. For the option, you must set this attribute on the child structure. You can't set it from the parent structure.
The Order Management work area doesn't display the transaction attribute.	The Configurator Models work area doesn't come predefined to include transactional attributes. You must import the model so Order Management can populate them at runtime. Try this. 1. Go to the Configurator Models work area, click **Tasks > Manage Snapshots**, click **Actions > Import Model Item**, then import your model. 2. Take action depending on whether your model is new or revised. - **New**. Release the workspace. - **Revised**. - Update the item class. - Add it and the model to an unreleased workspace. - Release the workspace.
Order Management fails to create the item. I configure the item in Order Management, then click Submit. Order Management displays an error message and sets the sales order status to Draft.	Try this. 1. Make sure all applications and servers are up and running. 2. If the problem persists, there's probably a problem in your setup. Use the error message to investigate the root cause of the error. Start with the Manage Items page in Product Information Management. 3. Resubmit the order in Order Management.

Trouble	Shoot
The assemble-to-order flow attempts to create the configured item but fails.	
An exception occurs after i click Submit. I configure the item in Order Management, then click Submit. Order Management accepts the configuration and sets the sales order status to Processing. The flow creates the configured item but an error occurs in downstream processing.	Learn how to fix this problem. For details, see *Manage Exceptions for Configured Items*.
An assemble-to-order flow doesn't create a supply order.	Different problems might cause this problem. They're similar to problems that occur in back-to-back fulfillment with an item that isn't configured. Here are some fixes you can try. • Run the scheduled process to release planning recommendations. • Examine your available-to-promise and sourcing rules in Global Order Promising. Make sure you aren't missing rules and that they don't conflict with one another. • Supply for a component in the configured item isn't available in the requested time frame. • Examine your work definition. Global Order Promising can't do a Make recommendation if there's a problem in your work definition. • Collect data for Global Order Promising. Do the fix, then release the planning recommendations to create the supply order.
The flow creates a purchase requisition but doesn't create a purchase order.	If a blanket purchase agreement exists with the supplier, and if the flow sends a purchase requisition, then it normally creates the purchase order. Try this. 1. Sign into the purchasing application and examine the state of the requisition. If an error message indicates that pricing is missing, then a setup problem might exist in the item definition. 2. Go to the Product Information Management work area, then click **Tasks > Manage Items**. 3. Search for and open you item, then examine the item structure. Make sure the Specifications area for each component includes a price. 4. In Purchasing, make sure an approver approved the blanket purchase agreement. If not, correct errors and resubmit it.
The flow fails to create a work order. I set up a pick-to-order model that includes pick components and a child assemble-to-order component. I successfully assign a Make At or Buy From sourcing rule for each of my options. I set the lead time for each item in the model to 0 days or empty at the organization level. I notice that if on-hand availability doesn't exist for any pick-to-order or assemble-to-order option, then the flow fails to create a work order.	Note • You must create a sourcing rule, assignment set, and available-to-promise rule for an assemble-to-order model so the flow can do a supply chain search. • You don't need a sourcing rule for components that you keep on hand, but you must set the ship date for the order schedule for them to Infinite. • To schedule the work order on a specific ship date, you must maintain on-hand inventory for the components or set up a sourcing rule. • If you get the components through your own purchase order process or manually, and if Global Order Promising must schedule the order in a specific period that isn't infinite, then you must create a separate sourcing rule for your components and use infinite availability in your available-to-promise rule.

Trouble	Shoot
I wonder whether the flow expects on-hand inventory to exist for all items in the model before it creates the work order.	
Global Order Promising doesn't make a recommendation. Inventory contains the on-hand quantities for the configured item, but Global Order Promising doesn't make an available-to-promise recommendation.	Promising is only aware of on-hand quantities that you collect and refresh. To collect on-hand quantities from your warehouse, run the Refresh and Start the Order Promising Server scheduled process.
I create a sales order that includes a configuration model, click Submit, then encounter an error. `JBO-FND:::FND_CMN_SYS_ERR: FND-2An application error occurred. Your help desk was notified.JBO-FND:::FND_CMN_INVALID_ATTRB_API_S The value of the attribute User Item Type is not valid.`	You set the User Item Type when you create the configuration model in Product Information Management. Its possible you chose a type that isn't enabled. For details, see *Create Your Configuration Model*. Try one of these fixes. • Use a different value in the Template attribute when you create the configuration model. The Template attribute sets the value for the User Item Type attribute. Use a type that's enabled, such as Finished Goods or ATO Option Class. • Enable the lookup code for the User Item Type attribute that you're currently using.

Exception Messages

Here's an example exception message.

```
com.oracle.bpel.client.BPELFault: faultName: {{http://schemas.oracle.com/bpel/extension}remoteFault}
messageType: {{http://schemas.oracle.com/bpel/extension}RuntimeFaultMessage} parts:
{{ summary=<summary>oracle.fabric.common.FabricInvocationException: Unable to invoke endpoint URI
"http://scm-internal.oracleoutsourcing.com:10617/invUom/UnitOfMeasureService" successfully due to:
javax.xml.soap.SOAPException: javax.xml.soap.SOAPException: Bad response: 503 Service Temporarily Unavailable
from url
```

The text `Bad response: 503 Service Temporarily Unavailable` indicates that the web service isn't available. Its possible something is restarting, such as the server, a component or system that the service calls, or the service itself. Wait a few minutes, then click Resubmit.

If the message describes a setup problem, then fix the setup, then Resubmit.

For details, see *Manage Exceptions for Configured Items*.

Getting Help

It might be necessary to contact Oracle Support. If you do, see *SRDC Configure to Order, Data Collection for Configure to Order (Doc ID 2141338.1)* on My Oracle Support to help you prepare.

Related Topics

- Create Your Configuration Model
- Collect Planning Data for Your Configuration Model
- Troubleshoot Order Management
- Overview of Setting Up Configuration Models
- Manage Exceptions for Configured Items

Web Services

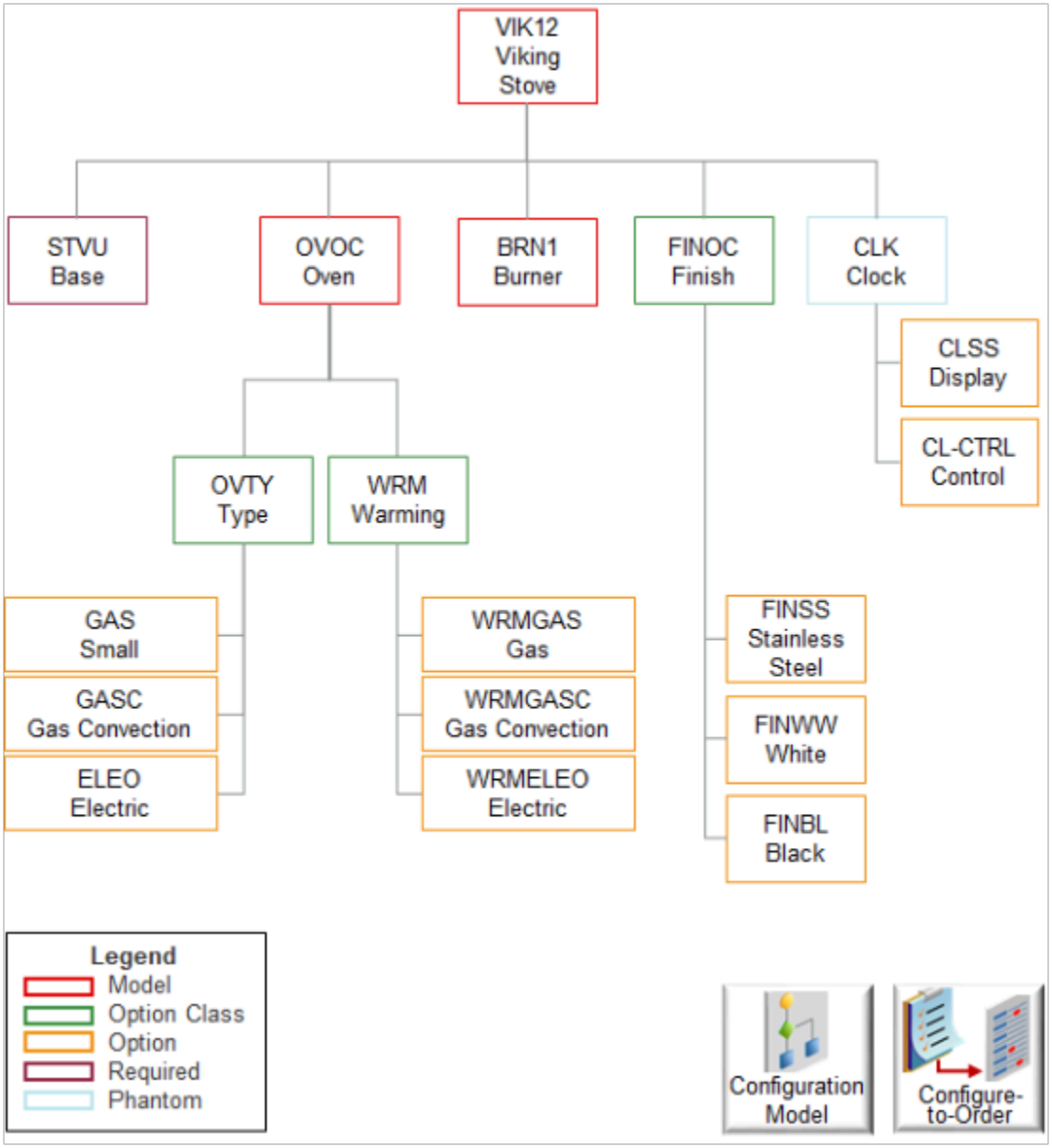
Overview of Using Web Services with Configure-to-Order

Use the ConfiguredItemService web service to create, read, update, or delete a configured item.

Get and view the options that your user selects for the model and for the configured item's structure, including the referenced model, options that the user selected, and transaction attributes that these options reference.

Example

Here's an example of an assemble-to-order model for the VIK12 Stove.



Get an introduction to this model. For details, see *Examples of Using Configure-to-Order*.

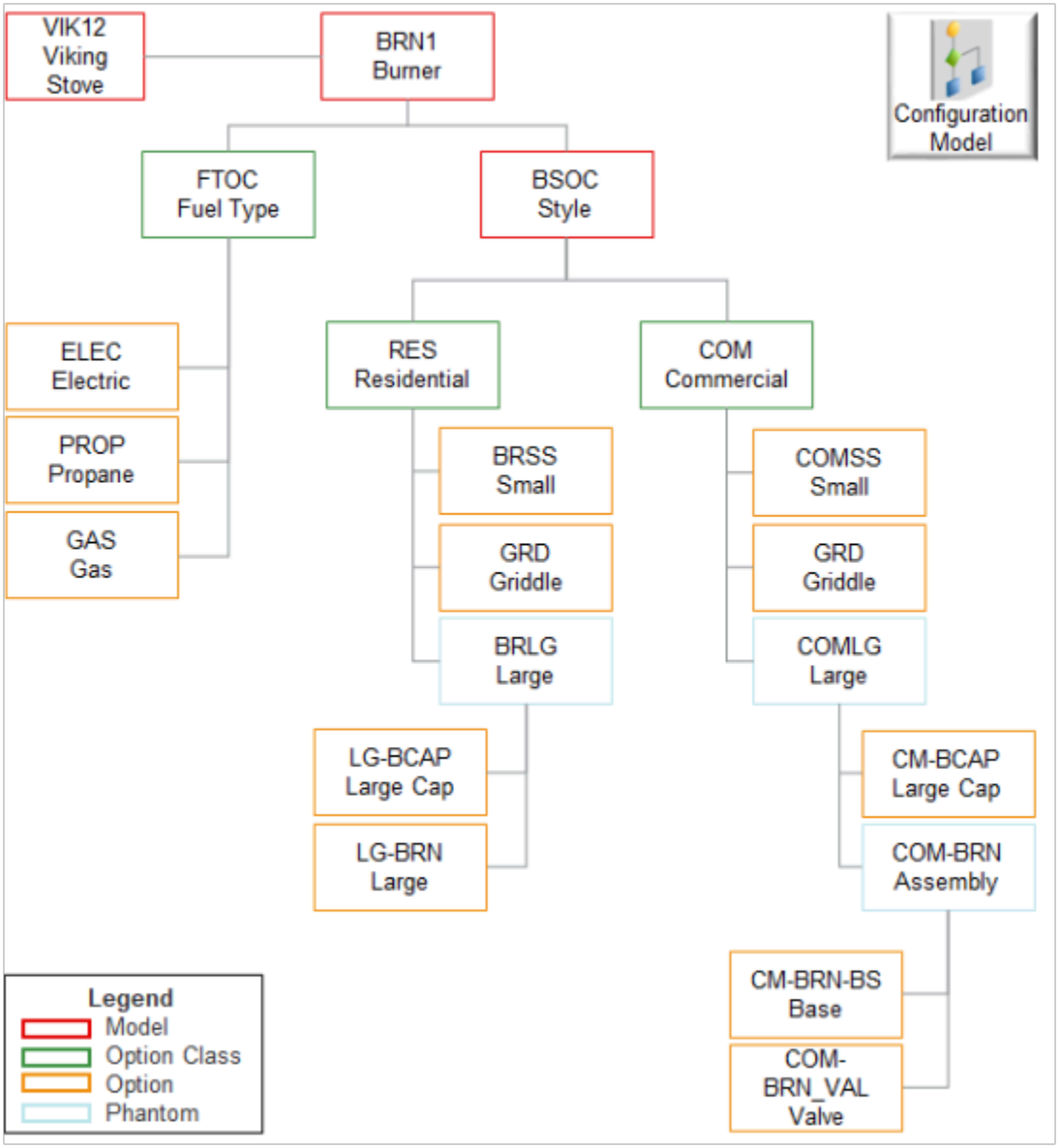
The top parent, VIK12 Stove, includes:

- STVU, which is the stove unit base.
- OVOC, an assemble-to-order model for the oven.
- BRN1, an assemble-to-order model for the burners.
- FINOC, an option class for the finish. It contains options your user can choose to specify the type of finish, such as stainless steel.
- CLK, a phantom for the clock assembly. It includes options your user can choose to specify the display and type of control. A phantom is an item that you physically build but don't stock. In this example, you assemble the clock on demand. You don't build it in anticipation of demand and stock it.

OVOC, the oven, contains option classes.

- Option class OVTY specifies the oven type. It contains options Gas, Gas Convection, and Electric. Gas Convection and Electric aren't configure options.
- Option class WRM specifies options for the warming oven.

Here's the hierarchy for the BRN1 model.

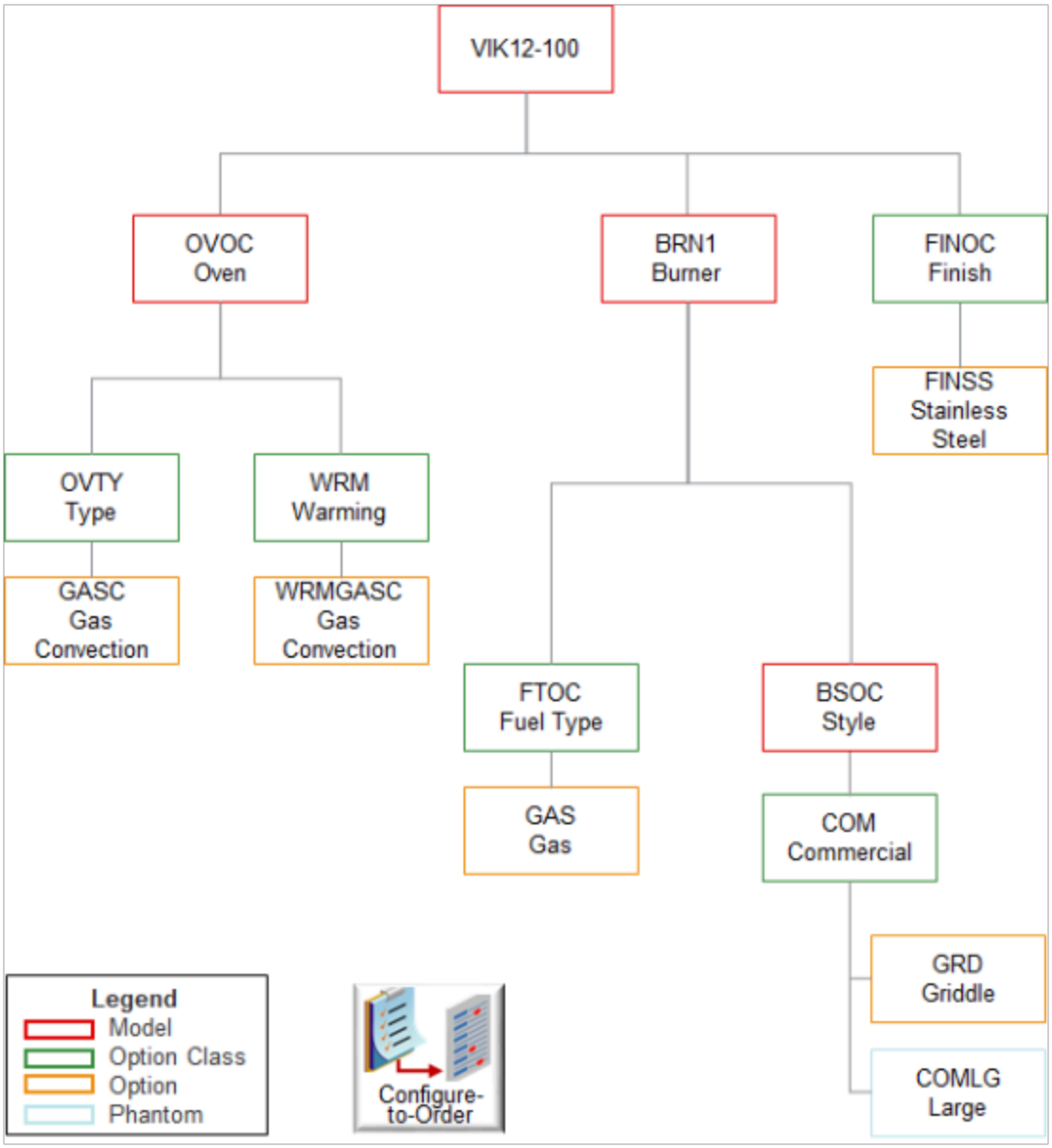


Note

- BRN1 is a complex child model that includes its own assemble-to-order model.
- FTOC is an option class your user uses to choose the fuel type, such as electric, propane, or gas.
- BSOC is an assemble-to-order model that specifies the style as residential or commercial.
- BSOC is a child of BRN1 and a grandchild of VIK12.

- RES and COM each contain options and phantoms.
- RES contains BRSS and GRD, which are items you stock. It also contains BRLG, a phantom you don't stock. You don't stock BRLG and COMLG because your market research indicates customers don't order these options very often, its costly to keep in inventory, so you build it on demand.
- COMLG is a phantom of COM. It contains its own phantom COMLG, and COMLG contains a phantom, COM-BRN. Customers use the CM-BRN-BS to choose the type of base and COM-BRN_VAL to choose the type of value to use when building COM-BRN.

Here's an example configured item, VIK12-100, that your customer ordered. It includes only the options the customer chose.



Note

- STVU is the stove unit base. Its required. The work order includes STVU but VIK12-100 doesn't.

- An oven can use only one type of fuel, burner style, oven type, and warming oven. So each of these components includes only the one option you customer chooses.
- The customer didn't want the clock, didn't choose CLK, so its not in VIK12-100.
- The customer chose COM, the commercial burner, and GRD, the griddle for the commercial burner. The customer also chose COMLG, the large commercial burner.

Operations You Can Use with Web Services for Configured Items

Use operations with the ConfiguredItemService web service to create, read, update, or delete a configured item.

Operation	Value
findCtoSalesStructure	Get the set of options, option classes, and transaction attributes you user selects at runtime.
findCtoItemStructure	Get the entire structure of the configured item that your user selects at runtime. It includes child configured items, options, required components, phantoms, substitutions and transaction attributes.
refreshConfigStructure	Refresh the item structure in the explosion repository.
deleteConfigStrucure	Delete the item structure from the explosion repository.

findCtoSalesStructure and findCtoItemStructure Operations

Get Sales Structure

Here's the payload that the findCtoSalesStructure operation returns for the VIK12-100 configured item. Learn about this item. For details, see *Overview of Using Web Services with Configure-to-Order*.

LINE_ID	SUB_ITEM_TYPE	ITEM_NUMBER	PARENT_ LINE_ID	CONFIG ITEM NUMBER	QTY	UOM
1	ATO	VIK12	1	VIK12-100	1	EA
2	ATO	BRN1	1	BRN1-11	1	EA
3	OPTION CLASS	FTOC	2	FTOC	1	EA
4	RES	GAS	3	GAS	1	EA
5	ATO	BSOC	2	BSOC-2	1	EA
6	OPTION CLASS	COM	5	COM	1	EA

LINE_ID	SUB_ITEM_TYPE	ITEM_NUMBER	PARENT_ LINE_ID	CONFIG ITEM NUMBER	QTY	UOM
7	RES	COMLG	6	COMLG	1	EA
8	RES	GRD	6	GRD	1	EA

The payload includes each of the child assemble-to-order items. Assume BRN-11, BSOC-2, and OVOC-30 are the child configured items.

Get Item Structure

Here's the payload that the findCtoItemStructure operation returns for the VIK12-100 configured item.

LINE_ID	SUB_ITEM_ TYPE	ITEM_ NUMBER	INVENTORY_ ITEM_ID	PARENT_ LINE_ID	CONFIG ITEM_ NUMBER	QTY	UOM	WIP_ SUPPLY_ TYPE	RootPhantomItemId
1	ATO	VIK12	100	1	VIK12-100	1	EA	-	-
2	ATO	BRN1	101	1	BRN1-11	1	EA	-	-
3	OPTION CLASS	FTOC	102	2	FTOC	1	EA	-	-
4	STD	GAS	103	3	GAS	1	EA	-	-
5	ATO	BSOC	200	2	BSOC-2	1	EA	-	-
6	OPTION CLASS	COM	201	5	COM	1	EA	-	-
7	RES	COMLG	202	6	COMLG	1	EA	Phantom	-
8	RES	COM-BRN	203	7	COM-BRN	1	EA	Phantom	202
9	RES	COM-BRN-BS	204	8	COM-BRN-BS	1	EA	-	202
10	RES	COM-BRN-VAL	205	8	COM-BRN-VAL	1	EA	-	202
11	RES	COM-BCAP	206	7	COM-BCAP	1	EA	-	202
12	RES	GRD	207	6	GRD	1	EA	-	-
13	ATO	OVOC	300	1	OVOC-30	1	EA	-	-

LINE_ID	SUB_ITEM_ TYPE	ITEM_ NUMBER	INVENTORY_ ITEM_ID	PARENT_ LINE_ID	CONFIG ITEM_ NUMBER	QTY	UOM	WIP_ SUPPLY_ TYPE	RootPhantomItemId
14	OPTION CLASS	OVTY	301	13	OVTY	1	EA	-	-
15	RES	GASC	302	14	GASC	1	EA	-	-
16	OPTION CLASS	WRM	303	13	WRM	1	EA	-	-
17	RES	WRMGASC	304	16	WRMGASC	1	EA	-	-
18	OPTION CLASS	FINOC	104	1	FINOC	1	EA	-	-
19	RES	FINSS	105	18	FINSS	1	EA	-	-
20	RES	STVU	106	1	STVU	1	EA	-	-
21	RES	CLK	107	1	CLK	1	EA	Phantom	-
22	RES	CLSS	108	21	CLSS	1	EA	-	107
23	RES	CL-CTRL	109	21	CL-CTRL	1	EA	-	107

findCtoItemStructure includes details that findCtoSalesStructure doesn't include.

- The payload includes the required items STVU and CLK.
- PARENT_LINE_ ID identifies the parent of the item. For example, PARENT_LINE_ ID for BRN1 is 1. BRN1 is a child in VIK12. The LINE_ID for VIK12 is 1.
- WIP_SUPPLY_TYPE indicates whether the item is a phantom.
- COMLG and CLK are phantom items, so the detail includes their subassemblies.
- RootPhantomItemId identifies the phantom that contains the item. For example, the RootPhantomItemId for CM-BRN is 202. CM-BRN is an item in phantom COMLG. The INVENTORY_ITEM_ ID for COMLG is 202.

Transaction Attributes

The web service also returns transaction attributes when it returns the item.

LINE_ID	ATTRIBUTE_NAME	CHARACTER_ATTRIBUTE_ VALUE	NUMBER_AT TRIBUTE_ VALUE	TIMESTAMP_ ATTRIBUTE_ VALUE
7	FINISH	Black	-	-

refreshConfigStructure and deleteConfigStrucure Operations

Configure-to-order saves structure details for the configured item in the explosion repository when the flow requests the structure. You use the refreshConfigStructure operation to refresh this data. Refreshing helps to reduce the amount of data that findCtoItemStructure returns and improves performance.

Details for the configured item are already available in the repository, so it isn't necessary to bring data into the repository as part of findCtoItemStructure.

- Use refreshConfigStructure before you use findCtoItemStructure.
- Use the same set of parameters you use with findCtoItemStructure.
- Set the RefreshStructure attribute in the first row of your payload. Set it to.
    - **true**. Refresh data that already exists in the repository before getting it.
    - **false**. Save only the missing structure details in the repository. Don't refresh data that already exists in the repository. The default value is false.

Use deleteConfigStrucure to delete structure details of the configured item from the repository.

Parameters You Use with Operations

Here are the parameters you can use with the operations to identify the item you want to find, refresh or delete. Use them with operations.

- findCtoSalesStructure
- findCtoItemStructure
- refreshConfigStructure
- deleteConfigStrucure

You must use at least one of parameter. If you don't, the web service will display an error.

Attribute	Type	Description
ConfigItemId	Long	Value that identifies the configured item.
ConfigItemNumber	String	Name of the configured item.
BaseModelId	Long	Value that identifies the assemble-to-order model.
BaseModelItemNumber	String	Name of the assemble-to-order model.
CreationDateFrom	Date	Date when the flow started to create the configured item.
CreationDateTo	Date	Date when the flow finished creating the configured item.

For example, to refresh data for the VIK12-100, set ConfigItemNumber to VIK12-100.

Integrate Configured Items

Order Management supports Oracle Configurator Runtime so the user can create and validate a configured item.

Order Management calls the ConfiguratorManager service when a sales order includes a configured item. This service creates and validates the configuration.

Get more details about:

- Configurator models, see *Configurator Models*.
- Configuration Effective Date parameter, see *Manage Order Management Parameters*.
- Use a file to import orders, see *Import Orders Into Order Management*.
- Use a web service to create an integration that automatically imports source orders, see *Web Services That You Can Use to Integrate Order Management*.

Validations That the Configurator Does on Import Payloads

The configurator makes sure:

- Model definition exists in Product Information Management for each item that the import payload contains.
  - Configuration node
  - Each item
  - Primary UOM for each item
- Import payload includes a value for attribute ProductNumber for each order line.
- Import payload includes a quantity for each order line.
- The Product Information Management work area allows a decimal quantity for each decimal quantity that the import payload contains.
- Import payload doesn't include more than one root model instance.
- Import payload doesn't specify an item more than one time, and that Order Management can resolve this item according to a model in Product Information Management.
- If the import payload includes an optional model reference, then the payload explicitly states the full input path to this model reference.

How Web Services Resolve Missing Nodes in Import Payloads

If a configuration hierarchy in the import payload includes two levels, but if the model in Product Information Management that the payload references includes a different hierarchy, then the createConfigForModelLine web service attempts to modify the structure in the payload so it matches the hierarchy in Product Information Management.

For example, assume the import payload includes a hierarchy.

- M1
  - SI1
  - SI2

- SI5

Assume the model in Product Information Management includes a hierarchy.

- M1
  - SI1
  - OC1
  - SI5
    - SI2
    - SI5
    - M2
      - OC2
        - SI4
        - SI5

createConfigForModelLine will modify the structure from the import payload to this structure.

- M1
  - SI1
  - OC1
    - SI2
  - M2
    - OC2
      - SI5

How Web Services Handle Quantities in Import Payloads

If the order import payload specifies only one of these quantities for the configuration line.

- **Unit quantity**. Then createConfigForModelLine calculates the ordered quantity.
- **Ordered quantity**. Then createConfigForModelLine calculates the unit quantity.

If the import payload specifies the unit quantity and the ordered quantity, and if the unit quantity for the configuration line multiplied by the ordered quantity for the immediate parent line.

- **Equals the ordered quantity for the configuration line.** createConfigForModelLine can successfully process the line.
- **Doesn't equal the ordered quantity for the configuration line.** createConfigForModelLine creates an error and order import fails.

If the import payload specifies a value in the Quantity Per Model attribute, then createConfigForModelLine ignores it and calculates the unit quantity and ordered quantity according to the quantities that the import payload specifies for all configuration lines.

Structure Your Configured Item's Payload

Structure your payload when you call the ConfiguredItemService web service.

Structure your payload when you call the ConfiguredItemService web service.

Here's the hierarchy that you use.

- **1 findCriteria**. Structure that contains the entire object and attribute filter criteria.
    - **1.1 fetchStart and fetchSize**. Required. Specify the number of rows in the result set, and specify which row to start with.
        - **1.1.1 fetchStart**. Row to use as the starting point. The default is 0. If you set the value to 0, then the result set begins with the first row of the data set. If you set the value to 99, then the result set begins with the 100th row of the data set.

          > **Note:** Caution: If you use findCtoItemStructure, and if you set fetchStart to any value other than 0, then you must use refreshConfigStructure before you use findCtoItemStructure. The web service assumes the configured item is available in the explosion repository and gets records beginning with the starting point. Using refreshConfigStructure makes sure the starting point contains a value.

        - **1.1.2 fetchSize**. Maximum number of rows to get. If you set fetchSize to -1 (negative one), then the search gets all rows that it finds, starting with the row that fetchStart specifies.

          If fetchSize is greater than the number of remaining rows, then the search returns only the remaining rows. The predefined maximum fetchSize is 500. You can't exceed it in a single query.

If you set fetchSize to a value greater than 500 or to -1, and if more than 500 records match your search criteria, then the web service creates an exception.

- **1.2 filter**. Structure that contains runtime search conditions. The view criteria restricts data that your query returns to the rows that match the filters you specify. If no view criteria exists, then the web service doesn't apply filters when it processes your request.

  Think of view criteria as a collection of one or more query-by-example (`QBE`) rows that define a data filter for a view object. The web service converts view criteria to an SQL WHERE clause.

  - **1.2.1 conjunction**. Specify how to evaluate search conditions in relation to each other. For example, And, Or, Not, AndNot, and OrNot.

    If you don't specify a value, then the web service uses AND. Behavior is similar to how Oracle Application Development Framework works when a conjunction exists between view criteria and the previous view criteria.

    Conjunction is case sensitive.
  - **1.2.2 Structure that contains a set of runtime filter conditions for one or more attributes**. Behavior is similar to how Oracle Application Development Framework handles a list of view criteria rows, where each row defines one view criteria.

    - **1.2.2.1 conjunction**. See level 1.2.1.
    - **1.2.2.2 item**. Structure that contains one runtime search condition for an attribute.

      - **1.2.2.2.1 conjunction**. See level 1.2.1.
      - **1.2.2.2.2 upperCaseCompare**. Boolean value that specifies whether to convert the attribute value that the search returns to uppercase. Conversion happens before filtering.
      - **1.2.2.2.3 attribute**. For details, see the Attributes subtopic, below.
      - **1.2.2.2.4 operator**. For details, see the Operators subtopic, below.
      - **1.2.2.2.5 value**. Filter criteria the web service uses when it searches for an attribute. You can use % as a wildcard, but you must not use % as the first character. For example, you can use `MySearch%`, but you can't use `%MySearch`.
- **1.3 sortOrder**. Sort the result set in ascending order or descending order.
- **1.4 findAttribute**. Attributes to get after the web service applies your filter criteria. If you specify attribute x, then the web service only gets the value for attribute x. If you don't specify a value, then the web service gets all attributes. A child object is an attribute of the parent object.
- **1.5 excludeAttribute**. Boolean value. If True, get all attributes except the ones you specify in findAttribute. If False, get the attributes you specify in findAttribute.

- **2 findControl**. Get all translations of attributes that exist. This parameter affects the entire result set.

1.2.2.2.3 Attributes

Include the case sensitive name of the attribute to filter on. The root node of a child object is an attribute of the parent object.

Here are the attributes you can use. You must use at least one of them. If you don't, the web service will raise an error.

Attribute	Type	Description
ConfigItemId	Long	Value that identifies the configured item.
ConfigItemNumber	String	Name of the configured item.
BaseModelId	Long	Value that identifies the assemble-to-order model.
BaseModelItemNumber	String	Name of the of the assemble-to-order model.
CreationDate	Date	Date when the flow created the configured item.

You can also include one of these attributes, as necessary.

Attribute	Type	Description
OrganizationId	Big Decimal	Value that identifies the organization you associate with the configured item.
OrganizationCode	String	Abbreviation that identifies the organization.
EffectiveAsOfDate	Date	Date that the configured item goes into effect.
RefreshStructure	Boolean	Contains. • **True**. Refresh the structure in the explosion repository before the web service does the search. • **False**. Don't refresh. The default value is false.

1.2.2.2.4 Operators

Note

- Use one of =, >=, <=, >, <,<>, ISBLANK, ISNOTBLANK, CONTAINS, STARTSWITH, AFTER, ONORAFTER, BEFORE, ONORBEFORE, BETWEEN, or NOTBETWEEN.
- Some operators won't work for all attribute types, such as String, Integer, Date, and so on.
- You typically use numeric operators, such as = or < for a String attribute.
- Each text operator is case sensitive. For example, ISBLANK is a text operator and is case sensitive.

Here are the operators you can use to search a required attribute when you use findCtoItemStructure. If you use any other operator, the web service will create an error.

Attribute	Operator
ConfigItemId	=
ConfigItemNumber	=
BaseModelId	=

Attribute	Operator
BaseModelItemNumber	=
CreationDate	BETWEEN

You can use any operator on an attribute that isn't required when you use findCtoSalesStructure or findCtoItemStructure.

Call Web Service and Process Response

Use your preferred development tool to call the web service, then process the response.

Call Web Service

1. Use a URL in a browser, Soap UI, or some other development tool to call the web service.

   ```
   http://hostname:port/ctoUtilitiesPublicService/ConfiguredItemService
   ```

2. Choose findCtoItemStructure, findCtoSalesStructure, refreshConfigStructure or deleteConfigStructure from the list of operations.
3. Enter your payload.
4. Click **Invoke**.

Process the Response

Use different techniques to process the response.

Use SQL

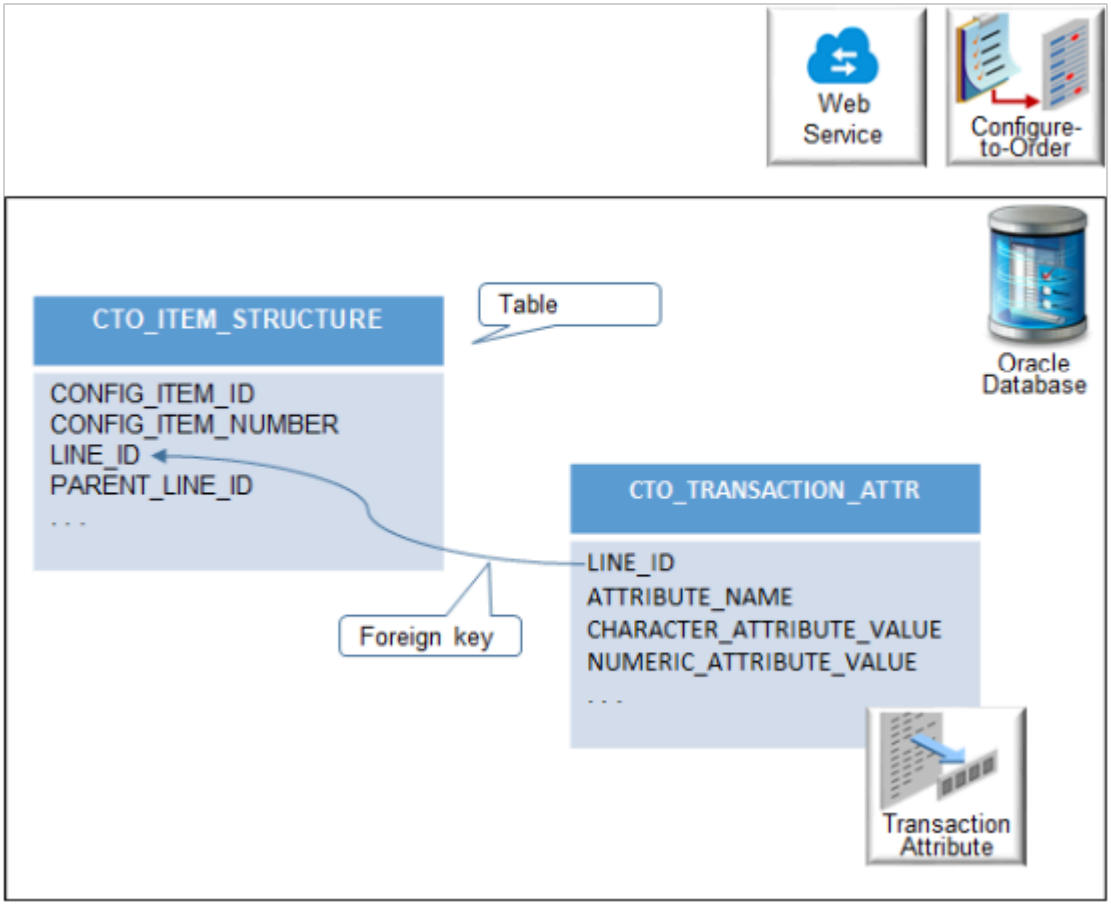
1. Use a proxy to call the web service.

2. Store the response for each row of the response in a relational database table.

   You create two tables.

   - Table CTO_ITEM_STRUCTURE includes a column for each attribute in the response.
   - Table CTO_TRANSACTION_ATTR stores values for transaction attributes. It also includes foreign keys to CTO_ITEM_STRUCTURE according to the LINE_ID attribute.

   For example:



3. Use SQL to query the table to get details about the response.

```
SELECT * FROM CTO_ITEM_STRUCTURE WHERE <CONDITION>
```

Use Java and SQL

1. Use a proxy to call the service.
2. Convert each row of the response to XML.

   Here's an example of the Java code you use to convert.

```
JAXBContext contextObj = JAXBContext.newInstance(new Class[] {
 CtoItemStructure.class
});
Marshaller marshallerObj = contextObj.createMarshaller();
marshallerObj.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
```

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
JAXBElement < CtoItemStructure > rootElement = new JAXBElement < CtoItemStructure > (new
 QName("CtoItemStructureRow"), CtoItemStructure.class, opTempList.get(Index));
marshallerObj.marshal(rootElement, baos);
String xmlContent = new String(baos.toByteArray());
ClobDomain objClobDomain = new ClobDomain(xmlContent);
```

3. Store data in the CTO_ITEM_STRUCTURE table. Use this structure.

Attribute Name	Type	Description
CONFIG_ITEM_ID	Long	Value that identifies the configured item.
ITEM_STRUCTURE	Clob	XML data of the row. This is the output of your Java code.

4. Use SQL to query the table.

   For example:

```
select CONFIG_ITEM_ID,
extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /ConfigItemId ')CONFIG_ITEM_ID,
extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /ConfigItemNumber ')CONFIG_ITEM_NUMBER,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /BaseModelId ')BASE_MODEL_ID,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /BaseModelItemNumber ')BASE_MODEL_ITEM_NUMBER,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /InventoryItemId ') INVENTORY_ITEM_ID,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /InventoryItemNumber ')INVENTORY_ITEM_NUMBER,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /LineId ')LINE_ID,
extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /ParentLineId ')PARENT_LINE_ID,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /SubItemType ') SUB_ITEM_TYPE,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /OptionalComponent') OPTIONAL_COMPONENT,
 extractvalue(ITEM_STRUCTURE,'/ CtoItemStructureRow /ComponentHierarchy ')COMPONENT_HEIRARCHY from
 CTO_ITEM_STRUCTURE where CONFIG_ITEM_ID =<CONFIGURED ITEM ID>
```

Format the Response

The XML output of the web service isn't indented or organized hierarchically according to parent-child relationships. Use an XSLT (Extensible Stylesheet Language Transformations) style sheet to format the web service response so you can visualize the hierarchy.

Here's some XSLT code you can use to format the response. Use this code with your favorite XML tool.

```
< xsl: stylesheet version = "1.0"
 xmlns: xsl = "http://www.w3.org/1999/XSL/Transform"
 xmlns: ns0 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureSer
 vice / types / "
 xmlns: ns2 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureSer vice/
types/"
 xmlns: ns1 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureSer
 vice / "
 xmlns: env = "http://schemas.xmlsoap.org/soap/envelope/"
 xmlns: wsa = "http://www.w3.org/2005/08/addressing"
 xmlns: typ = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureService/
types/ " >
 <
 xsl: output method = "xml"
 indent = "yes"
 version = "1.0"
```

```
encoding = "UTF-8" / >
<
xsl: template match = "*" >
<
xsl: element name = "{local-name()}" >
<
xsl: if test = "not(not(text()) and not(node()))" >
<
xsl: apply - templates select = " node()" / >
<
/ xsl : if > <
/ xsl : element > <
/ xsl : template > <
xsl: template match = "env:Envelope" >
<
xsl: apply - templates select = "env:Body" / >
<
/ xsl : template > <
xsl: template match = "env:Body" >
<
xsl: apply - templates select = " ns0:findCtoItemStructureResponse | ns0:findCtoItemStructureAsyncResponse
" / >
<
/ xsl : template > <
xsl: template match = "ns0 ns0:findCtoItemStructureResponse | ns0:findCtoItemStructureAsyncResponse" >
<
xsl: copy >
<
xsl: apply - templates select = ".//ns2:result[not(ns1:ParentLineId/node())]" / >
<
/ xsl : copy > <
/ xsl : template > <
xsl: template match = "ns2:result" >
<
xsl: variable name = "LineId"
select = "./ns1:LineId" / >
<
xsl: variable name = "OrganizationId"
select = "./ns1:OrganizationId" / >
<
xsl: variable name = "InventoryItemId"
select = "./ns1:InventoryItemId" / >
<
xsl: variable name = "ComponentItemHierarchy"
select = "./ns1:ComponentItemHierarchy" /
>
<
xsl: element name = "ns2:result" >

<
xsl: apply - templates select = "node()" / >
<
xsl: if test = "../ns2:result[(./ns1:ParentLineId = $LineId and ./ns1:OrganizationId =
$OrganizationId and concat($ComponentItemHierarchy, '-', . / ns1: InventoryItemId) =
. / ns1: ComponentItemHierarchy)]
">

<
xsl: apply - templates select = "../ns2:result[(./ns1:ParentLineId = $LineId and
. / ns1: OrganizationId = $OrganizationId)]
" /> <
/ xsl : if > <
/ xsl : element > <
xsl: variable name = "nl"
select = "'
'" / >
```

```
<
xsl: value - of select = "$nl"
disable - output - escaping = "no" / >
<
/ xsl : template > <
/ xsl : stylesheet >
```

Use this XSLT code only to format the output of the findCtoItemStructure operation.

As an alternative, here's some example code you can use to programmatically format the response.

```
String xml = "input.xml ; // input XML file location
String xslt = "transformations.xsl"; // XSLT file location
String output = "formatted.xml"; //output XML file location
 TransformerFactory tf = TransformerFactory.newInstance();
 Transformer tr = tf.newTransformer(new StreamSource(new File(xslt)));
 tr.transform(new StreamSource(new File(xml)),
```

Call the Web Service Through Proxy

Create the proxy for the web service, then use the Java programming language to call the service through your proxy. You send filter criteria as the input.

Get the Item Structure According to a Simple Condition

Here's part of a payload that gets the item structure for configured items you create between 09-Dec-2018 and 09-Dec-2019. You call the service in a loop, where each iteration calls the service with a fetch size of 500. If the number of records you fetch is less than 500, then the code exits the loop.

```
ObjectFactory obj = new ObjectFactory();
ViewCriteriaRow vcr = obj.createViewCriteriaRow();
ViewCriteria vc = obj.createViewCriteria();
ViewCriteriaItem vci = obj.createViewCriteriaItem();
vci.setAttribute("CreationDate");
vci.getValue().add("2018-12-09");
vci.getValue().add("2019-12-09");
vci.setOperator("BETWEEN");
vci.setConjunction(Conjunction.AND);
vcr.getItem().add(vci);
vc.setConjunction(Conjunction.AND);
vc.getGroup().add(vcr);
FindCriteria fc = obj.createFindCriteria();

fc.setFilter(vc);
int fetchCounter = 0;
List < CtoItemStructure > opList = new ArrayList < CtoItemStructure > ;
List
for storing all the output records List < CtoItemStructure > opTempList = null;
List
for storing the result of one service call
while (opTempList == null || opTempList.size() == 500) {
 fc.setFetchStart(fetchCounter);
 fc.setFetchSize(500);
 opTempList = ctoConfigItemService.findCtoItemStructure(fc, null);
 opList.addAll(opTempList);
 items of result list from the service call be will be added to the final list fetchCounter = fetchCounter +
 500;
}
```

Get the Item Structure According to Transaction Attribute

Here's part of a payload that gets the item structure for configured items you create between 09-Dec-2017 and 09-Dec-2019, and where the SCO_color transaction attribute contains the value Black.

```
//Get the list of configured items according to transaction attribute
ObjectFactory obj = new ObjectFactory();
ViewCriteriaRow vcr = obj.createViewCriteriaRow();
ViewCriteria vc = obj.createViewCriteria();
ViewCriteriaItem vci = obj.createViewCriteriaItem();
vci.setAttribute("CreationDate");
vci.getValue().add("2017-12-09");
vci.getValue().add("2019-12-09");
vci.setOperator("BETWEEN");
vci.setConjunction(Conjunction.AND);
vcr.getItem().add(vci);

//Set the filter criteria for the transaction attribute
ViewCriteriaItem childExistsItem = obj.createViewCriteriaItem();
 childExistsItem.setAttribute("CtoTransactionAttr"); childExistsItem.setOperator("EXISTS");
 childExistsItem.setConjunction(Conjunction.AND); vcr.getItem().add(childExistsItem);
 vc.setConjunction(Conjunction.AND);
ViewCriteriaRow childCriteriaRow = obj.createViewCriteriaRow();
ViewCriteria childCriteria = obj.createViewCriteria();
ViewCriteriaItem childCriteriaItem = obj.createViewCriteriaItem();
childCriteriaItem.setAttribute("AttributeName");
childCriteriaItem.getValue().add("SCO_Color");
childCriteriaItem.setOperator("=");
childCriteriaItem.setConjunction(Conjunction.AND);
childCriteriaRow.getItem().add(childCriteriaItem);
childCriteriaItem = obj.createViewCriteriaItem();
childCriteriaItem.setAttribute("CharacterAttributeValue");
childCriteriaItem.getValue().add("Black");
childCriteriaItem.setOperator("=");
childCriteriaItem.setConjunction(Conjunction.AND);
childCriteriaRow.getItem().add(childCriteriaItem);
childCriteria.getGroup().add(childCriteriaRow);
childExistsItem.setNested(childCriteria);
vc.getGroup().add(vcr);

//Make the first call
FindCriteria fc = obj.createFindCriteria(); fc.setFilter(vc);
int counter = 0;
List < CtoItemStructure > configItemList = new ArrayList < CtoItemStructure > (); //List that holds the
 output from the first call
List < CtoItemStructure > opTempList = null;
while (opTempList == null || opTempList.size() == 500) {
 fc.setFetchStart(counter);
 fc.setFetchSize(500);
 opTempList = ctoConfigItemService.findCtoItemStructure(fc, null);
 counter = counter + 500;
 configItemList.addAll(opTempList);

}
//End of making the first call

//End of get the list of configured items according to transaction attribute

//Get the item structure for configured items that the first call returned
ViewCriteriaRow configVcr = obj.createViewCriteriaRow();
ViewCriteria configVc = obj.createViewCriteria();
ViewCriteriaItem configVci = null;
for (CtoItemStructure itemStructRow: configItemList) //For loop to iterate through the list of configured
 items
{
```

```
 if (itemStructRow.getSubItemType().getValue().intValue() == 1) // if it is a config item create condition
 based on config item id
 {
 configVci = obj.createViewCriteriaItem();
 configVci.setAttribute("ConfigItemId");
 configVci.getValue().add(itemStructRow.getConfigItemId().getValue());

 configVci.setOperator("=");
 configVci.setConjunction(Conjunction.OR);
 configVcr.getItem().add(configVci);

 }
}
configVc.getGroup().add(configVcr);
//Make the second call
FindCriteria configFc = obj.createFindCriteria(); configFc.setFilter(configVc);
counter = 0;
List < CtoItemStructure > opList = new ArrayList < CtoItemStructure > (); //List for storing the final
 output opTempList=null;
while (opTempList == null || opTempList.size() == 500) {
 configFc.setFetchStart(counter);
 configFc.setFetchSize(500);
 opTempList = ctoConfigItemService.findCtoItemStructure(configFc, null);
 counter = counter + 500;
 opList.addAll(opTempList);
}
//End of making the second call
//End of get the item structure for configured items that the first call returned
```

Here's an example of the response you receive for the entire structure.

```
< ns0: findCtoItemStructureResponse xmlns: ns0 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/
utilities/configItemStructureService/ty
pes / "> <
 ns2: result
xmlns: ns2 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureService/ty
 pes/"
xmlns: ns1 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureService/"
xmlns: ns0 = "http://xmlns.oracle.com/adf/svc/types/"
xmlns: xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi: type = "ns1:CtoItemStructure" >
 <
 ns1: ConfigItemId > 300100071637243 < /ns1:ConfigItemId> <
 ns1: ConfigItemNumber > DOS - BAT - CTO * 101 * < /ns1:ConfigItemNumber> <
 ns1: BaseModelId > 100100035462338 < /ns1:BaseModelId> <
 ns1: BaseModelItemNumber > DOS - BAT - CTO < /ns1:BaseModelItemNumber> <
 ns1: InventoryItemId > 300100071637243 < /ns1:InventoryItemId> <
 ns1: InventoryItemNumber > DOS - BAT - CTO * 101 * < /ns1:InventoryItemNumber> <
 ns1: ParentInventoryItemId > 300100071637243 < /ns1:ParentInventoryItemId> <
 ns1: ParentInventoryItemNumber > DOS - BAT - CTO * 101 * < /ns1:ParentInventoryItemNumber> <
 ns1: HeaderId > 300100071637281 < /ns1:HeaderId> <
 ns1: LineId > 300100071637284 < /ns1:LineId> <
 ns1: ParentLineId xsi: nil = "true" / >
 <
 ns1: SubItemType > 1 < /ns1:SubItemType> <
 ns1: UnitQuantity xmlns: tns = "http://xmlns.oracle.com/adf/svc/errors/"
unitCode = "Ea" > 1 < /ns1:UnitQuantity> <
 ns1: UnitUOM > Ea < /ns1:UnitUOM> <
 ns1: CreationDate > 2018 - 12 - 02 T10: 59: 08.119 + 05: 30 < /ns1:CreationDate> <
 ns1: RevisionId xsi: nil = "true" / >
 <
 ns1: EffectivityDate > 2018 - 12 - 02 T10: 59: 08.119 + 05: 30 < /ns1:EffectivityDate> <
 ns1: DisableDate xsi: nil = "true" / >
 <
 ns1: OrganizationId > 204 < /ns1:OrganizationId> <
 ns1: OrganizationCode > V1 < /ns1:OrganizationCode> <
```

```
ns1: BillSequenceId > 300100046112964 < /ns1:BillSequenceId> <
ns1: ComponentSequenceId xsi: nil = "true" / >
<
ns1: ComponentType > 1 < /ns1:ComponentType> <
ns1: WIPSupplyType xsi: nil = "true" / >
<
ns1: OptionalComponent > Y < /ns1:OptionalComponent> <
ns1: SubstituteComponent > N < /ns1:SubstituteComponent> <
ns1: ComponentYieldFactor xsi: nil = "true" / >
<
ns1: PlanningFactor xsi: nil = "true" / >
<
ns1: RoundingDirection xsi: nil = "true" / >
<
ns1: BasisType xsi: nil = "true" / >
<
ns1: CheckATP xsi: nil = "true" / >
<
ns1: ComponentHierarchy > 300100046112964 < /ns1:ComponentHierarchy> <
ns1: ComponentItemHierarchy > 300100071637243 < /ns1:ComponentItemHierarchy> <
ns1: ItemClassId > 100000011369001 < /ns1:ItemClassId> <
ns1: RootPhantomItemId > < /ns1:RootPhantomItemId> <
/ns2:result> <
ns2: result >. . .< /ns2:result> <
ns2: result >. . .< /ns2:result>. . .<
/ns0:findCtoItemStructureResponse>
```

Here's an example of the response you receive for the structure according to configure options your user sets in the sales order.

```
< ns2: result xsi: type = "ns1:CtoSalesStructure"
xmlns: ns2 = "http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureService/
types /
 " xmlns:ns1="
http: //xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/configItemStructureService/"
 xmlns:ns0="http://xmlns.oracle.com/adf/svc/types/" xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance">
 <
 ns1: ConfigItemId > 300100090366280 < /ns1:ConfigItemId> <
 ns1: ConfigItemNumber > BSOC * 220 * < /ns1:ConfigItemNumber> <
 ns1: BaseModelId > 300100032201662 < /ns1:BaseModelId> <
 ns1: BaseModelItemNumber > BSOC < /ns1:BaseModelItemNumber> <
 ns1: InventoryItemId > 300100032201442 < /ns1:InventoryItemId> <
 ns1: InventoryItemNumber > STD - Burner < /ns1:InventoryItemNumber> <
 ns1: ParentInventoryItemId > 300100090366280 < /ns1:ParentInventoryItemId> <
 ns1: ParentInventoryItemNumber > BSOC * 220 * < /ns1:ParentInventoryItemNumber> <
 ns1: HeaderId > 300100090366310 < /ns1:HeaderId> <
 ns1: LineId > 300100090366315 < /ns1:LineId> <
 ns1: ParentLineId > 300100090366312 < /ns1:ParentLineId> <
 ns1: SubItemType > 2 < /ns1:SubItemType> <
 ns1: UnitQuantity unitCode = "Ea"
xmlns: tns = "http://xmlns.oracle.com/adf/svc/errors/" > 1 < /ns1:UnitQuantity> <
 ns1: UnitUOM > Ea < /ns1:UnitUOM> <
 ns1: ItemHierarchy > /300100090366280/
300100032201442 < /ns1:ItemHierarchy> <
 ns1: CreationDate > 2018 - 10 - 05 T06: 49: 45.836 - 07: 00 < /ns1:CreationDate> <
 ns1: RootHeaderId > 300100090366310 < /ns1:RootHeaderId> <
 ns1: CtoTransactionAttr >
 <
 ns1: LineId > 300100090366315 < /ns1:LineId> <
 ns1: TimestampAttributeValue xsi: nil = "true" / >
 <
 ns1: NumberAttributeValue > 0 < /ns1:NumberAttributeValue> <
 ns1: DateAttributeValue xsi: nil = "true" / >
 <
 ns1: CharacterAttributeValue xsi: nil = "true" / >
 <
```

```
ns1: AttributeName > Number < /ns1:AttributeName> <
/ns1:CtoTransactionAttr> <
/ns2:result>
```

Examples of Using Web Services with Configure-to-Order

Use these example payloads to get some ideas on how you can use web services with configure-to-order.

Basic Queries

Query on Simple Condition

Use findCtoItemStructure to get the entire structure of configured item VIK12-100.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 -- start at index 0
 <ns2:fetchSize>-1</ns2:fetchSize>
 -- Get all rows
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>ConfigItemNumber</ns2:attribute>
 -- Condition on ConfigItemNumber
 <ns2:operator>=</ns2:operator>
 <ns2:value>VIK12-100</ns2:value>
 --value for Configured Item Number
 </ns2:item>
 </ns2:group>
 <ns2:nested />
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:retrieveAllTranslations />
 </ns1:findControl>
</ns1:findCtoItemStructure>
```

Query on More Than One Condition and Filter Results

Use findCtoItemStructure to get the entire structure for each configured item created between 09-Dec-2018 and 09-Dec-2019 where the configuration model is VIK12. Don't include the InventoryItemId attribute or the ParentInventoryItemId attribute in the response.

```
<ns1: findCtoItemStructurexmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStr uctureService/types/">
<ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>-1</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
```

```
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>between</ns2:operator>
 <ns2:value>2018-10-9</ns2:value> -- Use format YYYY-MM-DD.
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute> BaseModelItemNumber </ns2:attribute>
 <ns2:operator>CONTAINS</ns2:operator>
 <ns2:value>VIK12</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:findAttribute>InventoryItemId</ns2:findAttribute> --List of attributes to include or not include in
 response.
 <ns2:findAttribute>ParentInventoryItemId</ns2:findAttribute>
 <ns2:excludeAttribute>true</ns2:excludeAttribute> --true means don't include in response.
</ns1:findCriteria>
</ns1: findCtoItemStructure>
```

Refresh, Then Query on More Than One Condition

Use findCtoItemStructure to get the entire structure for each configured item created between 09-Dec-2018 and 09-Dec-2019 where the configuration model is VIK12. Use RefreshStructure to refresh the explosion repository.

```
<ns1: findCtoItemStructurexmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
<ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>-1</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>between</ns2:operator>
 <ns2:value>2018-10-09</ns2:value> -- Use format YYYY-MM-DD.
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute> BaseModelItemNumber </ns2:attribute>
 <ns2:operator>CONTAINS</ns2:operator>
 <ns2:value>VIK12</ns2:value>
 </ns2:item>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare></ns2:upperCaseCompare>
 <ns2:attribute> RefreshStructure </ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>true</ns2:value> -- Use true or false with RefreshStructure.
 </ns2:item>
 </ns2:group>
```

```
 </ns2:filter>
</ns1:findCriteria>
</ns1: findCtoItemStructure>
```

Get Large Structures

Get Large Structures

If the web service response for a configured item structure includes more than 500 rows, then you must call the service one time for each set of 500 rows. For example, if the structure contains 1550 rows, then you make four calls.

- First call gets rows 0 through 499
- Second call gets rows 500 through 999
- Third call gets rows 1,000 through 1,499
- Fourth call gets rows 1,500 through 1,550

You must not set the start value to a value greater than 0 in the first call. The start value for the second call is greater than zero, so the web service doesn't refresh the repository. Setting the start value to 0 in the first call makes sure the web service refreshes the entire structure.

In this example, assume your response contains 690 rows.

- The first call sets the start to 0 and size to 500 to get rows 0 through 499.
- The second call sets the start to 500 to get rows 500 to 690.

Here's the first call.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>500</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>BETWEEN</ns2:operator>
 <ns2:value>2018-12-09</ns2:value>
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:retrieveAllTranslations />
 </ns1:findControl>
</ns1:findCtoItemStructure>
```

Here's the second call.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
```

```
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>500</ns2:fetchStart>
 <ns2:fetchSize>500</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>BETWEEN</ns2:operator>
 <ns2:value>2018-12-09</ns2:value>
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:retrieveAllTranslations />
 </ns1:findControl>
</ns1:findCtoItemStructure>
```

Refresh, Then Get Large Structures

Using findCtoItemStructure to get a large structure might take a lot of time because it must first refresh the explosion repository to get structure details, and then query the data. To avoid this problem, make two calls.

- Refresh the repository.
- Get the structure.

Use the same parameter values in each call.

Here's the first call.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:refreshConfigStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:configList xmlns:ns2="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/">
 <ns2:CreationDateFrom>2018-12-09</ns2:CreationDateFrom>
 <ns2:CreationDateTo>2019-12-09</ns2:CreationDateTo>
 </ns1:configList>
</ns1:refreshConfigStructure>
```

Add subsequent calls to get the structure. See the Get Large Structures section, above.

Other Queries

Group Your Conditions

In this example, use findCtoItemStructure to get the entire structure of each configured item created between 09-Dec-2018 and 09-Dec-2019 for configuration model VIK12. Use the OR conjunction to group your conditions.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>500</ns2:fetchSize>
```

```
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>BETWEEN</ns2:operator>
 <ns2:value>2018-12-09</ns2:value>
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 </ns2:group>
 <ns2:group>
 <ns2:conjunction>Or</ns2:conjunction>
 --Group with the OR condition.
 <ns2:upperCaseCompare />
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>BaseModelItemNumber</ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>VIK12</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
 <ns1:findControl xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:retrieveAllTranslations />
 </ns1:findControl>
</ns1:findCtoItemStructure>
```

Get Transaction Attributes

Use findCtoItemStructure to get the entire structure of the VIK12-100 configured item. Filter the result so it only includes details for transactional attribute names that start with MFG_Transactional_Attributes, and that contain the value Black.

```
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/"
 >
<ns1:findCtoItemStructure>
<ns1:findCriteria>
<ns2:fetchStart>0</ns2:fetchStart>
<ns2:fetchSize>-1</ns2:fetchSize>
<ns2:filter>
<ns2:conjunction>And</ns2:conjunction>
<ns2:group>
<ns2:conjunction>And</ns2:conjunction>
<ns2:upperCaseCompare>false</ns2:upperCaseCompare>
<ns2:item>
<ns2:conjunction>And</ns2:conjunction>
<ns2:upperCaseCompare>false</ns2:upperCaseCompare>
<ns2:attribute>ConfigItemNumber</ns2:attribute>
<ns2:operator>=</ns2:operator>
<ns2:value>VIK12*100</ns2:value>
</ns2:item>
</ns2:group>
</ns2:filter>
<ns2:childFindCriteria> -- This is the structure for adding conditions based on TIA
<ns2:fetchStart>0</ns2:fetchStart>
<ns2:fetchSize>-1</ns2:fetchSize>
<ns2:filter>
<ns2:conjunction>And</ns2:conjunction>
```

```
<ns2:group>
<ns2:conjunction>And</ns2:conjunction>
<ns2:upperCaseCompare>false</ns2:upperCaseCompare>
<ns2:item>
<ns2:conjunction>And</ns2:conjunction>
<ns2:upperCaseCompare>false</ns2:upperCaseCompare>
<ns2:attribute>AttributeName</ns2:attribute>
<ns2:operator>STARTSWITH</ns2:operator>
<ns2:value>MFG_Transactional_Attributes</ns2:value>
</ns2:item>
<ns2:item>
<ns2:conjunction>And</ns2:conjunction>
<ns2:upperCaseCompare>false</ns2:upperCaseCompare>
<ns2:attribute>CharacterAttributeValue</ns2:attribute>
<ns2:operator>=</ns2:operator>
<ns2:value>Black</ns2:value>
</ns2:item>
</ns2:group>
</ns2:filter>
<ns2:childFindCriteria/>
<ns2:childAttrName>CtoTransactionAttr</ns2:childAttrName> -- This value is fixed and needs to be passed if
 we want filter based on the TIA
</ns2:childFindCriteria>
</ns1:findCriteria>
<ns1:findControl>
<ns2:retrieveAllTranslations>false</ns2:retrieveAllTranslations>
</ns1:findControl>
</ns1:findCtoItemStructure>
```

Get Entire Item Structure According to Transaction Attribute

Get the entire item structure for configured items created between 09-DEC-2018 and 09-DEC-2019, and where the value of transaction attribute SCO_color is Black.

You call the service two times.

- Get configured items that match the filter.
- Use values that the first call returns as the parameters, such as ConfigItemNumber.

Here's the first call.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 <ns2:fetchSize>500</ns2:fetchSize>
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:attribute>CreationDate</ns2:attribute>
 <ns2:operator>BETWEEN</ns2:operator>
 <ns2:value>2018-12-09</ns2:value>
 <ns2:value>2019-12-09</ns2:value>
 </ns2:item>
 <ns2:item>
 --This block contains logic for filtering configured items according to TIA
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
```

```
 <ns2:attribute>CtoTransactionAttr</ns2:attribute>
 <ns2:operator>Exists</ns2:operator>
 <ns2:nested>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:attribute>AttributeName</ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>SCO_Color</ns2:value>
 </ns2:item>
 <ns2:item>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare>false</ns2:upperCaseCompare>
 <ns2:attribute>CharacterAttributeValue</ns2:attribute>
 <ns2:operator>=</ns2:operator>
 <ns2:value>Black</ns2:value>
 </ns2:item>
 </ns2:group>
 </ns2:nested>
 </ns2:item>
 </ns2:group>
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
</ns1:findCtoItemStructure>
```

Here's the second call.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:findCtoItemStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
 <ns2:fetchStart>0</ns2:fetchStart>
 -- start at index 0
 <ns2:fetchSize>-1</ns2:fetchSize>
 -- Indicates all the rows should be returned
 <ns2:filter>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:group>
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:item>
 -- This block has to be repeated for all the configured items returned from the previous call
 <ns2:conjunction>And</ns2:conjunction>
 <ns2:upperCaseCompare />
 <ns2:attribute>ConfigItemNumber</ns2:attribute>
 -- Condition is Based on ConfigItemNumber
 <ns2:operator>=</ns2:operator>
 <ns2:value>VIK12*100</ns2:value>
 -- Configured Item Number returned from the previous call
 </ns2:item>
 </ns2:group>
 <ns2:nested />
 </ns2:filter>
 <ns2:excludeAttribute />
 </ns1:findCriteria>
</ns1:findCtoItemStructure>
```

Remove the Item Structure

Remove the structure of a configured item from the repository.

Here's a payload that removes the structure of all configured items that include VIK12 as the item number.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:deleteConfigStructure xmlns:ns1="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/types/">
 <ns1:configList xmlns:ns2="http://xmlns.oracle.com/apps/scm/cto/matchRepository/utilities/
configItemStructureService/">
 <ns2:BaseModelItemNumber>VIK12</ns2:BaseModelItemNumber>
 </ns1:configList>
</ns1:deleteConfigStructure>
```

Attributes in the Response Payload

Get details about attributes that the ConfiguredItemService web service returns in the response payload.

Structure of the Sales Order

Here are the attributes that the response payload includes when you get the structure of the sales order.

Attribute	Type	Description
ConfigItemId	Long	Value that identifies the configuration item that the flow creates.
ConfigItemNumber	String	Number that identifies the configured item.
BaseModelId	Long	Value that identifies the inventory item for the assemble-to-order model that your user uses to choose the configure options.
BaseModelItemNumber	String	Number that identifies the name of the assemble-to-order model.
InventoryItemId	Long	Value that identifies the inventory item for the line item.
InventoryItemNumber	String	Number that identifies Item name of the line item.
ParentInventoryItemId	Long	Value that identifies the parent of the inventory item.
ParentInventoryItemNumber	String	Number that identifies the name of the parent of the inventory item.
HeaderId	Long	Value that identifies the match header record.
LineId	Long	Value that uniquely identifies the line assigned to the match detail record.

Attribute	Type	Description
ParentLineId	Long	Value that identifies the line that represents the immediate parent item of the component.
SubItemType	Long	Value that identifies the type of sub item. The attribute value is 1, 2, or 4. 1. Assemble-to-order item. 2. Option class. 4. Configure option.
UnitQuantity	BigDecimal	Quantity needed to make one unit of the top level assembly.
UnitUOM	String	Unit of measure for the quantity.
ItemHierarchy	String	Hierarchy of inventory item identifiers, separated by a forward slash (/).
CreationDate	Timestamp	Date when the user created the configured item.
RootHeaderId	Long	Value that identifies the header of the top level assemble-to-order item.

Structure of the Entire Instance

Here are the attributes that the response includes when you get the structure of the entire instance. The response also includes the same attributes for the structure of the sales order except for ItemHierarchy and RootHeaderId.

Attribute	Type	Description
RevisionId	Long	Value that identifies the item revision.
EffectivityDate	Timestamp	Date when the item or component goes into effect.
DisableDate	Timestamp	Date when the flow disables the item or component.
OrganizationId	Long	Value that identifies the organization of the inventory item.
OrganizationCode	String	Abbreviation that identifies the organization of the inventory item.

Attribute	Type	Description
BillSequenceId	Long	Value that identifies the bill sequence for the assemble-to-order model. The structure of the configuration model defines the sequence.
ComponentSequenceId	Long	Value that identifies the sequence for the selected component. The structure of the configuration model defines the sequence.
ComponentType	String	The type of component. Contains 1 or -1 (negative value of 1). 1. Assembled component. -1. Component that isn't assembled. Its typically part of an assembled component.
WIPSupplyType	BigDecimal	Type of supply that's work-in-process. The structure of the configuration model defines the type.
OptionalComponent	String	Contains Y or N. Y. The component is optional. N. The component is required.
SubstituteComponent	String	Contains Y or N. Y. The component is a substitute. N. The component isn't a substitute.
ComponentYieldFactor	BigDecimal	Multiplication factor the flow uses to determine the quantity needed to fulfill the component. The structure of the configuration model defines the factor.
PlanningFactor	BigDecimal	Multiplication factor the flow uses to determine the quantity needed to plan the component. The structure of the configuration model defines the factor.
RoundingDirection	BigDecimal	Specifies the rounding direction as up or down. The structure of the configuration model defines the direction.
BasisType	BigDecimal	Specifies whether the value is an item. The value 1 specifies its an item.
CheckATP	BigDecimal	Specify whether to run available-to-promise rule. Contains 1 or 2.

Attribute	Type	Description
		1. Run rule. 2. Don't run rule. The structure of the configuration model determines whether to run the rule.
ComponentHierarchy	String	Hierarchy of the components in the configuration model. It's a hierarchical list of identifiers, where a dash (-) separates each identifier.
ComponentItemHierarchy	String	Hierarchy of the inventory items in the configuration model. It's a hierarchical list of identifiers, where a dash (-) separates each identifier.
ItemClassId	Long	Value that identifies the item class of the configured item.
RootPhantomItemId	Long	Value that uniquely identifies the phantom of a subassembly item.

Transaction Attribute

Here are the attributes that the response contains when your payload includes a transaction attribute.

Attribute	Type	Description
LineId	Long	Value that uniquely identifies the order line.
AttributeName	String	Name of the transaction attribute.
CharacterAttributeValue	String	Contains the character value of the transaction attribute.
NumberAttributeValue	Long	Contains the number value of the transaction attribute.
DateAttributeValue	Date	Contains the date value of the transaction attribute.
TimestampAttributeValue	Timestamp	Contains the time stamp value of the transaction attribute.

Other Setups

Work Definitions

Guidelines for Creating Work Definitions for Configured Items

Assemble-to-order in a make flow creates the work order for the configured item dynamically at runtime according to the primary work definition that you set up at design time.

Here's what Oracle Manufacturing Cloud does:

1. Receives a request to create a work order for a configured item:
2. Uses the work definition that you set up to create a work order.

Note

- Manufacturing uses the assemble-to-order (ATO) model as the ordered item when it creates the customer order. The item includes configure options that your user sets at runtime.
- You must set up a work definition for the assemble-to-order model.
- You must set the structure name of the model to Primary in Product Information Management. If the name isn't Primary, then you can't create a work definition for the model.
- Manufacturing currently supports work definitions for an assemble-to-order model only for discrete manufacturing.

Create Work Definition

- Use the right side of the page in the work definition to view each level of the model.
- You can expand each option class and view the options in the class.
- Set Production Priority to 1.

Specify Operations

- You can assign the component of a model and a manufacturing resource to each operation.
- You can assign an option class, which will assign all of its components.
- You can apply an item to an operation only if the item is a component in the model.

To specify that the operation depends on the configure option, open your work definition for editing. On the Edit Work Definition Details page, click your **operation**, click **Actions > Edit**, then, in the Edit Operation dialog, set the Option Dependent attribute.

Value	Description
Contains a check mark.	The work order for the configured item includes the operation if: • Your user chooses the configure option that the flow assigns to the operation. or • The flow meets the criteria you set up in the applicability rule for the operation.

Value	Description
Doesn't contain a check mark.	The operation is required. The work order for the configured item always includes a required operation.

Specify Operations in the Hierarchy

You can assign each component to an operation from any level of the item structure.

- You can expand an option class to view the configure options that the class contains.
- You can expand a phantom to view the configure options that phantom contains.
- If you assign the parent, then you can't assign its children.
- If you already assigned the children, you can assign the parent, but the work definition will delete the child assignments.
- You can choose to assign the option class or the options.
- For a phantom that isn't an assemble-to-order item, you can choose to assign the parent phantom or the components.
- If you assign an option class to an operation that depends on an option, and if the user selects at least one option in the class, then the work order for the configured item includes the operation that you specify in the work definition.
- You can't expand a child assemble-to-order model.
- You must assign the entire quantity for the component to the operation.
    - You can't update or split the quantity to more than one operation.
    - You can't inverse the quantity, unit of measure, or yield.
    - You can't assign an item that isn't part of the model structure.

Specify Child Models

If the supply type of a child model:

- **Is phantom**. You don't need to create a separate work definition for the child model. The work order for the parent configured item includes the components of the configured item of the child model.
- **Isn't phantom**. You must create a separate work definition for the child model. Use Planning to create supply for the configured item of the child.

Set Up Phantoms

A phantom is an item that you physically build but don't stock. If you specify an item as a phantom, and if your user chooses the item, then manufacturing stores it as part of the structure.

- If you set the supply type for the component to Phantom in Product Information Management, then you can expand the phantom and view its components in the work definition.
- You can keep expanding until you encounter a component that isn't a phantom.
- You can't expand a child model even if its supply type is Phantom.
- Manufacturing replaces each phantom with its physical counterpart at the next level of the hierarchy when configure-to-order creates the work order.

For details, see *Overview of Phantom Explosion in Work Orders*.

Apply Operation to Work Order

Use an applicability rule to assign an option-dependent operation.

- Specify the criteria that determines whether to include the operation in the work order.
- Create a rule for a configure option, option class, transaction attribute, or a combination of them.
- Add a check mark to the Option Dependent option in the Option Dependent Details area. Enabling Option Dependent turns on the Add icon. Click Add to open the Add Applicability Rule dialog.
- The Add Applicability Rule dialog displays the item structure for your model, but it only displays the configure options.
- The dialog displays configure options and transaction attributes according to the date you select in the Date list on the Edit Work Definition Details page. It doesn't options and attributes that aren't currently in effect.
- Expand each option class to view the configure options that the class contains.
- You can't expand a child model.

Transaction Attribute

If you use a transaction attribute in an applicability rule, then:

- Create the transaction attribute in the Product Information Management work area.
- Assign the application scope to Configuration Matching when you create the transaction attribute. Configuration matching uses the value of the transaction attribute to determine whether the configuration already exists. Manufacturing also uses the transaction attribute to identify the operations it needs to build the configured item.
- Use the item class to apply the transaction attribute to the configuration model, configure option, or child model. You associate a transaction attribute with an item class. Each item in the item class inherits the transaction attribute.
- Assign either the configure option or the applicability rule to your option-dependent operation. Don't assign both. If you assign both, then the work definition doesn't evaluate the rule.
- If your transaction attribute uses numeric data or string data, then make sure the validation type for the value set is independent or subset.
- Expand the configure option in your rule to display the transaction attribute. Expand the transaction attribute to display the attribute value.
- Use the internal name. The Add Applicability Rule dialog displays the display name of the transaction attribute. You can translate the display name. If the value set that the transaction attribute uses is translatable, then the item structure displays the internal name of the value and the translated value in parenthesis. For example, `internal name(translated value)`. The work definition uses the value of the internal name to evaluate the rule. It doesn't use the translated value.

Operators

Use operators in the Rule Text window.

- Include a combination of items and transaction attributes.
- Use AND and OR conditions to create a complex rule.

Validate Your Rule

Click **Validate** to validate your rule syntax.

Use valid syntax.

- Use single quotation marks to enclose each item number.
- Use dot notation to separate each level of the hierarchy.
- Use upper case letters for each operator.

Keywords

Syntax	Example Value
Use the ITEM keyword to specify whether the item a model or option class.	ITEM='CTO_474000'.'CTO_474300'.'CTO_474301' where • CTO_474000 is the name of the model • CTO_474300 is the name of the option class • CTO_474301 is the name of the configure option
Use the TRANSACTIONALATTRIBUTE keyword to specify a transaction attribute. Use double quotation marks to enclose each transaction attribute name and value.	'CTO_474000'.'CTO_474300'.'CTO_474301'. TRANSACTIONALATTRIBUTE ["Finish"] is equal to "Matte" where • Finish is the name of the transaction attribute. • Matte is the value of the transaction attribute.

Operators

Numeric Attributes	String Attributes
• Equal to • Not equal to • Less than • Less than or equal to • Greater Than • Greater than or equal to	• Equal to • Not equal to • STARTSWITH • ENDSWITH • CONTAINS • DOESNOTCONTAIN

Verify Assignments

Make sure you assign each component in your work definition. If you don't, the flow will fail to create the work order for the configured item.

Go to the Edit Work Definition Details page, then click **Actions > Export Operation Item Assignments**. The export saves the entire model structure to Microsoft Excel.

Examine the status in Excel. If the status of your configuration model is.

- **Complete**. You assigned all the required operations.
- **Incomplete**. Identify which part of the hierarchy you must assign, then assign it.

Assignment Status	Description
Complete	You assigned an operation to the parent and each child.
Incomplete	You didn't assign the parent and each child.
Explicit	You assigned a child configure option.
Implicit	You assigned the parent and the child inherited the parent assignment.

Modify Attributes

The work definition gets values for attributes from the model that you create in the Product Information Management work area.

- Basis
- Quantity
- Inverse Quantity
- UOM
- Item Yield
- Planning Percent
- Optional
- Supply Type

You can't modify any of these attributes in the work definition except for Supply Type.

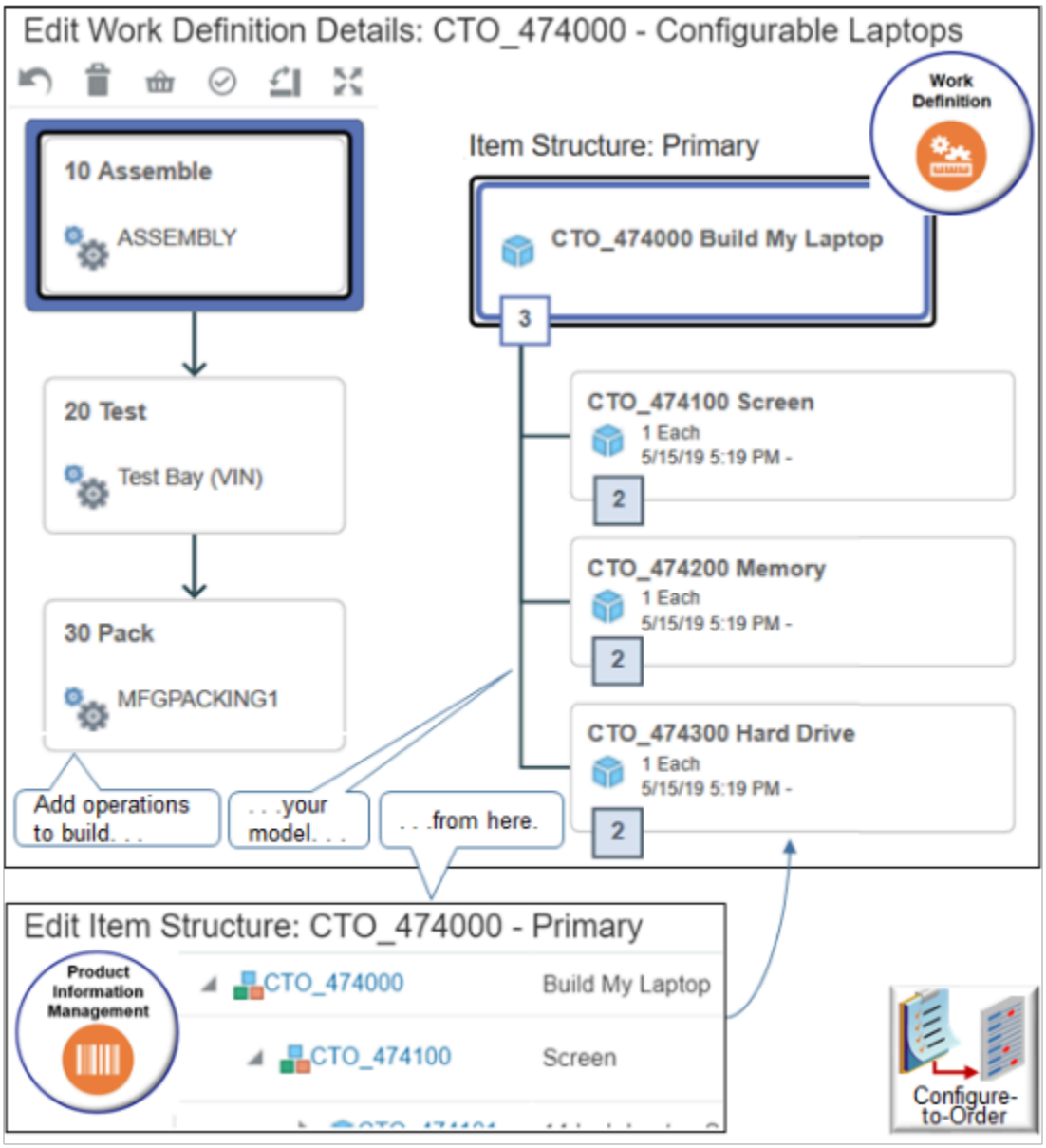
Related Topics

- Overview of Phantom Explosion in Work Orders
- How You Create an Assemble to Order (ATO) Model Work Definition

Create Work Definitions for Configured Items

Use the Work Definition work area to specify the operations needed to build the configuration model you set up in the Product Information Management work area, and the resources needed to do the operations.

For example, specify assemble, test, and pack operations for the CTO_474000 model.



For details, see *Overview of Work Definitions*.

This example assumes you already set up the work centers and you create your own supply. You don't get supply from a supplier.

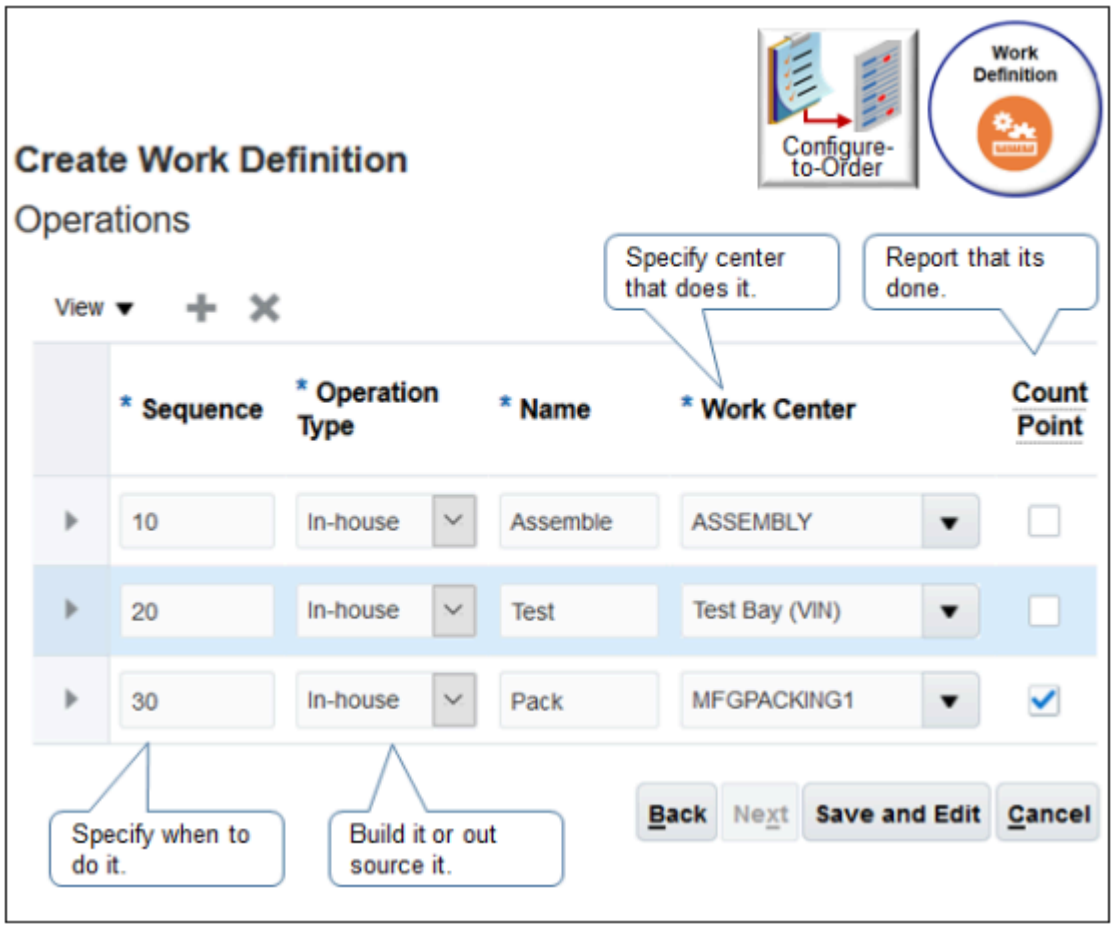
Try it.

1. Go to the Work Definition work area.

2. Create a name for your work definition.
    - On the Overview page, click **Tasks > Manage Work Definition Names**.
    - On the Manage Work Definition Names page, click **Add Row**, set the values, then click **Save and Close**.

Attribute	Value
Display Name	Configurable Laptops
Internal Name	Configurable Laptops
Description	Work definition for configure-to-order laptop computers.
Type	Standard
Used in Planning	Enabled

3. Create the work definition.



- Click **Tasks > Manage Work Definitions**.
- On the Manage Work Definitions page, click **Actions > Add**.
- In the Create Work Definition dialog, set the values, then click **Next**.

Attribute	Value
Item	CTO_474000
Name	Configurable Laptops
Production Priority	1 Priority 1 specifies to use the primary work definition.

- Click **Add Row** and set the values.

Attribute	Value
Sequence	10
Operation Type	In-house
Name	Assemble
Work Center	Assembly

- Click **Add Row** and set the values.

Attribute	Value
Sequence	20
Operation Type	In-house
Name	Test
Work Center	Test Bay

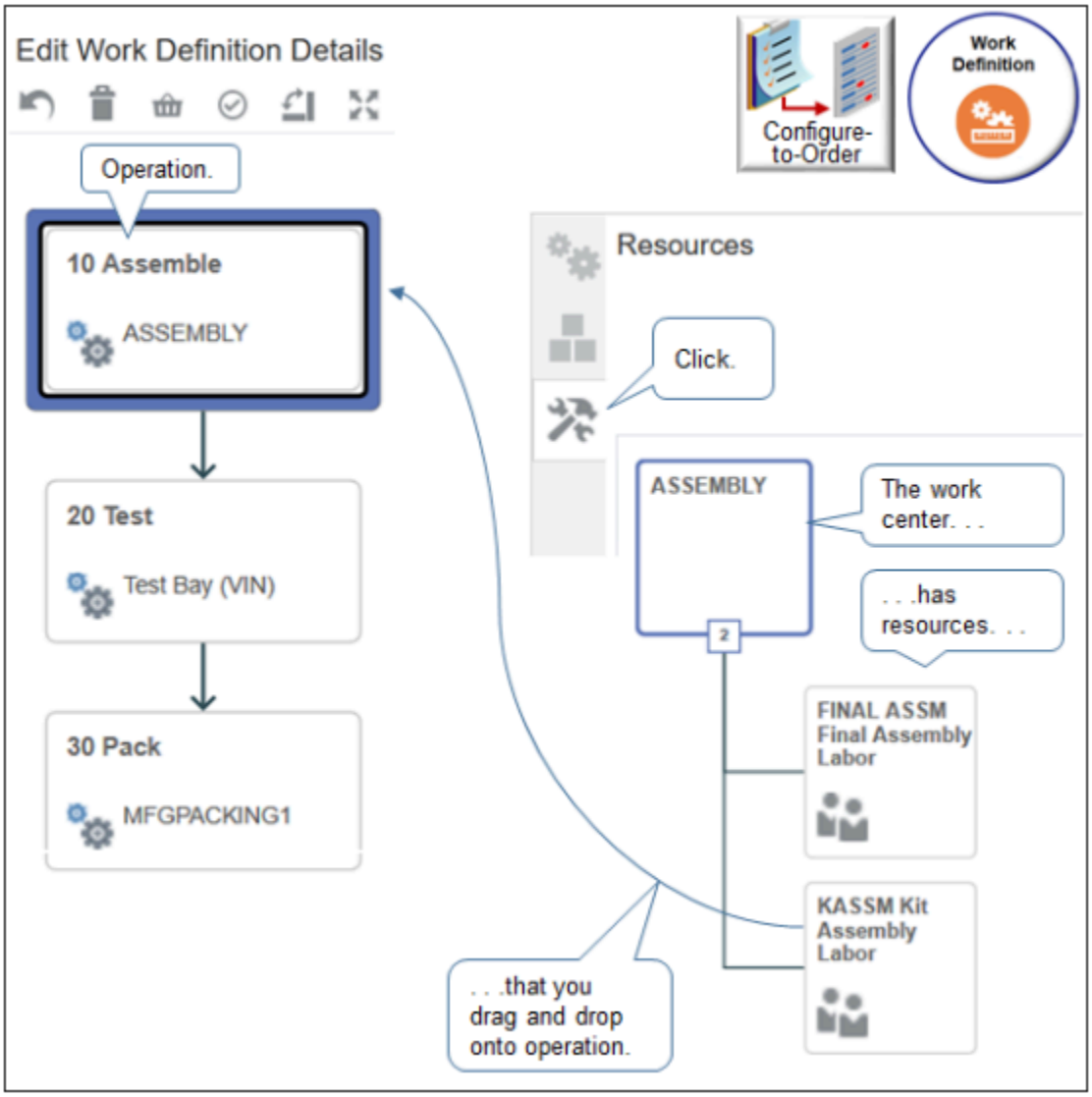
- Click **Add Row**, set the values, then click **Save and Edit**.

Attribute	Value
Sequence	30
Operation Type	In-house
Name	Pack
Work Center	MFGPACKING1
Count Point	Enabled Use this attribute to specify that the last step in the sequence is done.

Attribute	Value
	You must specify Count Point on the last step. You can also specify it on other steps to indicate activities that must finish for them.

Add Resources to Operations

You aren't required to add resources to operations, but you can if you need to specifically control who does the operation. You use resources to specify how.



Try it.

1. On the Edit Work Definition Details page, in the area that displays the item structure, click **Resources**.
2. Add resources to your first operation.
    - Click the Assemble operation.

- Click the Kit Assembly Labor resource, drag it, then drop it onto the Assemble operation.

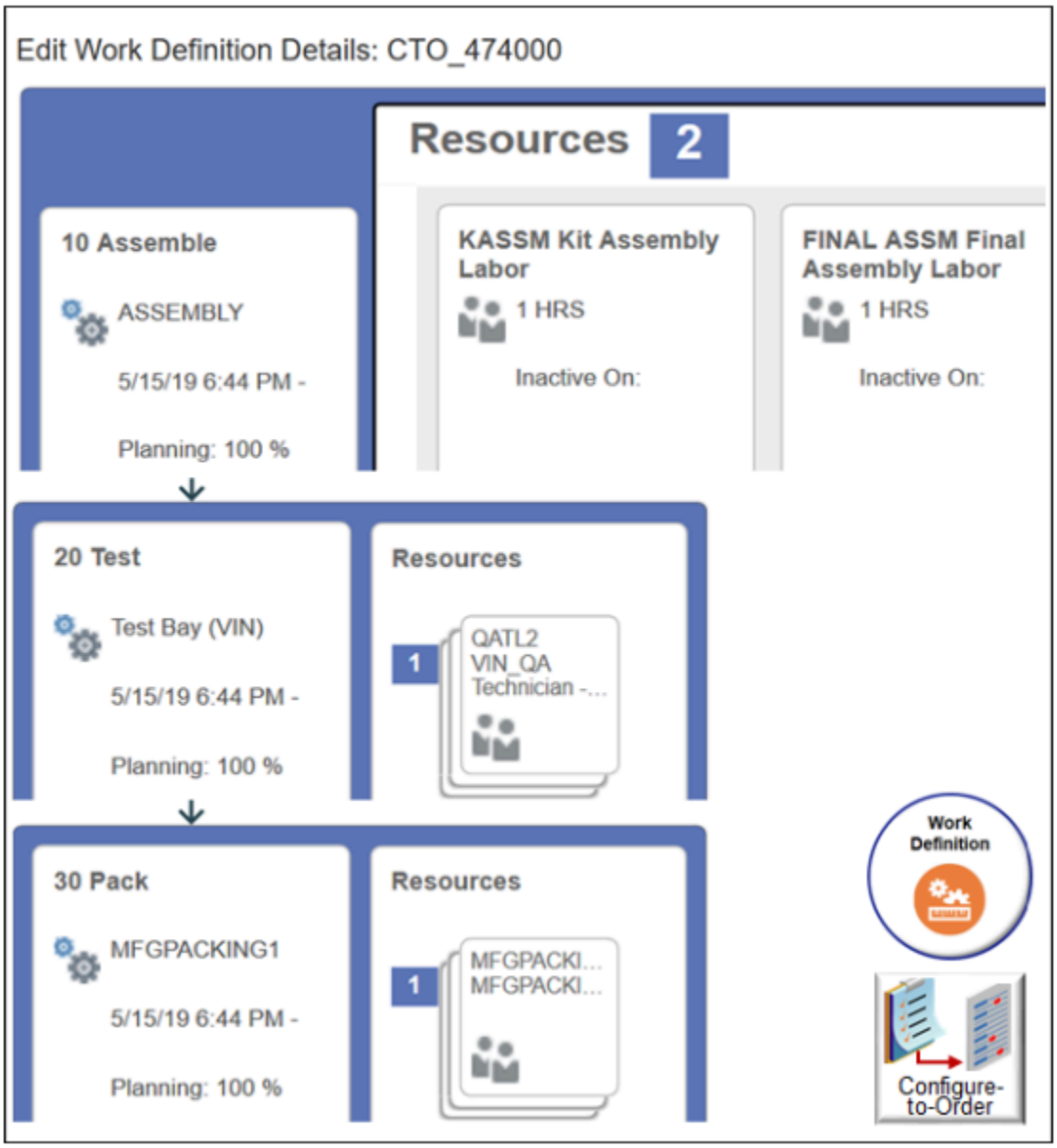
  Each work center has resources that do the operation. Your work center might use resources that are different than this example uses. Learn how to create a work center and it's resources. For details, see *How You Manage Work Centers*.
- In the Assign Operation Resource dialog, set the values, then click **OK**.

Attribute	Value
Units Assigned	1
Basis	Variable This attribute determines whether you need more resources as the quantity increases when you assemble the item. You typically need more resources to assemble a laptop as quantity increases, so set it to Variable.
Usage	1
Inverse Usage	1
Scheduled	No
Principal	Not checked Specify priority when you use the same sequence number for more than one resource. In general, don't use the same sequence number for more than one resource. Instead, sequence them consecutively so it isn't necessary to specify a principal.
Charge Type	Manual
Activity	Setup

  For details, see *How You Edit Work Definitions*.
- Click the Final Assembly Labor resource, drag it and drop it onto the Assemble operation, set the values in the dialog, then click **OK**.

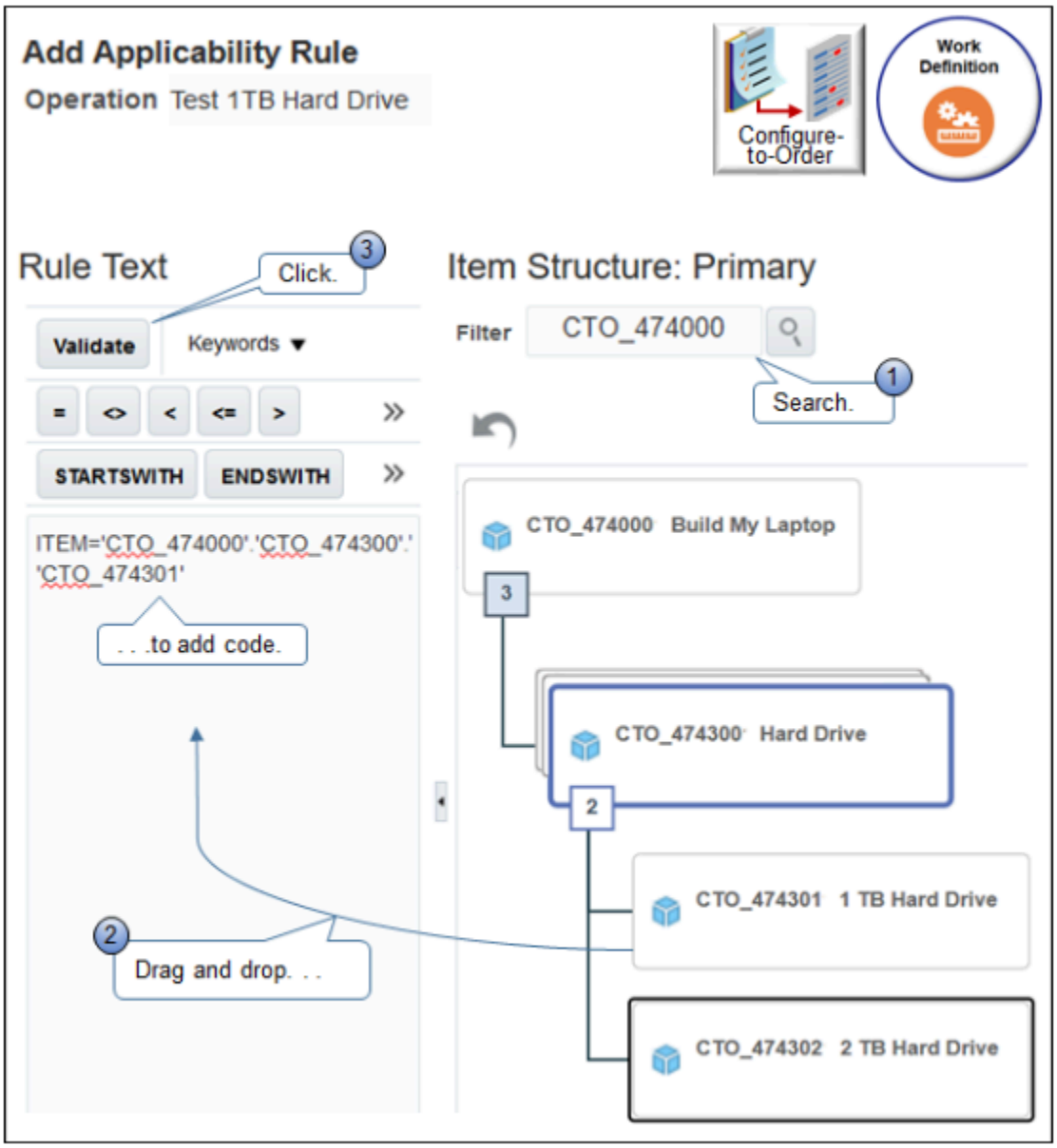
3. Repeat step 2 for the Test operation.
4. Repeat step 2 for the Pack operation.

Your set up should like something like this.



Apply Operation to Work Order

As an option, create a rule that determines when to do an operation in the work order that creates the configured item.



Assume you must do a performance test that depends on a configure option your user sets in Order Management. You can create two different operations.

- Do the 1 Terabyte Hard Drive Test operation only if the user chooses the 1 terabyte hard drive.
- Do the 2 Terabyte Hard Drive Test operation only if the user chooses the 2 terabyte hard drive.

Try it.

1. Go to the Work Definition work area.
2. Create a work definition for the CTO_474000 item.

3. Add two operations.

Attribute	Value
Name	Test 1 TB Hard Drive
Name	Test 2 TB Hard Drive

4. Add a test for the 1 TB hard drive.
    - On the Edit Work Definition Details page, click the **Test 1 TB Hard Drive** operation, then click **Actions > Edit**.
    - In the Edit Operation dialog, add a check mark to **Option Dependent**.
    - Next to Applicability Rule, click **Add**.
    - In the Applicability Rule dialog, in the Item Structure area, click the **CTO_474300 Hard Drive** component.
    - Drag and drop the CTO_474301 1 TB Hard Drive option onto the Rule Text area.

      Notice the text area adds some code.

      ```
      ITEM='CTO_474000'.'CTO_474300'.'CTO_474301'
      ```

      CTO_474000 is the model, CTO_474300 is the option class for the hard drive, and CTO_474301 is the 1 TB hard drive.

      Here's the hierarchy the code uses.

      ```
      ITEM='model'.'option class'.'option'
      ```
    - Click **Validate**, then make sure the validation succeeds.

      If it fails, then troubleshoot. For details, see *Troubleshoot Problems with Configure-to-Order*.
    - Click **OK**, notice that the Edit Operation dialog displays your rule text, then click **OK**.
5. Repeat step 4 for the 2 TB hard drive, except add your test to the Test 2 TB Hard Drive operation, and drag the CTO_474302 item when you create the rule.
6. On the Edit Work Definition Details page, click **Save**.
7. Test your rule. Go to the Order Management work area, create a sales order, add the CTO_474000 item, choose the 2 TB Hard Drive option, add it to an order line, then click **Submit**.

Related Topics

- How You Edit Work Definitions
- Overview of Work Definitions
- How You Manage Work Centers

Forecasts

Forecast an Assemble-to-Order Item

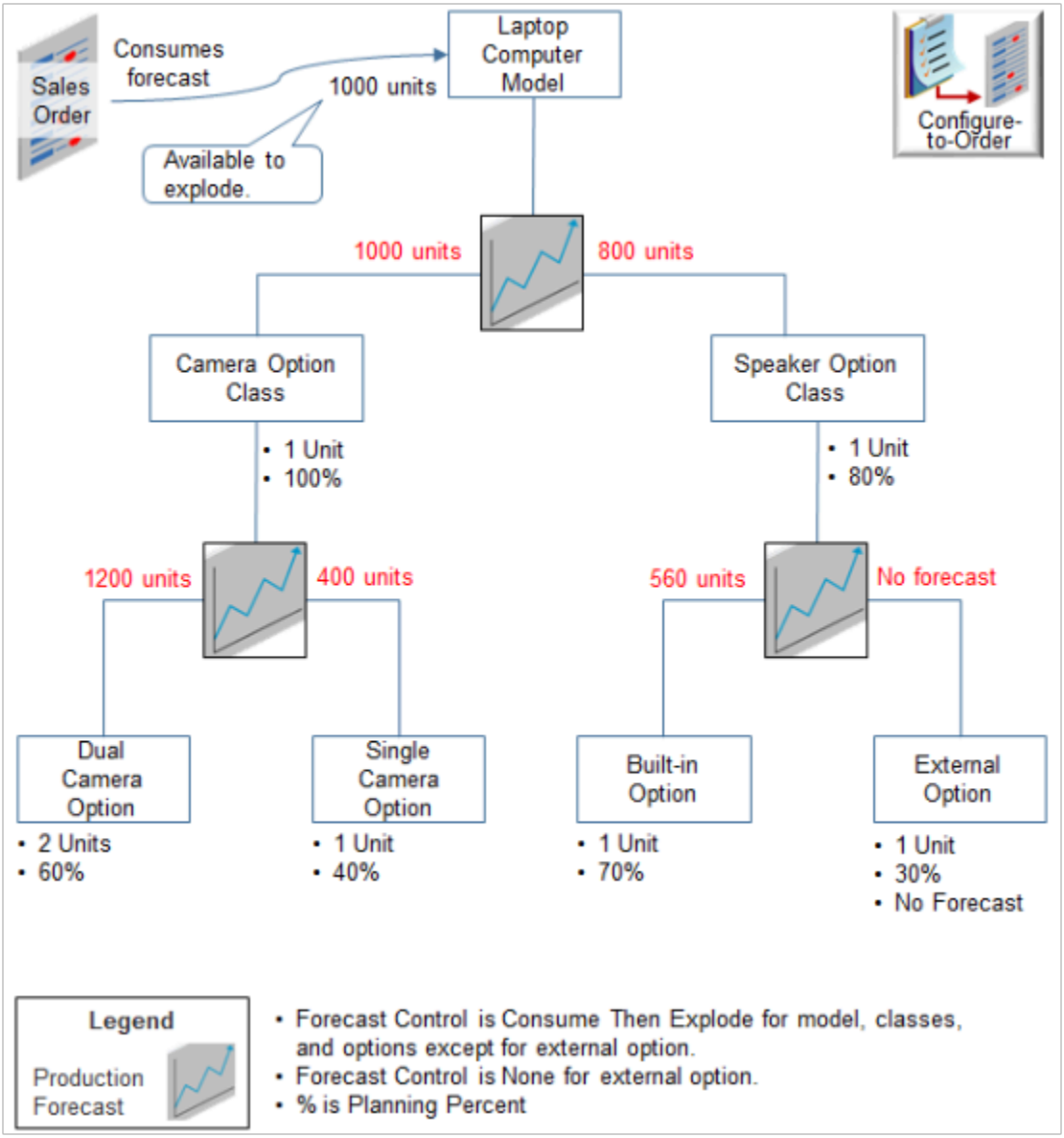
Set up a forecast according to your organization for an assemble-to-order item.

- Use the Planning Central work area to set up the forecast.
- Create a statistical forecast for an assemble-to-order item. Use shipment history and booking history to do the forecast.
- Consume the forecast for your model. Each sales order that contains your configured item consumes the forecast.
- Explode the forecast that remains after the model is done consuming. Generate a production forecast for your option classes and options.

  Planning Central displays the forecast for the options, option classes, and child models as a production forecast. It indicates the forecast that remains after planning consumes the forecast for the model that your sales orders needs to fulfill the configured item.

  Planning uses the planning percent you set on the class and option to forecast demand. You specify the percent when you set up the structure for the item in the Product Information Management work area.
- Create supply for your forecast according to organization. Set up a planning sourcing rule that sources components and subassemblies.
- Planning uses the fixed and variable lead times you set up on the model to offset the production forecast it creates for the options.

Here's an example.



Assume you need organization M1 to manufacture an assemble-to-order model that includes a hierarchy.

- Laptop Computer Model
  - Camera Option Class
    - Dual Camera Option
    - Single Camera Option
  - Speaker Option Class
    - Built-in Option
    - External Option

Assume your market research indicates you expect.

- 100% demand for the camera. Everybody wants one.

- 60% demand for a dual camera and 40% for a single camera.
- 80% demand for high-fidelity speakers.
- Of the 80% who want high-fidelity, 70% want built-in speakers, and 30% want external speakers.

Assume you set up the model.

Object	Forecast Control	Planning Percent
Laptop Computer Model	Consume Then Explode	Not applicable.
Camera Option Class	Consume Then Explode	100
Dual Camera Option	Consume Then Explode	60
Single Camera Option	Consume Then Explode	40
Speaker Option Class	Consume Then Explode	80
Built-in Option	Consume Then Explode	70
External Option	None	30

The configured item in the sales order consumes part of the forecast. Planning explodes what's left over according to your set up. Assume 1,000 units are available to explode after the sales order consumes the configured item.

Here's the math.

Object	Math
Camera Class	The model consumes one unit of the camera class for each laptop. You expect 100% of your customers will want a camera. So planning calculates the production forecast at 1,000. **100% multiplied by 1,000 units that are available to explode equals 1,000.** **1,000 multiplied by a quantity of one equals 1,000.**
Dual Camera Option	The dual camera option needs two cameras for each laptop, so the model consumes two units. You expect 60% of your customers will choose this option. Planning calculates the forecast at 1,200. **60% multiplied by 1,000 units in the camera class equals 600.** **600 multiplied by a quantity of two equals 1,200.**
Single Camera Option	The single camera option needs only one camera for each laptop, so the model consumes one unit. You expect 40% of your customers will choose this option. Planning calculates the forecast at 400. **40% multiplied by 1,000 units in the camera class equals 400.**

Object	Math
	`400 multiplied by a quantity of one equals 400.`

Speaker Class	The model consumes one unit of the speaker class for each laptop. Assume the speaker class allows your user to choose high-fidelity speakers, either built-in or external. You expect 80% of your customers will want high-fidelity speakers. Planning calculates the forecast at 800.
	
	`80% multiplied by 1,000 units that are available to explode equals 800.`
	
	`800 multiplied by a quantity of one equals 800.`

Built-in Option	You need only one set of built-in speakers for each laptop, so the model consumes 1 unit of the built-in option.
	
	You expect 70% of your customers will choose this option. Planning calculates the forecast at 560.
	
	`70% multiplied by 800 units in the built-in class equals 560.`
	
	`560 multiplied by a quantity of one equals 560.`

External Option	You need only one set of external speakers for each laptop, so the model consumes 1 unit of the external option. You expect 30% of your customers will choose this option, but you set the Production Forecast to None for this option. So planning doesn't calculate a forecast.

Assume you set the fixed lead time for the model to 5 days and the variable lead time to 0. Here's how planning explodes the forecast that's due on day 15.

Object	Production Forecast Date	Quantity, in Units
Model	Day 15	1000, after consuming forecast for the model
Camera Class	Day 10	1000
Dual Camera	Day 10	1200
Single Camera	Day 10	400

For details, see *Forecasting for CTO Products*.

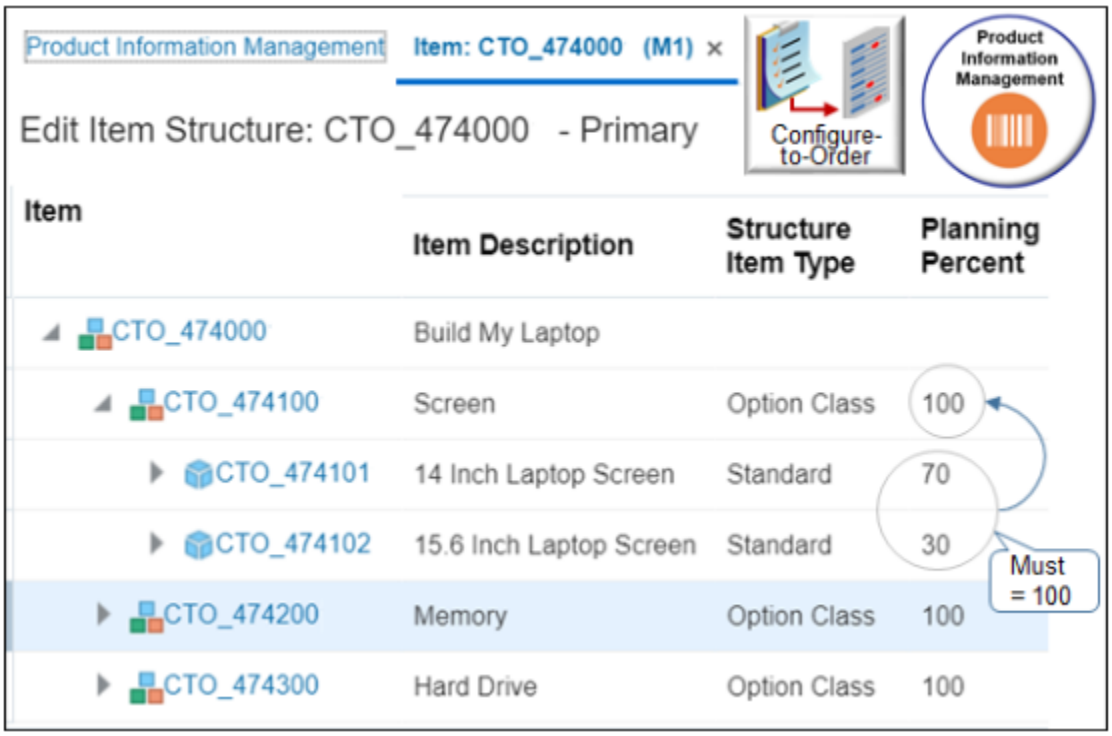
Related Topics

- Overview of Setting Up Configuration Models
- Forecasting for CTO Products

Weight Configure Options When Forecasting Demand

Specify the percent to use when forecasting demand for a configure option.

Assume your market research indicates you expect demand will be about 70% for the 14" screen option and 30% for the 15.6" screen option. Here's your set up.



Note

- The sum of the values you set for configure options in a single class must equal 100%.
- The Planning Central work area uses the value you set when it forecasts demand.

You will modify the CTO_474000 model. Learn how to set it up. For details, see *Create Your Configuration Model*.

Try it.

1. Navigate to your configure option.
    - In the Product Information Management work area, click **Tasks > Manage Items**.
    - On the Manage Items page, search for the CTO_474000 (M1) configuration model, and open it for editing.
    - On the Edit Item page, click **Structures**, then click **Primary**.
    - On the Edit Item Structure page, click **View**, then make sure Component Details contains a check mark.
    - Expand the **Build My Laptop** model, then click the **CTO_474100** screen option.
    - On the Edit Item CTO_474100 page, click **Structures**.
    - In the Name column, click **Primary**.

2. Set the value for the 14" screen.
    - On the Edit Item Structure CTO_474100 - Primary page, expand the **CTO_474100** screen model, click the **row** that contains 14 Inch Laptop Screen in the description column, then click **Actions > Edit**.
    - In the Edit Components Dialog, set the value, then click **OK**.

Attribute	Value
Planning Percent	70

3. Repeat step 2 but set the value for the 15.6" screen.

Attribute	Value
Planning Percent	30

4. Click **Done**.
5. On the Edit Item CTO_474100 page, click **Save and Close**.
6. On the Edit Item Structure: CTO_474000 - Primary page, expand the screen option, then verify your values.

Object	Planning Percent
CTO_474100 Screen	100
CTO_474101 14 Inch Laptop Screen	70
CTO_474102 15.6 Inch Laptop Screen	30

7. Click **Done**, then click **Save**.

Related Topics

- Overview of Setting Up Configuration Models
- Forecasting for CTO Products

Transaction Details

Guidelines for Capturing Transaction Details

Create a transaction attribute to capture details in an Oracle Cloud application where the details change for each transaction.

Oracle Order Management uses a sales order to record details about a transaction you make between you and your customer. Use a transaction attribute to capture details that are specific to each sales order transaction. For example, assume the item is Transmission Service, and License Plate Number is the transaction attribute. The license plate number changes for each transmission service transaction, but you can only capture the license plate number at runtime because its specific to each customer. You don't know who the customer is at design time so you can't use an extensible flexfield that references a predefined value set.

You use an item class to organize your transaction attributes. For example, create an item class named Auto Repair Services, then add transaction attribute Transmission Service to it. Add other relevant transaction attributes to the same class. For example, add transaction attribute Transmission Serial Number to capture the serial number of the transmission you repair.

Note

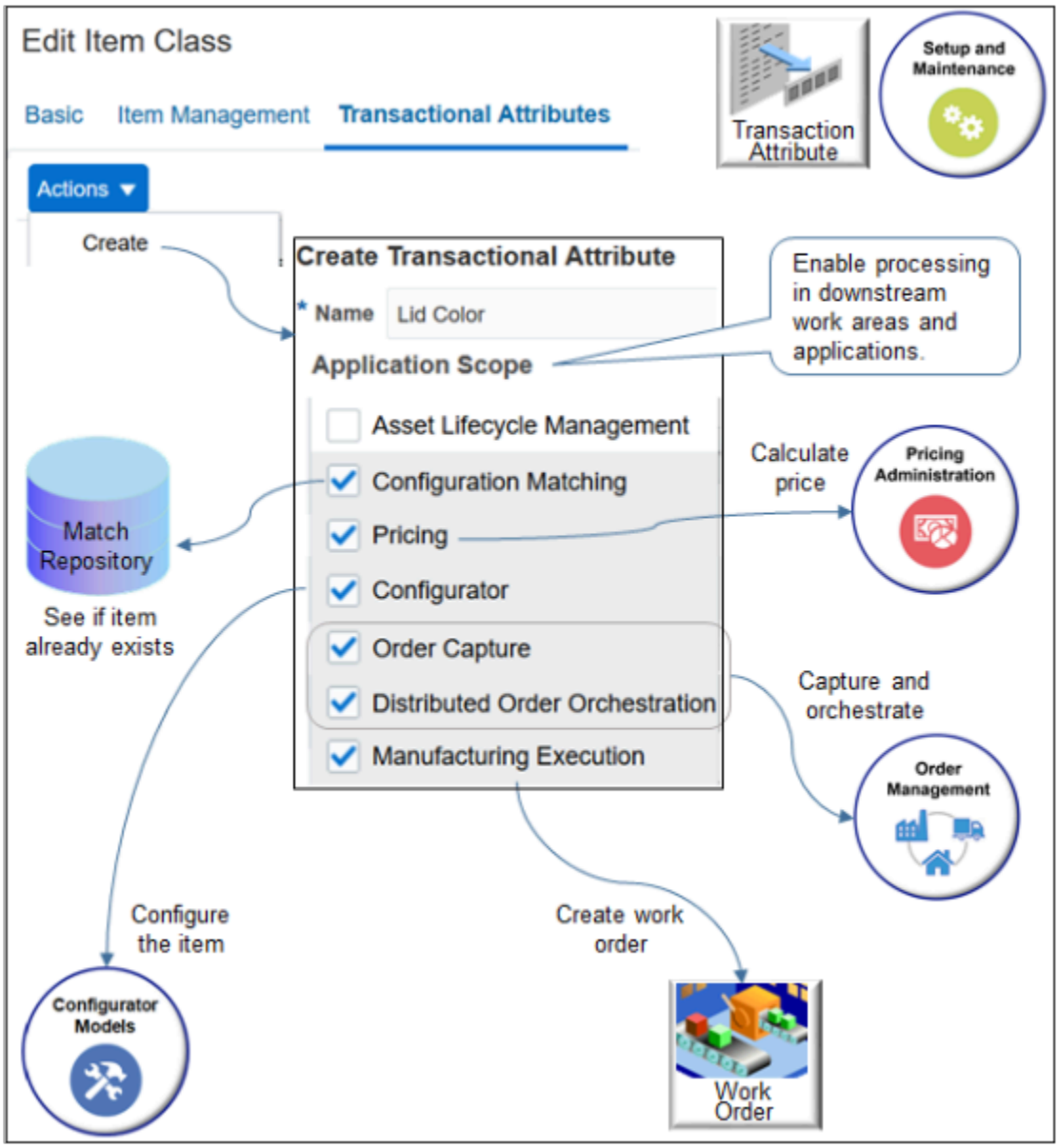
- Use a transaction attribute to store values that change at runtime.
- Set the attribute up as part of the item class. A child class can inherit the attribute, so don't place it on the root item class. A root class might inherit values from a wide range of items, but you typically use a transaction attribute for a specific item. You can't delete a transaction attribute if you assign it to a lower class.
- Supply Chain Orchestration can use a transaction attribute throughout fulfillment, but some fulfillment systems might not support them, such as purchasing or shipping.
- If your value set uses numeric values, then make sure you include a minimum value and a maximum value on the Manage Value Sets page. If you don't, the Configurator Models work area might display an error when you import the model snapshot.

For details, see *Manage Transactional Attributes*.

Learn how to add an attribute that allows your user to add details to a sales order while the user creates the sales order. For details, see *Overview of Using Extensible Flexfields in Order Management*.

Set the Application Scope

Use the Application Scope to choose the applications where you use the transaction attribute.



If you use Oracle Order Management, then you must add a check mark to each of these application scopes. They enable downstream work areas and applications to use your transaction item as part of a configure-to-order flow.

Scope	Description
Configuration Matching	Enable Supply Chain Orchestration to examine the match repository to see if there's a match. If your flow uses the transaction attribute to create a unique configured item, then you must add a check mark to Configuration Matching. If you do, Supply Chain Orchestration will examine the match repository. If it doesn't find a match, it will create a new configuration for each unique attribute value, then save the configuration.
Pricing	Use the values you set in the Pricing Administration work area to price the item.

Scope	Description
Configurator	Configure the item at runtime during order capture.
Order Capture	Capture the sales order that your user creates in the Order Management work area or that you import.
Distributed Order Orchestration	Use the orchestration part of Oracle Order Management, particularly the orchestration process.
Manufacturing Execution	Enable manufacturing to add your configured item to the work order.

Your user must also select an option that your model uses as a transaction attribute when the user configures the assemble-to-order item on the sales order in Order Management.

Capture Details During Transactions

Assume you need to use a transaction attribute to allow your customer to set the color of a stylized pattern to magenta, black, or blue on the lid of the CTO_474000 laptop computer. Your factory then stamps a unique color pattern onto the

Learn how to set up the CTO_474000. For details, see *Create Your Configuration Model*.

Summary of the Setup

1. Create the lookup.
2. Create the value set.
3. Create the transaction attribute.
4. Import your model into the configurator.
5. Test your setup.

Create the Lookup

Create the lookup you will use to provide values your user can choose.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Lookups
2. Click **Actions > New**, set the values, then click **Save**.

Attribute	Value
Attribute	Value
Lookup Type	DOO_COMPUTER_COLOR
Meaning	Color of Laptop Lid

Attribute	Value
Description	Colors for the top of a laptop computer.
Module	Orchestration

3. In the Lookup Codes area, use **Actions > New** three times to add three codes. Make sure Enabled contains a check mark for each code.

Lookup Code	Display Sequence	Meaning
MAGENTA	1	Magenta
BLACK	2	Black
BLUE	3	Blue

4. Click **Save and Close**.

Create the Value Set

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage SCM Common Value Sets
2. On the Manage SCM Common Value Sets page, click **Actions > Create**, then set the values.

Attribute	Value
Value Set Code	Lid Color
Description	Color of the lid of a laptop computer
Module	Supply Chain Management Common Components
Validation Type	Independent
Value Data Type	Character

3. Set the values.

Attribute	Value
From Clause	FND_LOOKUPS
Value Column Name	DESCRIPTION
Description Column Name	MEANING
ID Column Name	LOOKUP_CODE
Enabled Flag Column Name	'Y' You must enclose Y with single quotation marks (').
Start Date Column Name	START_DATE_ACTIVE
End Date Column Name	END_DATE_ACTIVE
WHERE Clause	LOOKUP_TYPE = 'DOO_COMPUTER_COLOR'

4. Click **Save and Close**.

These values filter the Oracle database for the lookup that you created earlier in this topic.

Setup varies for different types of value sets. For details, see *Set Attribute Values Before You Transform Source Orders*.

Create the Transaction Attribute

1. In the Setup and Maintenance work area, on the Setup page, click **Search**.
2. Search for, then open Manage Item Classes.
3. On the Manage Item Class page, in the Item Class column, click the **class** that your item references.
   For this example, assume you set up your item in a class named ATO Models.
4. On the Edit Item Class page, click **Transactional Attributes**.
5. Click **Actions > Create**.
6. In the Create Transactional Attribute dialog, set the values.

Attribute	Value
Name	Lid Color
Internal Name	Lid_Color

Attribute	Value
Description	Color of the lid of a laptop computer
Application Scope	For this example, add a check mark to each choice. - Configurator - Distributed Order Orchestration - Order Capture - Pricing - Configuration Matching
Data Type	Number or String. For this example, choose String.
Value Set	Lid Color Note - Lid Color is the value set you created earlier in this topic. - The value set is optional, but we recommend that you use it. It constrains the values that the transaction attribute can contain. For example, constrain values that your customer can set or enter for the attribute at runtime according to data type or length, or provide a pick list that your customer uses to set the value. - Make sure the values are valid with the model you set up for the item in the Configurator work area. For example, if you use the attribute to allow the user to choose black or purple for the lid, then make sure values in the value set also include Black and Purple.

7. Click **OK**, then click **Save and Close**.

Import Your Model Into the Configurator

1. Go to the Configurator Models work area.
2. Create a workspace.
   - Click **Tasks > Manage Workspaces**.
   - On the Manage Workspaces page, click **Actions > Create**.
   - In the Create Workspace dialog, set the value, then click **Save and Close**.

Attribute	Value
Name	Workspace for Assemble-to-Order Models in M1 (Seattle Manufacturing) You used the M1 organization when you set up the CTO_474000 in Product Information Management, so, for this example, you create a workspace for M1.

- On the Manage Workspaces page, search for your new work space, then click it in the search results.

3. Add your model to the workspace.
   - On the Workspace page, in the Workspace Participants area, click **Actions > Select and Add > Models**.
   - In the Select and Add dialog search for CTO_474000, click it in the search result, add a check mark to Include Updated Item Snapshots For Models, then click **Apply > OK**.

     You add the configuration model. You don't add an option class, configure option, component, or child model.
4. Click **Release**.

If you update a model that you already added to a workspace, then do this instead of the steps above.

1. On the Overview page, click **Tasks > Manage Snapshots**.
2. On the Manage Snapshots page, click **Actions > Import Model Item**.
3. In the dialog, search for and select item CTO_474000, then click **Submit**.
4. In the Confirmation dialog, notice the request ID, click **OK**, wait a moment, search for your item, then verify that the search results displays your item.
5. Click **Tasks > Manage Workspaces**, locate your workspace, then click **Release**.

   If you add a transaction attribute to a model that you already added to a workspace, then you must import the model into the configurator and release the workspace. The import makes the attribute visible in Order Management so the user can select it at runtime.

   If some time later you change the attribute, then you must create a new workspace, add your model, add an updated snapshot for the item class, then release the new workspace.

For details, see *Import Items Into Configurator Models* and *Release Your Workspace*.

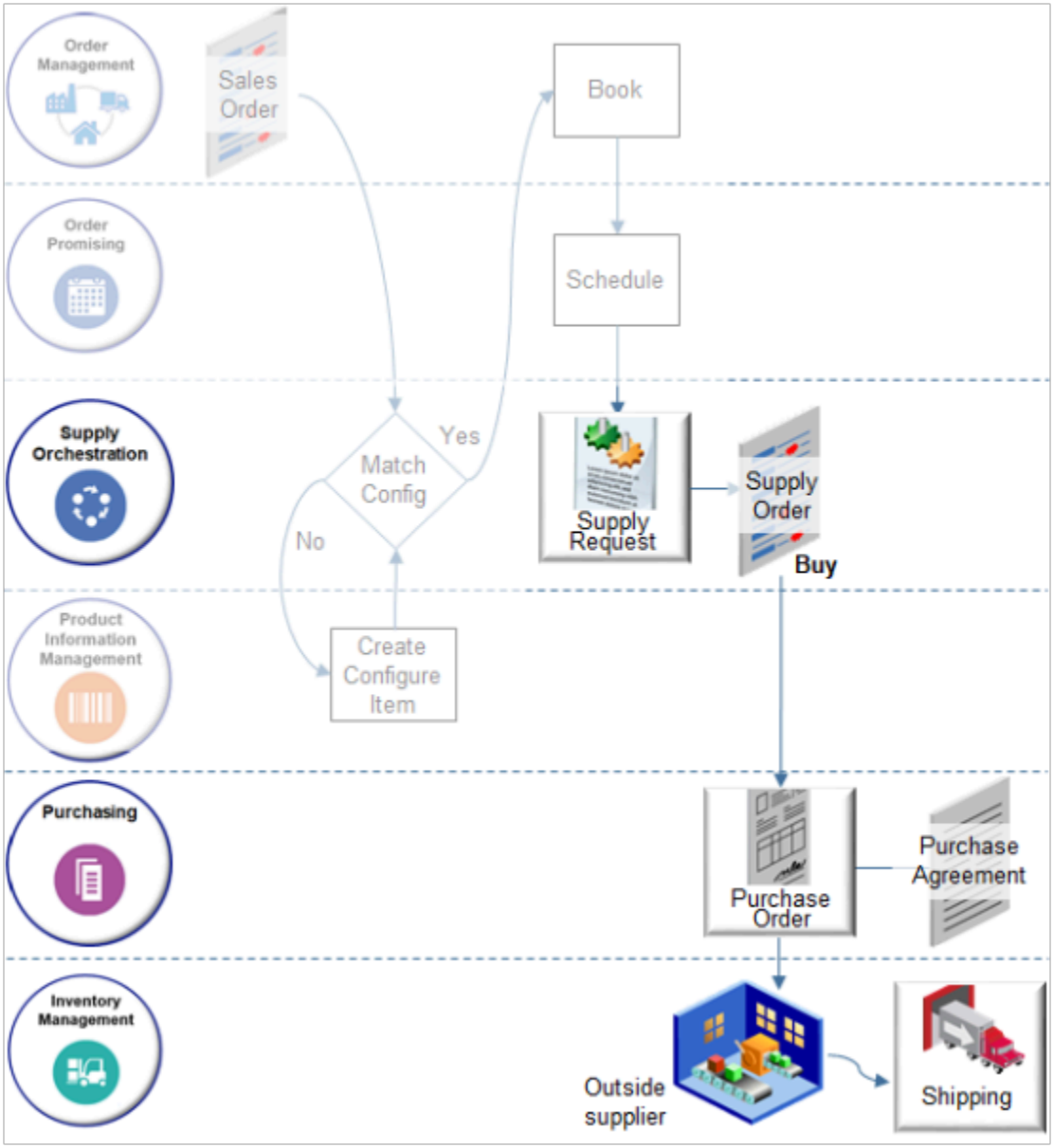
Test Your Setup

1. Go to the Order Management work area, create a sales order, then add the CTO_474000 item to an order line.
2. Verify that the order line displays the Lid Color attribute and sets the default value to Black.
3. Set the value of Computer Color to Magenta, then click **Submit**.
4. Open the work order and verify that it specifies to stamp the pattern onto the laptop lid with magenta ink.

Purchase Orders

How Purchase Orders Work with Configured Items

Learn how a blanket purchase agreement can create a purchase order for a configured item.



The buy flow works the same as the make flow except.

- You set up a blanket purchase agreement with your supplier in Purchasing.
- You create a purchase order in Purchasing instead of a work definition in Inventory Management.
- Your supplier uses a purchase order to supply the item instead of creating the work order in Inventory Management and making the item.
- During the buy flow, if a blanket purchase agreement.
    - **Exists**. Supply Chain Orchestration automatically converts the purchase requisition (PR) to a purchase order (PO).
    - **Doesn't exist**. You must manually do the conversion. For example, you must find the purchase request, then create a document to generate the purchase order. For details, see Use the Purchase Orders Work Area.

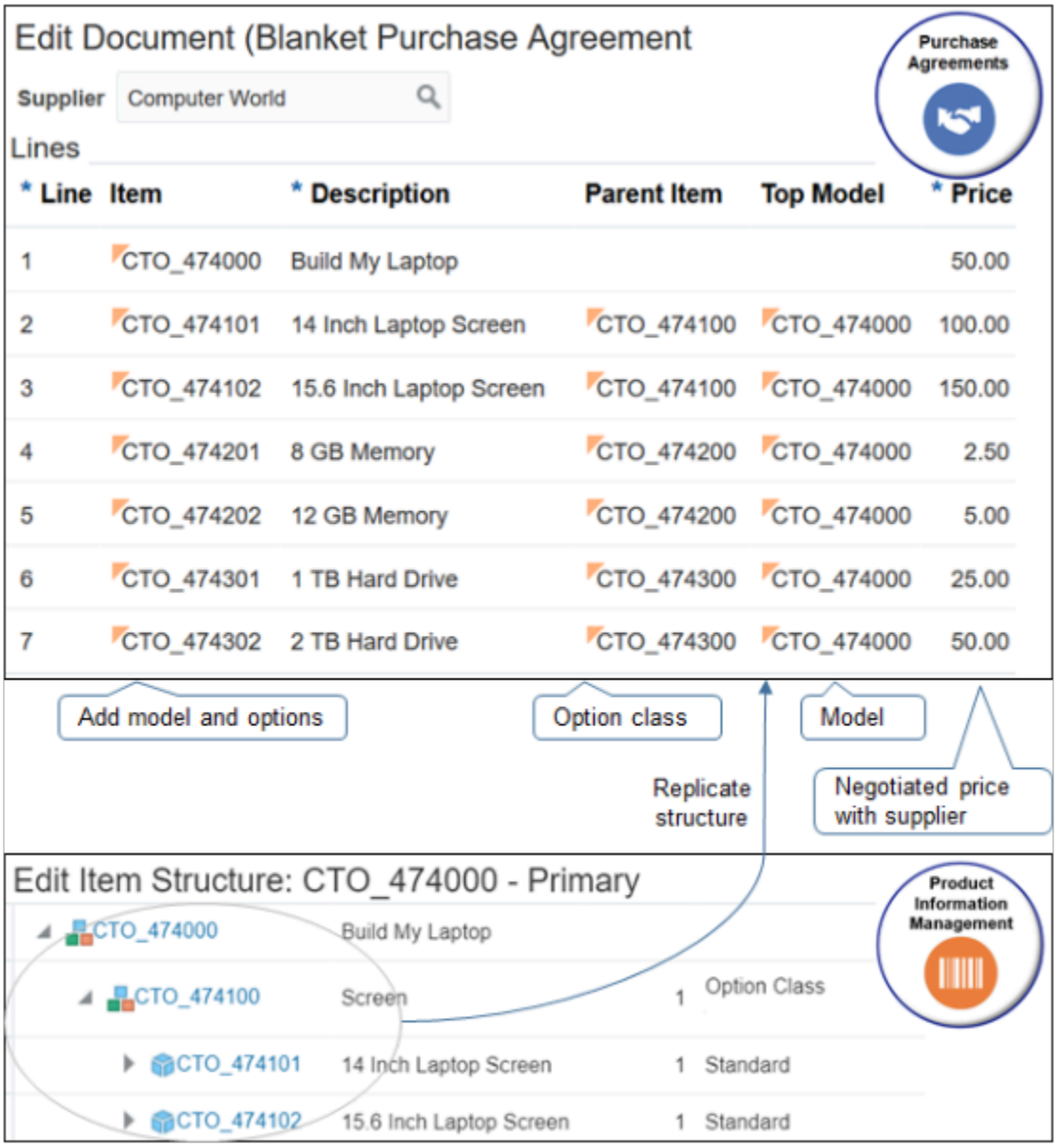
Configure-to-order uses the Configure-to-Order Blanket Purchase Agreement template to create a relationship between a component and the parent. Use the template to set up pricing for the model instead of for each component.

Learn about the make flow. For details, see *How Configure-to-Order Works*.

Create Purchase Orders for Configured Items

Set up a blanket purchase agreement with your supplier to automatically create a purchase order for a configured item.

Here's the agreement you set up in this topic.



Assume you must set up an agreement with your supplier.

- Computer World is your supply. They will supply configuration model CTO_474000 and its configure options.

- You negotiated pricing with your supplier that's 50% of the base price. The base price is the price you charge your customers.

Item	Description	Base Price	Price Negotiated with Supplier
CTO_474100	Build My Laptop	100	50
CTO_474101	14 Inch Laptop Screen	200	100
CTO_474102	15.6 Inch Laptop Screen	300	150
CTO_474201	8 GB Memory	5	2.50
CTO_474202	12 GB Memory	10	5
CTO_474301	1 TB Hard Drive	50	25
CTO_474302	2 TB Hard Drive	100	50

For details, see *Set Up Pricing for Your Configuration Model*.

Note

- You must enable the Configuration Ordering Enabled option on the style you use.
- You can't combine an assemble-to-order item and an item that isn't configured on the same blanket purchase agreement. Instead, create two agreements that cover all the components of a pick-to-order model. Use the configure-to-order template for your assemble-to-order model. Create another agreement for your item that isn't configured.
- The agreement stores price for the assemble-to-order model and uses configure options to procure the configured item. The agreement contains only the models and options, so the flow calculates the purchase price of the configured item each time it procures the item outside of Oracle Applications.
- You must add each configure option to the agreement.
- Don't add option classes to the agreement. The configure options in each class contain pricing, not the class.
- Don't add required items to the agreement. Each required item comes priced as part of the model.
- If necessary, use the Revision attribute on each line to specify the revision of each item.

Summary of the Setup

1. Enable your configuration model for purchasing.
2. Create the agreement.
3. Test your set up.

Enable Your Configuration Model for Purchasing

1. Go to the Product Information Management work area.
2. On the Product Information Management page, click **Search**, search for CTO_474000, then open it for editing.

   For details, see *Create Your Configuration Model*.
3. On the Edit Item page, click **Specifications**.
4. In the Item Organization area, click **Purchasing**, then set the value.

Attribute	Value
Purchased	Contains a check mark.

5. Click **Categories**, then verify that the Catalog column contains a row for the Purchasing catalog. If it doesn't, then click **Actions > Add Row**, and add it.

Attribute	Value
Catalog	Purchasing
Controlled At	Master Level
Category	Miscellaneous_1
Category Code	MISC.MISC

6. Click **Save**.
7. Repeat this procedure for each of your option classes and configure options.

Create the Agreement

1. Open another browser application, then sign in with a user who has privileges to set up purchasing.

   If you don't sign in with these privileges, then the work areas you use in this procedure don't display.
2. Set up your document style.

   Use the document style to control the look and feel of the application so it matches the usage of the purchasing document.

   You can typically use the predefined Configure to Order Blanket Purchase Agreement style. Do this step only if you can't find a predefined style that meets your needs.
   - Go to the Setup and Maintenance work area, then go to the task.
     - Offering: Procurement
     - Functional Area: Procurement Foundation

- Task: Manage Document Styles

- On the Manage Document Styles page, click **Actions > Create**.
- On the Create Document Style page, set the values, then click **Save and Close**.

Attribute	Value
Style	Style for Configure-to-Order Item
Configuration Ordering Enabled	Contains a check mark. Instructs the Purchase Agreements work area to display the Parent Item attribute and Top Model attribute.
Display Name in the Purchase Order area	Purchase Order for Configure-to-Order Item
Display Name in the Blanket Purchase Agreement area	Blanket Purchase Agreement for Configure-to-Order Item
Display Name in the Contract Agreement area	Contract Agreement for Configure-to-Order Item

3. Create the agreement.

- Go the Procurement work area, then click **Purchase Agreements**.
- Click **Tasks > Create Agreement**.
- In the Create Agreement dialog, set the values, then click **Create**.

Attribute	Value
Style	Blanket Purchase Agreement for Configure-to-Order Item This value identifies the style you created in step 2.
Supplier	Computer World
Supplier Site	Choose the address where your supplier ships the item from.

4. Add the configuration model to your agreement.
    - In the Lines area, click **Actions > Add Row**, set the values, then click **Save**.

Attribute	Value
Line	1
Type	Goods
Item	CTO_474000
Description	Build My Laptop
Parent Item	Leave empty Specify the number or code you use to identify the parent of the item you're adding on this line. If you add a configure option, then specify the option class that contains the option. You are adding the model, which doesn't have a parent, so leave it empty.
Top Model	Leave empty Specify the number or code that you use to identify the configuration model. You are adding the model, so leave it empty.
Price	50 Note - Enter the purchase price for the item you're adding on the line. It's the price you negotiated with Computer World, your supplier. - You can enter zero or a negative value to reflect a discount for an agreement that you use to price and source a configured item. - If you provide a price for a configure option, then the flow applies the price only if your customer chooses the option when ordering the configured item.

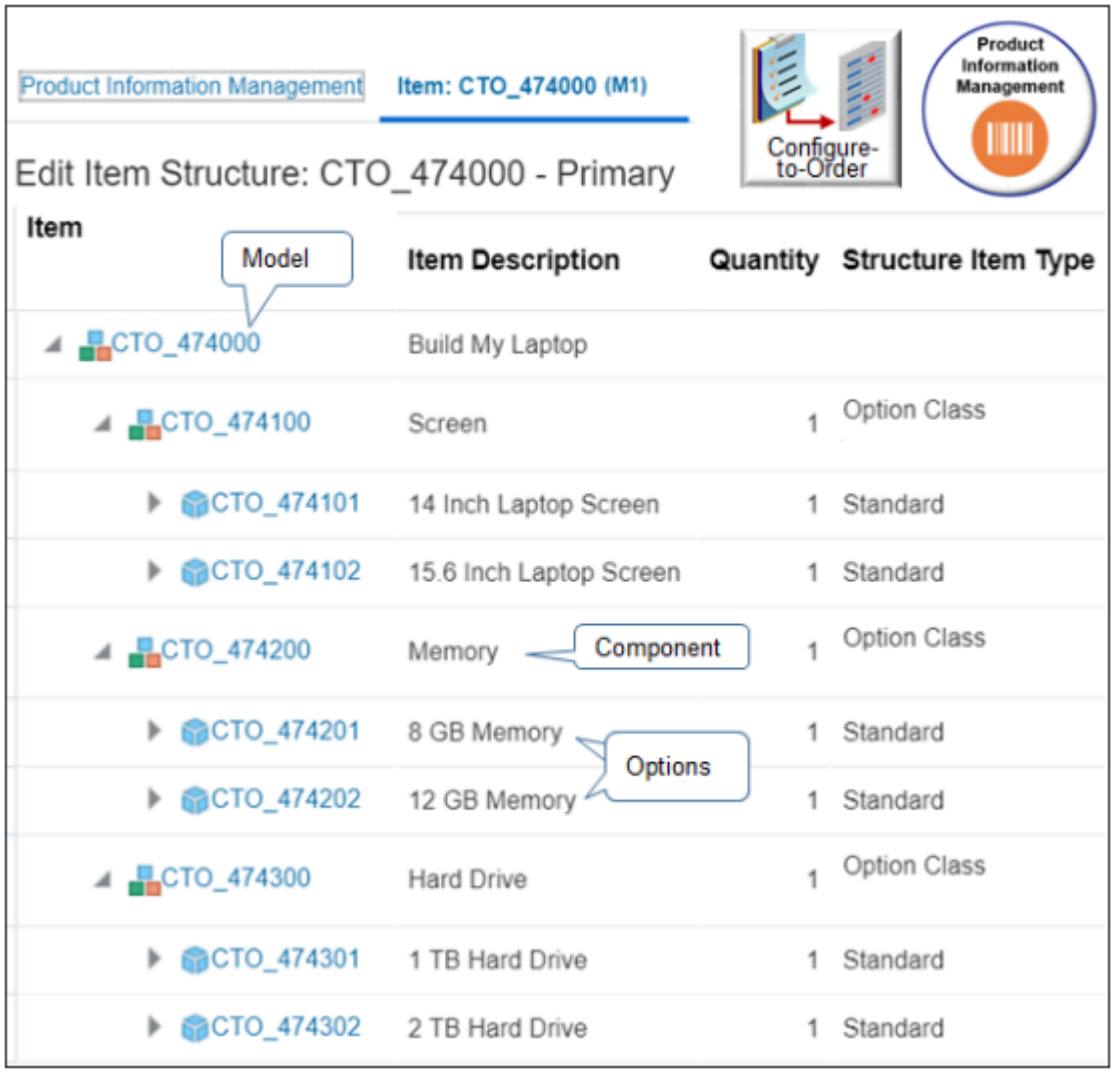
5. Add the configure options.
    - Add a separate row for each option.
    - Set the Type to Goods for each row.
    - Use the Parent Item attribute and Top Model attribute to replicate the structure of your configuration model.

Here's what your set up should look like.

Line	Item	Description	Parent Item	Top Model	Price
1	CTO_474100	Build My Laptop	Not applicable	Not applicable	50
2	CTO_474101	14 Inch Laptop Screen	CTO_474100	CTO_474100	100
3	CTO_474102	15.6 Inch Laptop Screen	CTO_474100	CTO_474100	150
4	CTO_474201	8 GB Memory	CTO_474200	CTO_474100	2.50
5	CTO_474202	12 GB Memory	CTO_474200	CTO_474100	5
6	CTO_474301	1 TB Hard Drive	CTO_474300	CTO_474100	25
7	CTO_474302	2 TB Hard Drive	CTO_474300	CTO_474100	50

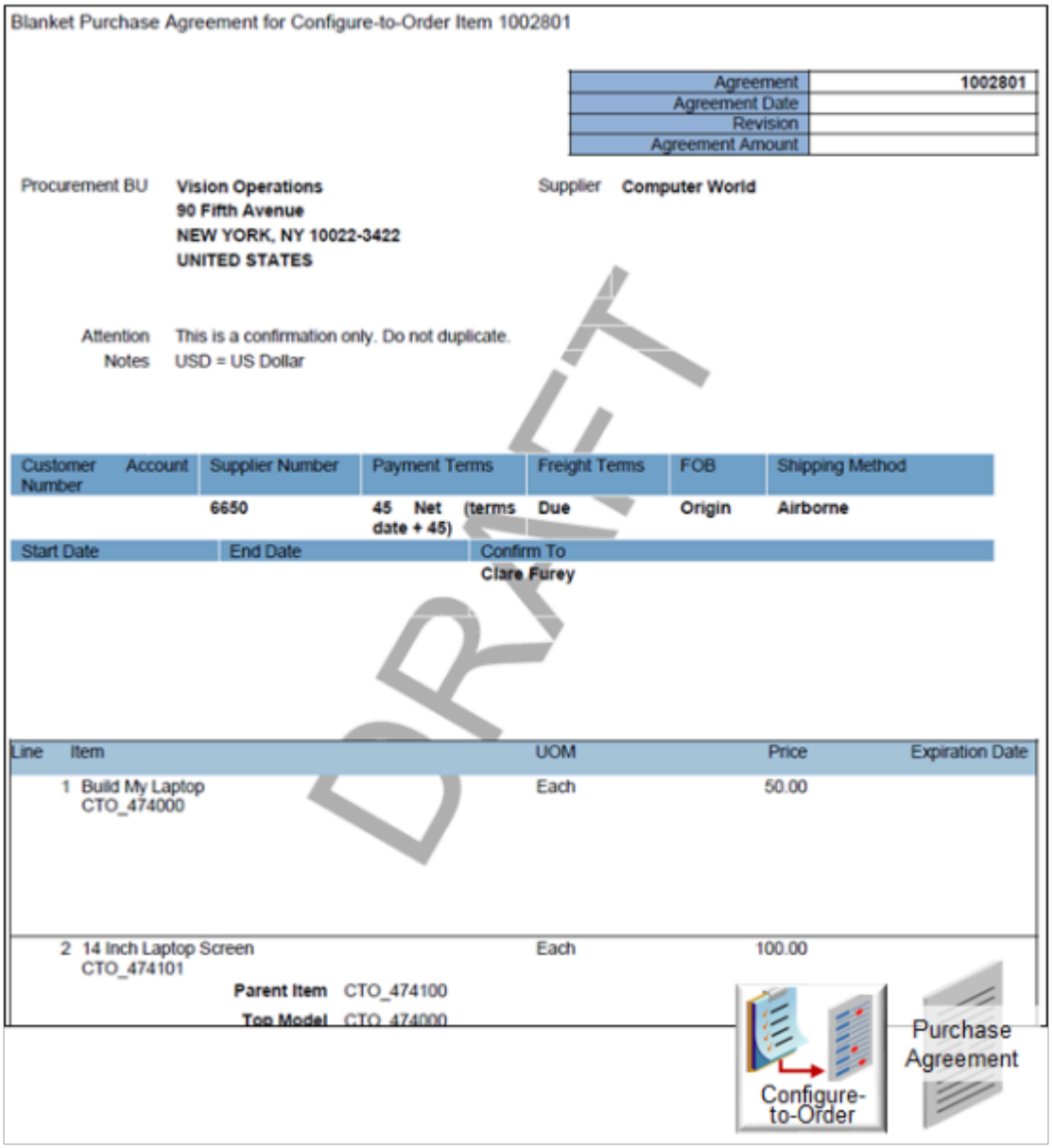
Line	Item	Description	Parent Item	Top Model	Price

For reference, here's the structure you replicate.



6. At the top of the page, click **View PDF**, then open it.

7. Examine the output. Make sure it contains your model, options, and pricing. The buy flow sends this document to your supplier at runtime.



For brevity, this screen print includes only the model and one option. The actual document includes a cover letter and another page that has the rest of the options.

8. Click **Save**, then sign out.

Don't click Submit. Submit sends the agreement to approval. If you clicked Submit, then query the document on the Manage Agreements page and cancel it.

Test Your Setup

Create a Sales Order

1. Go to the Order Management wok area, then create a new sales order.

Attribute	Value
Customer	Computer Service and Rentals

2. On the catalog line, search for CTO_474100.
3. Click **Configure and Add**, set some options, click **Finish**, then click **Submit**.

Examine the Purchase Order

1. Click **Actions > Switch to Fulfillment View**.
2. Click **Fulfillment Lines**.
3. In the Attributes area, click **Supply Details**, click the **link** next to Supply Order Number, then examine the details.
4. Click the **supply tracking line**, then examine the orchestration plan.

   The flow created the purchase order and reserved inventory for the sales order.
5. In the Item Details area, click **Buy**.
6. Click the **arrow** next to the purchase requisition number, then examine the requisition details.
7. Click the **arrow** next to the purchase order number, then examine the purchase order details.

   The value in the Document Number column identifies the purchase order number that the flow will process to fulfill the sales order. Make a note of this value. You use it soon. For this example, assume the value is 58459.
8. Click **View PDF** at the top of the page, save it, open it, and examine the contents.
9. Click **Done**.

Verify Configuration Details

1. Click **Tasks > Manage Orders**.
2. Search for purchase order 58459, then open it.
3. In the Lines area, click **View Configuration**.
4. Verify that the pricing uses the values you set when you created the blanket purchase agreement, then click **Done**.
5. Click **View PDF**, then verify that the configuration detail displays the structure for your configured item.

Receive Purchase Order for Configured Item

Assume your supplier shipped the purchase order for your configured item. You must now receive it, then put the item away in the destination warehouse.

Receive the item in the destination warehouse.

1. Receive the item.
   - Go to the Receipts work area.
   - Click **Tasks > Receive Expected Shipments**.

- Enter the purchase order for the configured item, then click **Search**.
- Select the **record**, then click **Receive**.
- Click **Show Receipt Quantity** to default the quantity entered for the purchase order.
- Click **Create Receipt**, then click **Submit**.
- Note the receipt number.
- Click **OK > Done**.

2. Put away the receipt.
   - Click **Tasks > Put Away Receipts**.
   - Enter the purchase order number, then click **Search**.
   - Select the record, click **Put Away**, then click **Submit**.
   - Click **OK > Done**.
3. Examine the on-hand quantity in the destination organization.
   - Go to the Inventory work area.
   - In the Advanced Search area, enter the configured item you just received, then click **Search**.
   - Click **Expand** next to the organization to examine the on-hand inventory in your subinventories.
   - Click **Expand** next to the subinventory that received the item.
   - Verify that the on-hand quantity is correct.

More

View Structures of Configured Items

Examine the values of a configured item without opening the sales order or work order.

The user, such as an Order Entry Specialist, sets configure options for a configured item in the sales order. The work order includes the configured item, required components, configure options, substitutions, and transactional item attributes.

You use the Product Information Management work area to set up a configured item. You can't use it to view how the user sets configuration options because the work area doesn't store the bill of materials for the configured item. Fulfillment uses values that the user sets at runtime to create the bill of materials. You can use the View Configured Item Structure page to examine the configured item the user set up without accessing the sales order.

Use the Item attribute in the Search area to search for the configured item, then use the Item Structure Details area to drill down to the configured item and configure options.

Use the Configured Item Service

You can also use the ConfiguredItemService web service to get the item structure.

- For details, see *Overview of Using Web Services with Configure-to-Order*.

- You must make sure the required components and configure options for each configured item are the same across organizations.

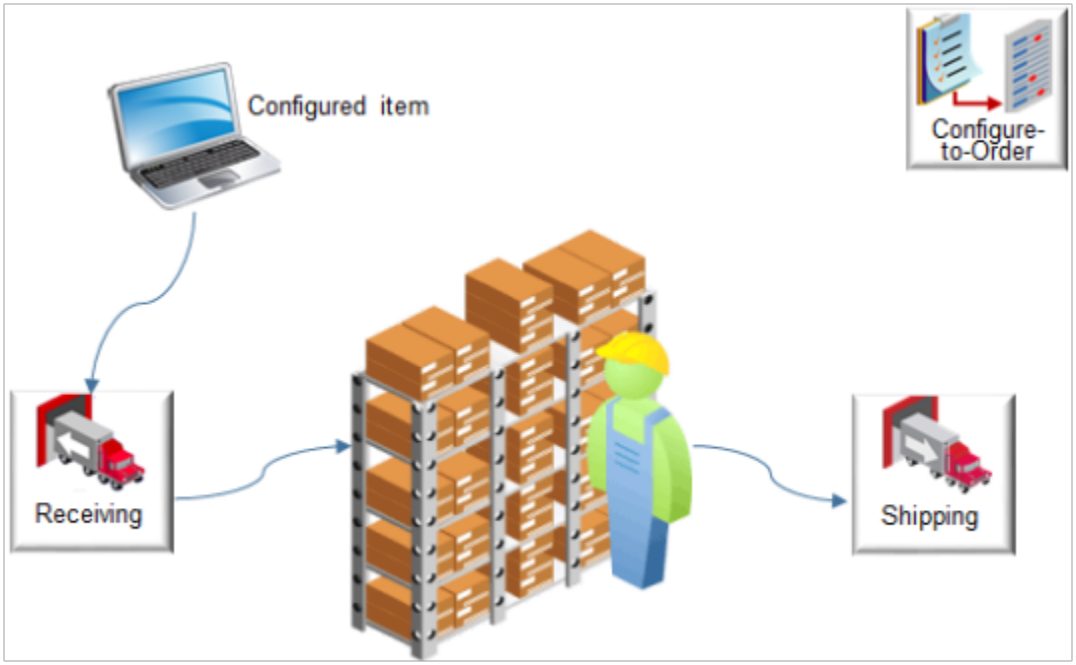
Related Topics

- Overview of Configure-to-Order

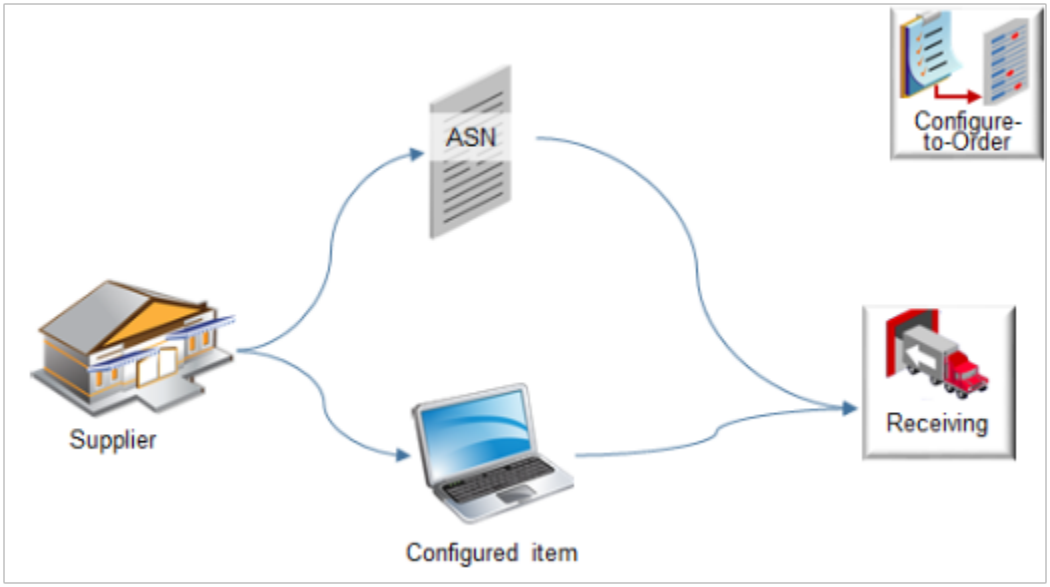
Monitor Downstream Fulfillment

A configured item meets the needs of a specific sales order. So the make flow picks and moves it directly to a shipping lane from receiving instead of putting it into stock. The receiving agent can view the sales order.

The flow receives the configured item into inventory, picks, then ships it just like it does any other item in inventory. The configured item doesn't need a unique process or special treatment to transact it in your shipping application.



The flow also supports an advanced shipment notice (ASN) that it receives from your supplier or during an internal material transfer from a shipping organization.



Assume.

- You finished creating your model. For details, see *Create Your Configuration Model*.
- Manufacturing finished the work order. Inventory contains the configured item and Shipping contains shipment lines for the configured item.
- You set up the pick release process for the organization so it automatically pick confirms and creates the shipment.

Next, put on your Shipping Manager hat (pretend you're the Shipping Manager). You receive an urgent request to fulfill a configured item, so you manually pick release and ship confirm it.

You will navigate to the Shipments work area and use it to search for shipments and shipment lines. You can also click graphs that display the outstanding and completed work for the current day. This is where you determine the work activities to do and drill into an activity.

Summary of the Setup

1. Release and confirm the shipment.
2. Examine and print reports.
3. Track sales order progress through the fulfillment lifecycle.

Release and Confirm the Shipment

1. Sign in with a user who has privileges to access the Warehouse Operations work area.
   If you don't have the privileges, the work areas you use in this topic don't display.
2. Go to the Shipments work area.
3. Accept the All default value in the dialog, then click **OK**.
   The Shipments work area displays. Get shipments or shipment lines according to status. Click a status on the Shipments or Shipment lines chart, or click a number link in one of the tables below the charts.
4. Click **Tasks > Manage Shipment Lines**.
5. In the Advanced Search area, enter the order number in the Order attribute, then click **Search**.
6. On the Edit Shipment Line page, verify that the shipment line status is Ready to Release.
7. Click **Actions > Pick Release**, click **OK** in the confirmation dialog, then click **Save and Close**.
8. On the Manage Shipment Lines page, click **Save** to refresh the page. Repeat until the flow creates the shipment and the line status displays Staged.
9. Click Shipment and note the shipment number on the Edit Shipment page.
10. Set the weight and volume, then click **Ship Confirm**.
11. Click **Yes** in the warning dialog.
    The flow confirms the shipment.

Examine and Print Reports

You can use the Print Commercial Invoice Report scheduled process.

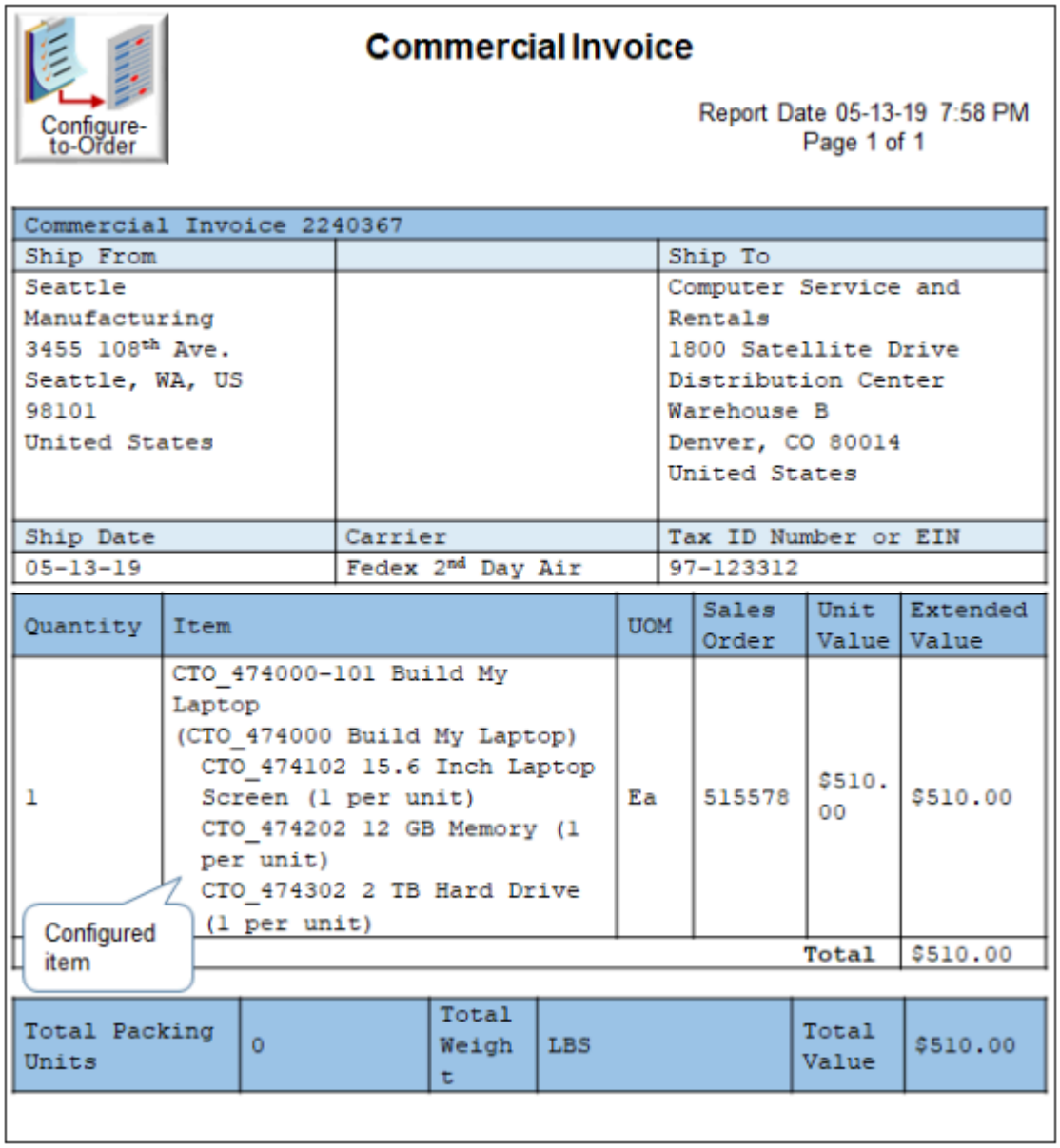
1. Go to the Scheduled Processes work area.
2. On the Overview page, in the Name column, notice the reports that ship confirm created.
   - Print Packing Slip Report
   - Manage Shipment Interface
   - Print Bill of Lading Report
3. Examine the Print Packing Slip Report.
   - Click **Print Packing Slip Report**.
     The Packing Slip Report displays the details of the configured item.
   - Click the output name in the Output area of the Print Packing slip report.
     Ship confirm doesn't automatically submit the Commercial Invoice. You must manually submit it.
4. Go to the Scheduled Processes page, then click **Schedule New Process**.
5. On the Schedule New Process page, set the value.

Attribute	Value
Name	*Print Commercial Invoice Report*

6. In the Process Details dialog, set the values, then click **Submit**.
   - Enter the ship-from organization, such as 002 Atlanta.

- Enter the same shipment number you entered earlier in this procedure when you used the Parameters section of the dialog.

7. Wait for the status to change to Succeeded.
8. Examine the invoice.



Configure-to-Order

Commercial Invoice

Report Date 05-13-19 7:58 PM
Page 1 of 1

Commercial Invoice 2240367		
Ship From		**Ship To**
Seattle Manufacturing 3455 108th Ave. Seattle, WA, US 98101 United States		Computer Service and Rentals 1800 Satellite Drive Distribution Center Warehouse B Denver, CO 80014 United States
Ship Date	**Carrier**	**Tax ID Number or EIN**
05-13-19	Fedex 2nd Day Air	97-123312

Quantity	Item	UOM	Sales Order	Unit Value	Extended Value
1	CTO_474000-101 Build My Laptop (CTO_474000 Build My Laptop) CTO_474102 15.6 Inch Laptop Screen (1 per unit) CTO_474202 12 GB Memory (1 per unit) CTO_474302 2 TB Hard Drive (1 per unit)	Ea	515578	$510.00	$510.00
				Total	$510.00

Configured item

Total Packing Units	0	Total Weight	LBS	Total Value	$510.00

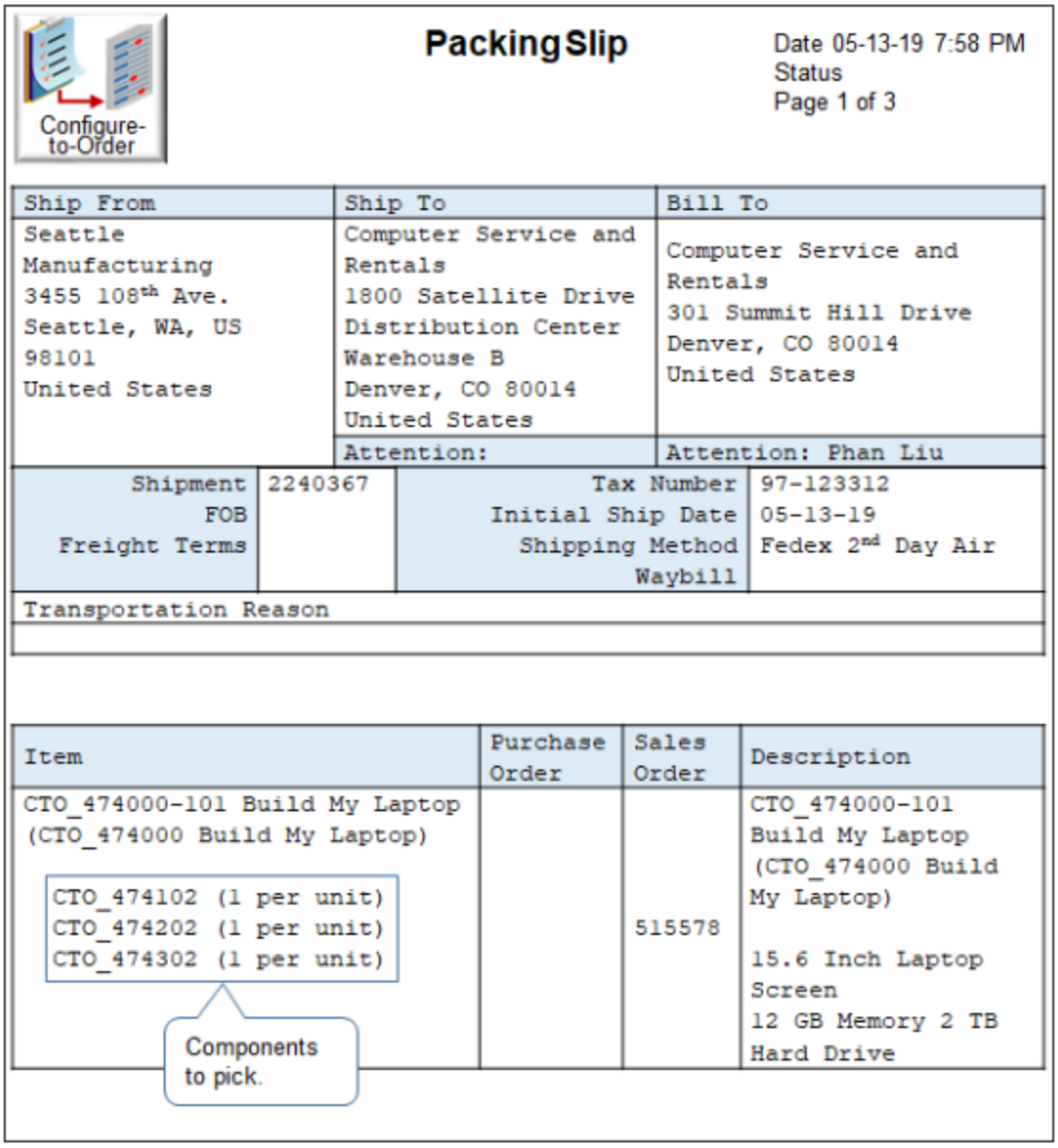
The invoice includes.

- The configured item, CTO_474000-101 Build My Laptop
- The model, CTO_474000 Build My Laptop
- Each option your user chose, such as CTO_474102 15.6 Inch Laptop Screen (1 per unit)

Note

- It isn't necessary to do any more set up to get and display the invoice.
- A web service in Supply Chain Orchestration gets details for this sales view when you ship confirm the sales order.
- You can view the model and options for the configured item on the commercial invoice and the packing slip.
- The invoice displays the quantity for each component in the configured item assuming that the configured item quantity is one.
- You don't have to do any special document setup for configured items.

Here's an example of the packing slip.



Track Sales Order Progress Through the Fulfillment Lifecycle

1. Go to the Order Management work area.
2. On the Overview page, enter the order number, then click **Search**.
   Notice the order line status is Awaiting Billing.
3. Click **Action > Switch to Fulfillment View**.
4. Click **Fulfillment Lines**, then, in the Attributes section at the bottom of the page, click **Supply Details**.
5. Click **Supply Order Number** to view the supply order details.
6. Click the **supply tracking line**, then notice the value of the status.

Control Explosion Dates for Configuration Models

Specify the date when you want order fulfillment to explode a configuration model.

Exploding the model is the act of breaking apart each assembly or subassembly into its component parts. You explode to improve order fulfillment efficiency.

You explode so you can get the structure of a configured item from Product Information Management. Explosion updates the structure, including items that the configuration model references. You can use the Configuration Effective Date for Exploding Included Items parameter to explode components in a model, such as configure options. For details, see *Manage Order Management Parameters*.

You can control the explosion date for:

- The ordered date in a kit
- The included items and current date in a pick-to-order model
- More than one split line and the current date

A bill-of-material (BOM) is a structure you use to store lists of items that you associate with a parent item. The structure stores details that describe how to relate each item to the parent. You arrange child items in the structure hierarchically so they reflect the composition of the parent item. For example, a laptop computer is the parent, and the hard drive, screen, and memory are each a child item.

Examples

Assume you have a bill-of-materials for the CTO_474000, Build My Laptop parent item.

Child Item	Description
CTO_474101	14 Inch Laptop Screen
CTO_474102	15.6 Inch Laptop Screen
CTO_474201	8 GB Memory
CTO_474202	12 GB Memory
CTO_474301	1 TB Hard Drive

Child Item	Description
CTO_474302	2 TB Hard Drive
CTO_474100	Screen
CTO_474200	Memory
CTO_474300	Hard Drive
CTO_474400	My Mouse Pad

You need to explode this bill of materials so you can pick each item separately, then assemble them into the parent CTO_474000. You pick items separately so you can more quickly fulfill them item.

For another example, assume you need to pick a newer part for an item from inventory instead of an older part. Assume you sell laptop computers to college students. You manufacture a mouse pad every week, and the pad is an included part in a kit. As part of a promotion, you stamp the name and logo of a popular music group on the pad. You stamp the pad each week with the group that's currently #1 on the Worldwide Music Singles Chart. The chart changes every week, but some groups return to the #1 spot several times in subsequent weeks with the same song or a different song, so you keep older pads in stock in case they return to the #1 spot. You use the Configuration Models work area to change the mouse pad, item CTO_474400, in parent model CTO_474000, Build My Laptop.

Here are some more examples.

- You prefer to use last-in-first-out (LIFO) where you consume the item you most recently manufactured rather than an older one.
- Assume Order Management sends a change request for a line. If the line is awaiting shipping, then you must make sure fulfillment doesn't change an existing child item.

Set the Order Management Parameters

You use the Configuration Effective Date for Exploding Included Items parameter to control which model to explode.

Value	Explode the Model That Exists When Fulfillment Creates:
Configuration Effective Date	The Configuration
Parent Creation Date	The Parent

Note

- The Product Information Management work area stores a different version each time you revise the model. The sales order uses the Configuration Effective Date parameter to determine which model to use.
- Fulfillment skips closed or canceled lines. Instead, it copies child items from the original model.
- This parameter doesn't affect the current explosion behavior when the user creates the sales order.
- Use this parameter only for a bill-of-material explosion.

- Don't change the validation behavior for configured items.
- You also use the Configuration Effective Date parameter to control behavior.
- Learn about the attributes. For details, go to *SOAP Web Services for Oracle Fusion Cloud SCM*, expand **Item Structure Version 2**, then click **Item Structure Explosion**.

Split Lines

You can split a fulfillment line so you can use a new item. You can specify whether to use the date when fulfillment creates the configuration for the bill-of-material structure, or the date when it creates the parent.

When you submit a revision or pick-to-order model you don't want to explode the existing kit and add new included items to it. Instead, you only want to explode the new kit.

You can use only one open item when you split the quantity where you reduce the quantity and add a new line for the split quantity. Assume the original quantity on line 1 is 10. You split line 1 into line 1 and line 2. You reduce the quantity on line 1 to 7 and set the quantity on line 2 to 3.

If you set the Configuration Effective Date parameter to Ordered Date, then fulfillment uses the same included item.

Assume you change the bill of materials for the kit and need to explode on the date when you make the change. If you set Configuration Effective Date for Exploding Included Items to Parent Creation Date, then fulfillment explodes the new line according to the date you make change instead of the date that Configuration Effective Date specifies.

Example of a Kit

Assume you create sales order 58697 on June 12 that includes the Movie Basket kit. The kit includes one fulfillment line, and the line includes two items, Popcorn and Ice Cream.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	Movie Basket	Kit	10	June 12	June 12	June 12
-	Popcorn	Included Item	10	June 12	June 12	June 12
-	Ice Cream	Included Item	10	June 12	June 12	June 12

Assume you use the Configuration Models work area on June 15 to change the model for the kit. You replace Ice Cream with Big Drink.

On June 20, you revise order 58697. You add a new fulfillment line that includes another Movie Basket kit.

Continue reading to see what happens.

Use the Ordered Date

Next, assume you set the Configuration Effective Date parameter to Ordered Date.

Here's what happens.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	Movie Basket	Kit	10	June 12	June 12	June 12
-	Popcorn	Included Item	10	June 12	June 12	June 12
-	Ice Cream	Included Item	10	June 12	June 12	June 12
2	Movie Basket	Kit	10	June 12	June 20	June 12
-	Popcorn	Included Item	10	June 12	June 20	June 12
-	**Ice Cream**	Included Item	10	June 12	June 20	June 12

The sales order uses the Configuration Effective Date of June 12 for fulfillment line 1 and fulfillment line 2. The Ordered Date is June 12, so the order doesn't use the new configuration you revised in the model. It continues to use Ice Cream instead of Big Drink.

Use the Parent Creation Date

Assume you set the Configuration Effective Date for Exploding Included Items parameter to Parent Creation Date.

Here's what happens.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	Movie Basket	Kit	10	June 12	June 12	June 12
-	Popcorn	Included Item	10	June 12	June 12	June 12
-	Ice Cream	Included Item	10	June 12	June 12	June 12
2	Movie Basket	Kit	10	June 12	June 20	June 20
-	Popcorn	Included Item	10	June 12	June 20	June 20
-	**Big Drink**	Included Item	10	June 12	June 20	**June 20**

The sales order uses a Configuration Effective Date of June 12 for fulfillment line 1. The Movie Basket in fulfillment line 2 is the parent, you created it on June 20, so the sales order uses a Configuration Effective Date of June 20 for fulfillment line 2. You replaced Ice Cream with Big Drink in the configuration model on June 15, so the sales order includes Big Drink instead of Ice Cream on line 2.

Pick-to-Order Example

Assume you create sales order 53867 on June 12 that includes the My Phone pick-to-order model, and it includes one fulfillment line.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	My Phone	Pick-to-Order	10	June 12	June 12	June 12
-	Wallet Case	Option Class	10	June 12	June 12	June 12
-	Charger	Included Item	10	June 12	June 12	June 12

Assume you use the Configuration Models work area on June 15 to change the model. You replace Charger with Wireless Charger.

On June 20, you revise order 53867. You add a new fulfillment line that includes another My Phone model.

Continue reading to see what happens.

Use Current Date

Assume you set the Configuration Effective Date parameter to Current Date.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	My Phone	Pick-to-Order	10	June 20	June 12	June 20
-	Wallet Case	Option Class	10	June 20	June 12	June 20
-	**Wireless Charger**	Included Item	10	June 20	June 12	**June 20**
2	My Phone	Pick-to-Order	10	June 20	June 20	June 20
-	Wallet Case	Option Class	10	June 20	June 20	June 20
-	Wireless Charger	Included Item	10	June 20	June 20	**June 20**

The Current Date is the date that when you click Revise or Submit.

The sales order uses the Current Date of June 20 for the Configuration Effective Date for line 1 and line 2. So, the sales order uses the new configuration you revised in the model on June 15. It uses Wireless Charger instead of Charger on both lines.

Use Ordered Date and Parent Creation Date

Assume you set.

Order Management Parameter	Value
Configuration Effective Date	Ordered Date
Configuration Effective Date for Exploding Included Items	Parent Creation Date

Here's what happens.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	My Phone	Pick-to-Order	10	June 20	June 12	June 12
-	Wallet Case	Option Class	10	June 20	June 12	June 12
-	**Charger**	Included Item	10	June 20	**June 12**	June 12
2	My Phone	Pick-to-Order	10	June 20	June 20	June 20
-	Wallet Case	Option Class	10	June 20	June 20	June 20
-	**Wireless Charger**	Included Item	10	June 20	**June 20**	June 20

The sales order uses the Parent Creation Date. The My Phone is the parent for line 1. You created line 1 on June 12, so the sales order uses the model that existed on June 12 for line 1. That model uses Charger, not Wireless Charger.

The My Phone is the parent for line 2. You created line 2 on June 20, so the sales order uses the model that exists on June 20 for line 2. That model uses Wireless Charger, not Charger.

Split Example

Assume you set the Configuration Effective Date parameter to Current Date.

Assume you create sales order 79639.

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
1	DVD	Not Configured	3	June 12	June 8	June 8
2	Movie Basket	Kit	10	June 12	June 12	June 12
-	Popcorn	Included Item	10	June 12	June 12	June 12

Line Number	Item	Item Type	Ordered Quantity	Ordered Date	Creation Date of Fulfillment Line	Configuration Effective Date
-	**Chocolates**	Included Item	10	June 12	June 12	June 12

Assume:

- You add line 1 on June 8. It includes the DVD item, which is a standard item.
- You use the Configuration Models work area on June 10 to change the model. You replace Ice Cream with Chocolates.
- You add line 2 on June 12. It includes the Movie Basket item, which is a kit that includes two items. The sales order uses the model that exists as of the Current Date, which is June 12. The current date is the date you revise or submit the sales order. You updated the model on June 10, so the order includes Chocolates.

Split the Line

Assume you use the Configuration Models work area on June 15 to change the model. You replace Chocolates with Big Drink.

On June 20, you revise the sales order. You split line 2 into two lines, lines 2.1 and 2.2.

Here's what happens.

Line Number	Item	Item Type	Ordered Quantity	Current Date	Creation Date of Fulfillment Line	Parent Creation Date	Configuration Effective Date
1	DVD	Not Configured	3	June 20	June 8	June 8	June 20
2.1	Movie Basket	Kit	6	June 20	June 12	June 12	June 20
-	Popcorn	Included Item	6	June 20	June 12	June 12	June 20
-	**Big Drink**	Included Item	6	June 20	June 12	June 12	**June 20**
2.2	Movie Basket	Kit	4	June 20	June 20	June 12	June 20
-	Popcorn	Included Item	4	June 20	June 20	June 12	June 20
-	**Big Drink**	Included Item	4	June 20	June 20	June 12	**June 20**

The order uses the Configuration Effective Date parameter to determine which model to use. In this example, you set Configuration Effective Date to Current Date. The current date is the date that you revise or submit the order, which is June 20, so the order uses the model that exists as of June 20 to identify the items it must include. On June 15, you updated the model to use Big Drink instead of Chocolates, so the order includes Big Drink on lines 2.1 and 2.2.

Split the Line Again

Assume you use the Configuration Models work area on June 24 to change the model. You replace Big Drink with Little Drink.

On June 25, you revise the sales order again. You split line 2.1 into two lines, lines 2.2 and 2.3.

Here's what happens.

Line Number	Item	Item Type	Ordered Quantity	Current Date	Creation Date of Fulfillment Line	Parent Creation Date	Configuration Effective Date
1	DVD	Not Configured	3	June 20	June 8	June 8	June 20
2.1	Movie Basket	Kit	6	June 20	June 12	June 12	June 20
-	Popcorn	Included Item	6	June 20	June 12	June 12	June 20
-	**Big Drink**	Included Item	6	June 20	June 12	June 12	**June 20**
2.2	Movie Basket	Kit	4	June 20	June 20	June 12	June 20
-	Popcorn	Included Item	4	June 20	June 20	June 12	June 20
-	**Big Drink**	Included Item	4	June 20	June 20	June 12	**June 20**

The order uses the Configuration Effective Date parameter to determine which model to use. In this example, you set Configuration Effective Date to Current Date. The current date is the date that you revise or submit the order, which is June 20, so the order uses the model that exists as of June 20 to identify the items it must include. On June 15, you updated the model to use Big Drink instead of Chocolates, so the order includes Big Drink on lines 2.1 and 2.2.

Use Current Date and Parent Creation Date

Assume you:

- Set the Configuration Effective Date to Current Date.
- Set the Configuration Effective Date for Exploding Included Items to Parent Creation Date.
- Use the Configuration Models work area on June 10 to change the model. You replace Ice Cream with Chocolates.

Here's what happens to sales order 79639 when you split line 2.

Line Number	Item	Item Type	Ordered Quantity	Current Date	Creation Date of Fulfillment Line	Parent Creation Date	Configuration Effective Date
1	DVD	Not Configured	3	June 20	June 8	June 8	June 12
2.1	Movie Basket	Kit	6	June 20	June 12	June 12	June 12

Line Number	Item	Item Type	Ordered Quantity	Current Date	Creation Date of Fulfillment Line	Parent Creation Date	Configuration Effective Date
-	Popcorn	Included Item	6	June 20	June 12	June 12	June 12
-	**Chocolates**	Included Item	6	June 20	June 12	**June 12**	June 12
2.2	Movie Basket	Kit	4	June 20	June 20	June 12	June 12
-	Popcorn	Included Item	4	June 20	June 20	June 12	June 12
-	**Chocolates**	Included Item	4	June 20	June 20	**June 12**	June 12

You set Configuration Effective Date to Current Date. The current date is June 20. Its the date you revise or submit the order. So the order uses the model that exists as of June 20 to identify the items it must include. On June 10, you updated the model to use Chocolates instead of Ice Cream, so the order includes Chocolates on lines 2.1 and 2.2.

Split the Line Again

Assume you use the Configuration Models work area on June 24 to change the model. You replace Ice Cream with Chocolates.

On June 25, you revise the sales order again. You split line 2.1 into two lines, lines 2.2 and 2.2.

Here's what happens to sales order 79639 when you split line 2.

Line Number	Item	Item Type	Ordered Quantity	Current Date	Creation Date of Fulfillment Line	Parent Creation Date	Configuration Effective Date
1	DVD	Not Configured	3	June 25	June 8	June 8	June 12
2.1	Movie Basket	Kit	6	June 25	June 12	June 12	June 12
-	**Chocolates**	Included Item	6	June 25	June 12	June 12	June 12
2.2	Movie Basket	Kit	4	June 25	June 20	June 12	June 12
-	**Chocolates**	Included Item	4	June 25	June 20	June 12	June 12
2.3	Movie Basket	Kit	4	June 25	June 25	June 12	June 12
-	**Chocolates**	Included Item	4	June 25	June 25	June 12	June 12

You set Configuration Effective Date to Current Date. The current date is June 25, so the order uses the model that exists as of June 25 to identify the items it must include. On June 24, you updated the model to use Chocolates instead of Ice Cream, so the order includes Chocolates on lines 2.1, 2.2, and 2.3.

Related Topics

- Overview of Configure-to-Order

10 Extend

Extensions

Overview

Overview of Creating Order Management Extensions

Write your own Groovy script that modifies your Order Management deployment, implements your own functionality, and specifies the extension point that determines when to run this script.

An order management extension is a Groovy script you write that meets your specific business requirements. An extension point is an event that you specify to determine when to run the script.

You can use a flexfield to modify the data model, or Page Composer to modify the user interface, but you can't use them to modify logic or modify your Order Management deployment. Instead, you can create an order management extension.

Here is some Groovy script in an example extension named Update Order Submit Date. It uses attribute _H1AttributeDateTime so it can set the value of an extensible flexfield to the date and time when Order Management submitted the sales order.

Business Requirements That You Can Meet with an Order Management Extension

Example Requirement	Description
Get or update data	Create an order management extension to get or update data from different sources. • Use the getAttr method to get data from Order Management. • Use a public view object (PVO) to get data from an Oracle application. • Use a web service to get data from a source that resides outside of Order Management. You can also use a transformation rule to get or update data, but an extension allows you to use more advanced code. You can also use it to add new order line.

Example Requirement	Description
Set the default value for sales order attributes or extensible flexfields	You can use a pretransformation rule, posttransformation rule, or do some simple personalization in a user's sandbox, but you might find that using Groovy to write your own logic more closely meets your need to set default values before you save the sales order or submit it to order fulfillment. You can use an extension to: • Create simple or complex scripts. • Set default values from the order header to order line. • Set default values from the original sales order to a return order. • Get default values from other Oracle applications or work areas, such as Oracle Procurement or Product Information Management. • Use a SOAP service to make a web service call to an application that sits outside of Oracle.
Validate data	Validate data on different attributes on the same entity, such as an order header, or on attributes on different entities, such as on the order header and the header's order lines. Compare data between attributes on an order header or order line to data from attributes in other applications or workareas, such as entities in Oracle Pricing or attributes on item's from Product Information Management.
Get values for return lines from the original sales order.	A return order doesn't include values from the original order on some order line attributes. For example, the return line doesn't include the original value for the sales credit or purchase order number. You can use an extension to get the value from the original line order.
Apply a hold	You can manually apply a hold in the Order Management work area or through a web service, but you might find it's easier to do it automatically through an extension: • Automatically apply a hold according to a condition or attribute value • Apply a hold with the On Save or the Start of Submission extension point.
Display error messages and warnings	Create an extension that displays an error message or warning.
Implement your own logic	Use Groovy to write your own logic when you code your extension. Your code can validate the sales order. If validation fails, then your code can stop the flow and create a message. For details about Groovy, see *Groovy Programming Language*.

Example Actions

Action	Description
Submit a sales order	• Reassign the warehouse for pick-to-order. Set up an item that determines whether the sales order references a pick-to-order or a hardware model. • Stop support lines that reference an incorrect accounting rule. Set up item entities and accounting rule entities. • Make sure only one contract is Active or Signed for a booking. Set up entities for the installed base, contracts, or Order Management. • Assign a sales representative. Set up sales compensation rules. • Update the accounting rule for a model for included items.

Action	Description
	• Update the quote status for each sales order from a store front that a third-party partner maintains. • Verify the purchase order number that the Order Entry Specialist enters on the sales order equals a purchase order in Oracle Procurement. • Assign the Salesperson according to a sales compensation rule that you create in some application outside of Oracle Applications, then use a web service to call the application. • During order import, copy attachments from a purchase-to-order kit to each item that the kit contains. • Set the default value for order preferences in each sales order. Get these values from the customer master.
Save or validate a sales order	Write a validation on an imported source order that rejects the import or prevents the Order Entry Specialist from saving the order without fixing the problem.

Parts of an Order Management Extension

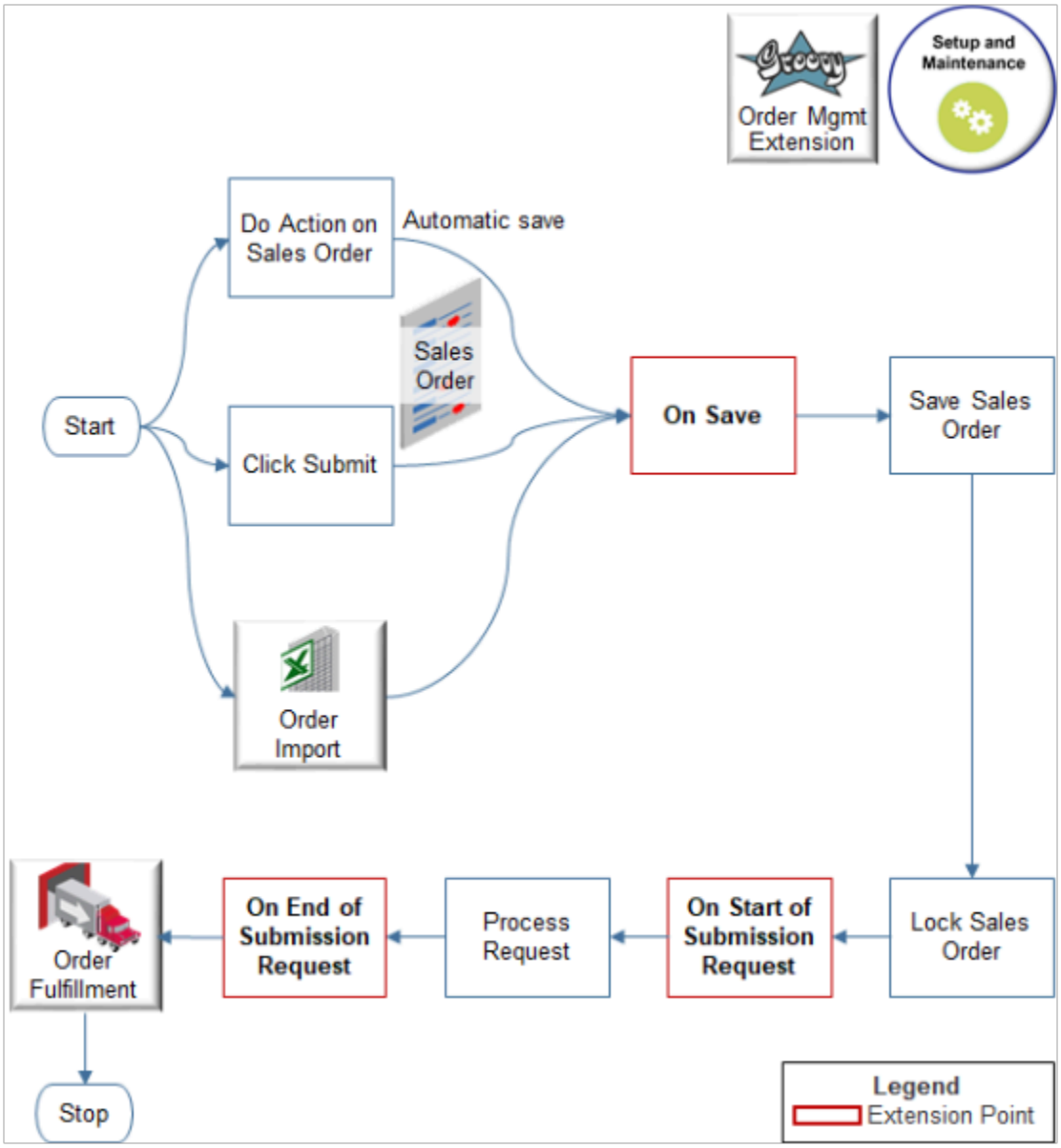
Part	Description
Function	A Groovy function, similar to a Java function, such as getAttr. Use a function only in the extension where you set it up. You can't define a function in one extension, and then use it in another extension.
Attribute	The attribute of a sales order, such as ShippingInstructions.
Web Service	The name of a web service. • Your extension code can only call a web service that you set up on the Manage Web Service Definitions page. • You must manually create the request payload in your code. • Your code must parse and interpret the response payload. • You must write your code so it works only for a synchronous interaction. You can't write code for an asynchronous interaction.
Message	A message you send to Order Management and that can display in the Order Management work area. • An extension can log an error or warning message in the messaging framework. • Order Management logs each message in a log file as part of order import or when a web service rejects a sales order. • An error message stops the flow and rejects the save or submit action. • A warning allows the flow to continue. If you use a warning at the beginning of a submission request, then Order Management displays the warning message to the Order Entry Specialist, then continues to process the sales order in order fulfillment. • Use the Manage Messages page to define messages and tokens. • Use tokens to insert dynamic content into each message. • Modify the DOO_MSG_REQUEST_FUNCTION lookup to categorize the errors that Order Management displays to the Order Entry Specialist and improve search for these errors.

Related Topics

- Points Where You Can Run Order Management Extensions
- Use Proven Coding Techniques
- Call Web Services from Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Points Where You Can Run Order Management Extensions

Run an order management extension at different extension points when creating a sales order or during order fulfillment.



Note

1. Something happens that triggers the On Save extension point.
    - An Order Entry Specialist does an action in the Order Management work area that causes Order Management to automatically save the sales order. Here are some example actions.
        - Validate
        - Save
        - Save and Close
        - Reprice
        - Submit
        - Copy Order
        - Create Revision
        - Create Return
    - Order Management imports a source order from a source system, then processes it, such as creating a cross-reference, defaulting a value, and so on.
2. You can create an extension that runs on the On Save extension point.
3. You can create an extension that runs on the On Start of Submission Request extension point. If the extension results in failure, then Order Management sets the sales order status to Draft, with errors.
4. Order Management processes the submission request, including running validations, doing credit check, communicating with Global Trade Management, order approvals, and so on.
5. You can create an extension that runs on the On End of Submission Request extension point. If the extension results in:
    - **Failure**. Order Management sets the sales order status to Draft, with errors.
    - **Warning**. Order Management displays a warning message, successfully submits the sales order, and continues to process it through order fulfillment. The Order Entry Specialist can examine the warning messages on the submitted order.
6. The fulfillment system fulfills the sales order, including shipping, invoicing, and so on. You can't run an extension after Order Management submits the sales order to order fulfillment and order orchestration.

Extension Point Behavior

On Save

If you run an order management extension during On Save and it creates a validation failure, then Order Management stops running the extension and moves control to the next extension that you specify to run during On Save. If the validation failure includes at least one error message, and if Order Management has already run each of the On Save extensions, then it aborts the save for the sales order.

Order Management handles each message that the extension uses differently depending on how it's saving the sales order.

- An Order Entry Specialist is saving a sales order in the Order Management work area. Order Management displays the message in a dialog. This user must correct the validation error and attempt to save the sales order again.
- A service is saving a source order during order import. Order Management returns the message in a response. The channel that's importing the source order must fix it, then import it again.

Order Management doesn't save the message in the messaging framework because it might not have saved the sales order and the change that caused the validation to fail.

On Start of Submission Request

- Order Management calls the extension when the extension point happens. It's the first operation that Order Management does when it validates the submit.
- An extension can modify values on each writable order attribute, including header attributes and order line attributes, flexfield attributes, and attachment attributes.
- Order Management runs the extension before it validates the submit or applies constraints.
- If an extension encounters a validation failure, then Order Management stops running the extension and moves control to the next extension that you specify to run during On Start of Submission Request.
- If the validation failure includes at least one error message, then Order Management aborts the submit for the sales order, returns the order status to Draft, then logs the error and warning messages.
- If the Order Entry Specialist is saving a sales order in the Order Management work area, and if the validation failure includes at least one error message, then Order Management displays a dialog that includes the message.
- If all messages are warnings, then Order Management submits the sales order to order fulfillment and doesn't display a dialog in the Order Management work area. The user can view the warning message after Order Management submits the sales order to order fulfillment.

On End of Submission Request

- Order Management calls the extension when the extension point happens. It makes this call as the last step before it sends the sales order to order fulfillment.
- The extension can modify values only on header flexfields and fulfillment line flexfields. If the extension attempts to modify any other attribute, then Order Management logs a runtime error and aborts all extensions that it's running for this extension point.

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Call Web Services from Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Guidelines

Determine Data Requirements and Consider Runtime Behavior

Determine the type of data you need and how you will access it. Then determine the runtime behavior your need.

Determine Your Data Requirements

Determine the type of data you need and how you will access it.

Type of Data You Need	How to Access Data	Limitation
Data from a sales order in Order Management.	To read data, use methodgetAttribute. To write data, use the setAttribute method.	Each method can read or write only to the sales order that the Order Entry Specialist is currently entering in work area Order Management. They don't read or write to some other version of the current sales order, or to some other sales order in Order Management. Use a public view object to access data from a sales order that isn't the current sales order. Each method can only read from or write to an attribute. They can't create an order line.
Data from an Oracle Application outside of Order Management, such as the customer master, the item master in Oracle Procurement or Oracle Receivables, and so on. Data from a sales order in Order Management that isn't the current sales order.	Public View Object (PVO). For details, see *Use Extensions to Get Data from Oracle Applications*.	The Oracle Application must share its data through a public view object. The public view object you need might not be available, or it might not provide access to the attribute that you require. It might be necessary to submit a request to get access to the data you need.
Data from a system that resides outside of an Oracle application	Web service.	You can use only a synchronous web service. You can't use an asynchronous web service. You can use only a SOAP service. You can't use REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.

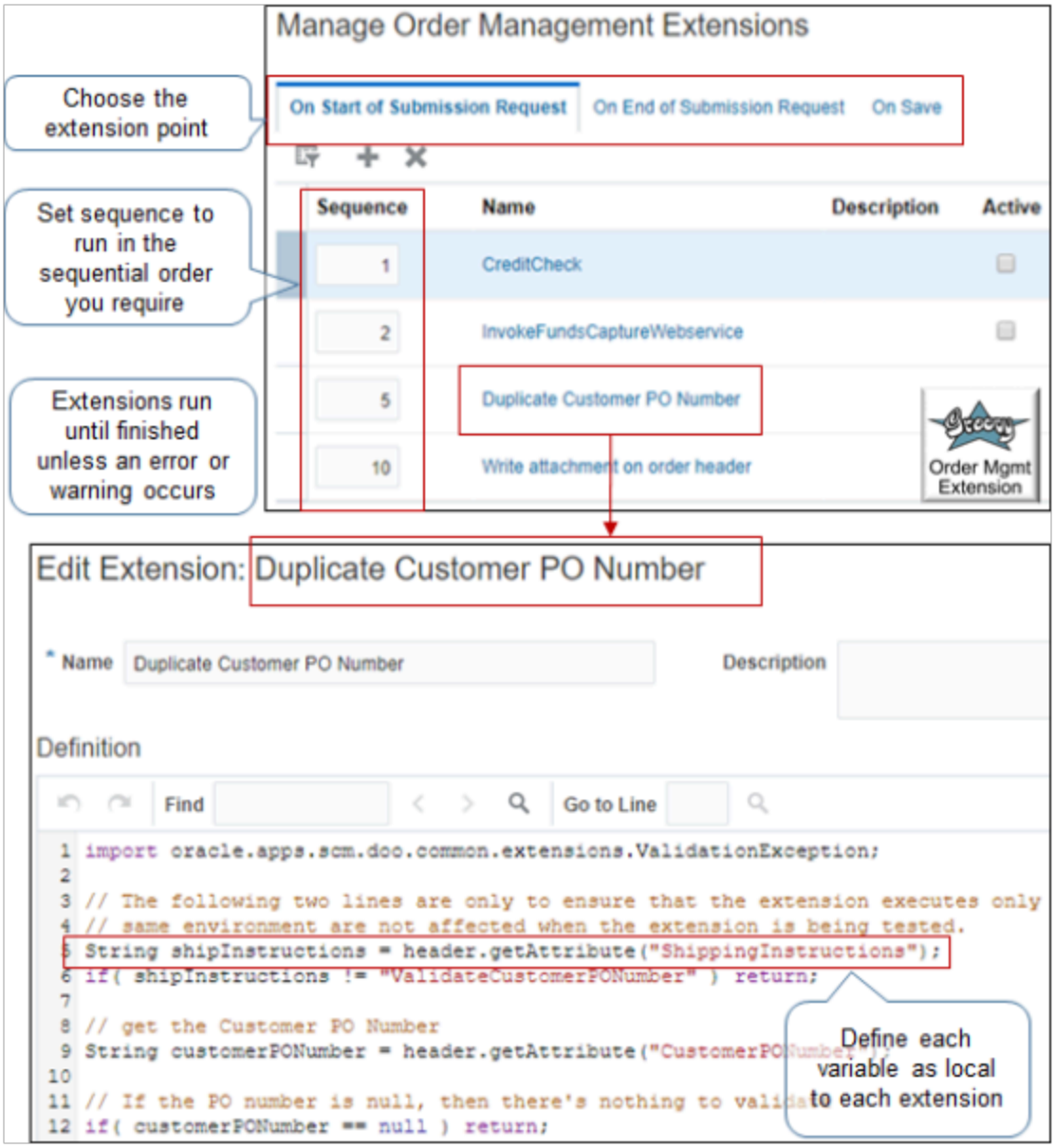
Consider the data that you can access.

Read Access	Write Access
Do a read operation on this data at any of the extension points. • Order header, including extensible flexfield • Fulfillment lines, including extensible flexfield • Sales credit • Pricing entity, such as charge, charge component, manual price adjustment, total, price validation, tax detail, and so on • Lot or serial • Transactional attribute • Payment • Billing plan	Do a write operation on this data during the On Save or the On Start of Submission Request extension points. • Order header, including extensible flexfield • Fulfillment line, including extensible flexfield • Attachment • Sales credit • Transactional attribute • Document reference You can't do a write operation on this data during the On End of Submission Request extension point because Order Management already performed a large set of predefined validations at this point. Perform a write operation on data during the On End of Submission Request extension point. • Extensible flexfield on the order header • Extensible flexfield on the fulfillment line

Read Access	Write Access
• Attachment • Fulfillment line detail • Document reference	• Attachment

For a complete list, see *Entities That You Can Use With Order Management Extensions*.

Consider Runtime Behavior



Note

- Order Management runs each extension in sequential order, and it only extensions where the Active attribute contains a check mark run. Assume you create extensions x, y, and z. Extension x runs first, then extension y. If you make z active, then Order Management runs extension x, and then z, and then y.

Sequence	Extension Name	Active
1	x	Contains a check mark
10	y	Contains a check mark
5	z	Doesn't contain a check mark

- Order Management runs each active extension until it finishes unless an error or warning happens. If an error or warning happens, then Order Management doesn't run any more extensions. For example, assume x, y, and z are active, x finishes, but z ends in error. Order Management won't run extension y.
- Each variable is local to each extension. You can't share them across extensions. For example, assume you require variable A in extensions x and y. If you define variable A in x, then you can't use A in y. Instead, you must write logic in y that references A in x.

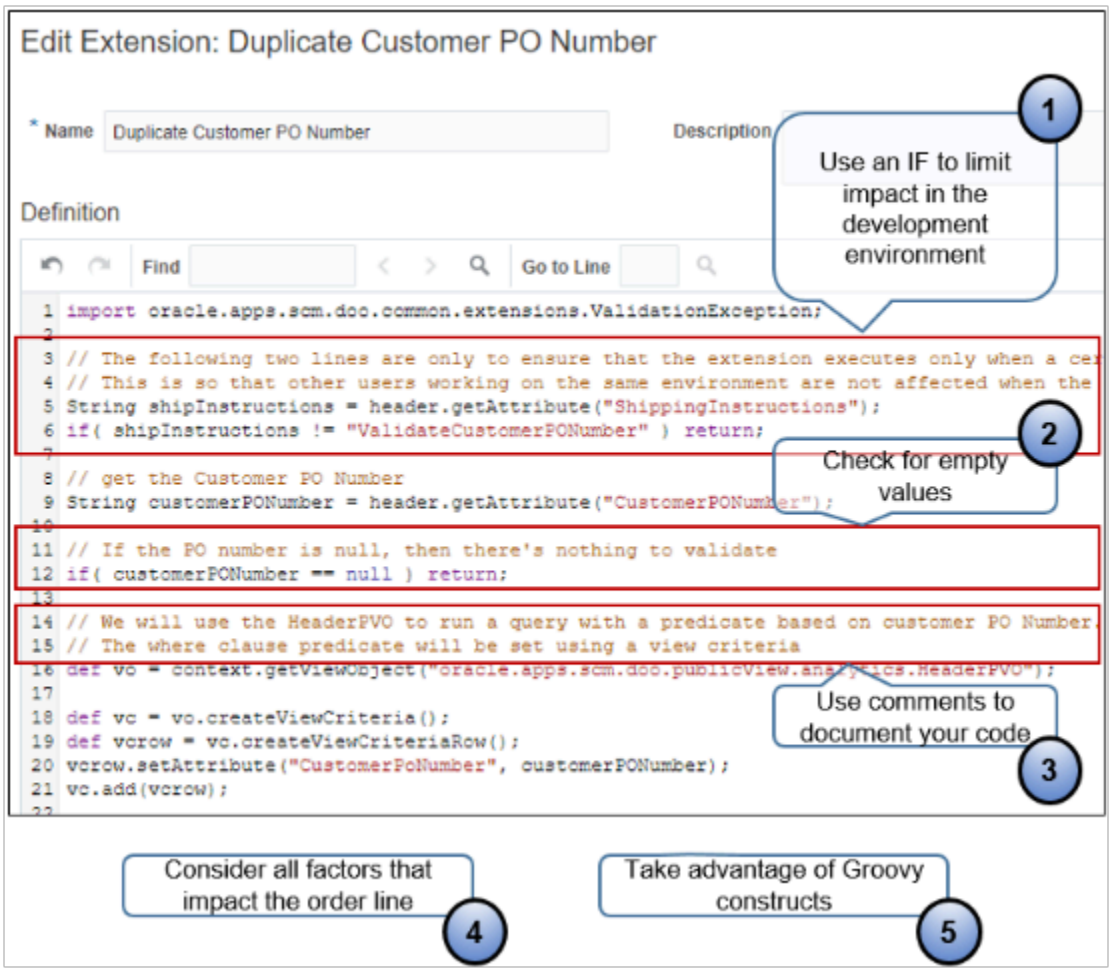
Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Use Proven Coding Techniques

Use these guidelines to help you create an order management extension.

Overview



Note

1. **Use an IF**. Avoid writing code that runs in every condition in your development environment. Instead, use an IF statement so your code runs only in your specific use case.

   Assume you write a statement that runs every time the extension point happens, without conditional logic, but the extension contains an error. The extension might cause every sales order in your development environment to fail in every situation, regardless of the condition you must test. This situation might negatively affect other developers, and also make it more difficult for you to manage each of your own extensions.

   For example, assume you write an extension that manipulates the shipInstructions attribute. You can add this statement as the first statement in your extension so your code runs only for your use case.

   ```
   if (shipInstructions != "ValidateCustomerPONumber" ) return;
   ```

   If the statement evaluates to true, then no more logic in your extension runs. Instead, the flow exits your extension.

   Remove the IF after you test your extension and are ready to deploy into production.
2. **Check for empty values**. Avoid a null pointer exception. Use an IF statement that makes sure each attribute you reference contains a value. If it doesn't, then return flow out of the extension.

3. **Add comments**. Add detailed comments that document each part of your code so you know exactly the logic that each statement implements. Comments help others to understand the intent of your code, and also help you to troubleshoot large complex code, or code that you haven't examined for a long period of time.

Notes

- If your extension does an action only on a new sales order or only on a revision, then add a condition that determines whether the sales order you're modifying is new or is a revision right at the beginning of your extension. This way, if the condition evaluates to false, you can exit the extension immediately without having to evaluate the rest of the extension code.

  Consider all factors that impact the order line, such as configured items, kits, or ship sets.
- Take advantage of Groovy constructs, such as simplified array usage and initializing maps.

Define Variable Type in Groovy

Groovy is a dynamic language and doesn't require you to define the type for each variable. For example, you can define soldToPartyName and Lines each as a variable without a type. As an option, specify a type, if necessary. For example, define soldToPartyName as type String.

Access each child entity as an attribute. For example:

```
header.getAttribute("Lines"), header.getAttribute("SalesCredits")
```

Use Implicit Variables

Use implicit variables with each extension. You don't need to define them.

Implicit Variable	Description
Header	Represents the order header object. Use the Header variable to access order header attributes and child entity rows.

Context	Context object that allows each extension to access context details, such as the name of the extension that's running, the event code, and so on. The context object also provides access to utility and helper methods, such as logging.
	
	You use this format.
	
	`context extension name`
	For example:
	
	`context Update Order Submit Date`

Reference Sales Order Entities

Use the implicit header variable to access each order entity. Order Management exposes each entity as a generic row object. Each extension can reference only get methods or set methods.

Action	Code
Read an attribute value.	getAttribute("**attributeName**")
Write to an attribute.	setAttribute("**attributeName**", **attributeValue**)

Action	Code

For example:

Code	Description
`def soldToPartyName = header.getAttribute("BuyingPartyN`	Get the value of the BuyingPartyName attribute.
`header.setAttribute("ShippingInst "Use only next-day air.");`	Set the value of the ShippingInstructions attribute to **Use only next-day air**.
`def lines = header.getAttribute("Lines");`	Return a set of lines. You can iterate over the iterator to access individual lines.

Loop Through Lines

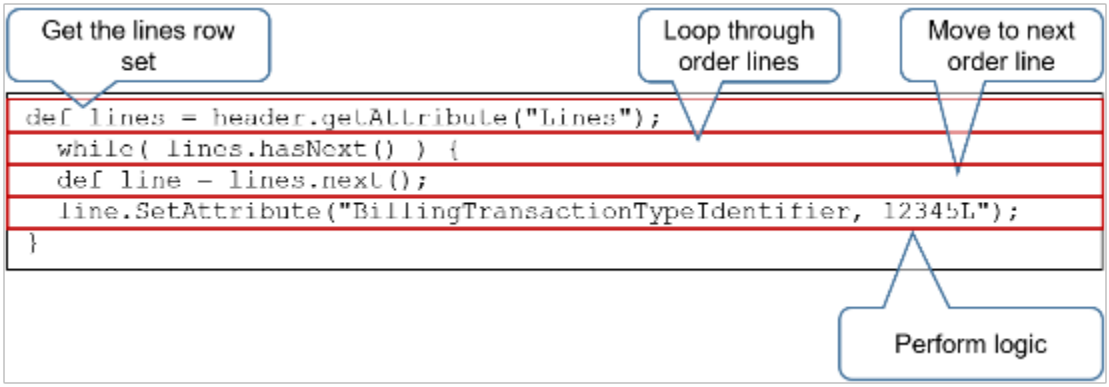
You might need to update a number of attributes on all the order lines in a sales order.

Assume you need to update the Shipping Method, Packing Instructions, and Payment Terms on all 50 order lines in sales order 54638. It's possible to create 50 extensions to update each line separately, but we recommend that you create a single extension to update all of the attributes on all lines, and then loop through the lines to update the attributes.

This approach avoids having to create 50 instances of the extension, doing 50 different line loop checks, and that improves performance.

Navigate Row Sets

Order Management returns each child entity as a set of rows. For example, an order line is a child of a sales order. This example illustrates how you can use `while next` to loop through each row, an order line, from the set, which is the sales order.



Here's the same code with descriptions.

Code	Description
`def lines = header.getAttribute("Lines");`	Call the getAttribute method on the order header to access the row set iterator for the order lines.
`while( lines.hasNext() ) {`	Determine whether the lines row set includes a next row, and loop until the row set doesn't contain any more rows.
`def line = lines.next();`	Get the next order line row. • On the first loop through the **`while`**, move from line 1 to line 2. • On the second loop through the **`while`**, move from line 2 to line 3. • And so on.
`line.SetAttribute("BillingTransac 12345L");`	Set the value of the billing transaction type on the order line to 12345L. For brevity, this example hard codes the value 12345L. In most situations, its more likely you would define a variable or some other logic rather than hard code the value.

Here's the same code without comments.

```
while( lines.hasNext() ) {
def line = lines.next();
line.SetAttribute("BillingTransactionTypeIdentifier, 12345L");
}
```

Navigate Row Sets of Grandchild Entities

Access a grandchild entity.

Here's an example extension that:

- Accesses the Charge Component row of an order line
- Gets the order line
- Accesses the order charge rows below the order line
- Accesses the charge component rows from the order charge

Code	Description
`def lines = header.getAttribute("Lines");`	Use method getAttribute on the order header to access the row set iterator for the order lines.
`while( lines.hasNext() ) {`	Determine whether the lines row set includes a next row, and loop until the row set doesn't contain any more rows.
`def line = lines.next();`	Get the next row.
`def charges = line.getAttribute("OrderCharges")`	Get the row set for the order charges on the order line.

Code	Description
`while( charges.hasNext() ) {`	Loop through the order charge rows.
`def charge = charges.next();`	Get the next order charge from the row set.
`def chargeComps = charge.getAttribute("ChargeCompor`	Get the row set for the charge components of the child entity.
`while( chargeCompo.hasNext() ) {`	Loop through the charge components for the current charge.
`def chargeComp = chargeCompo.next();`	Get the next charge component.
def currency = chargeComp.getAttribute("ChargeCurrency	Get the value of attribute ChargeCurrencyCode from the current charge component row.

Here's the same code without comments.

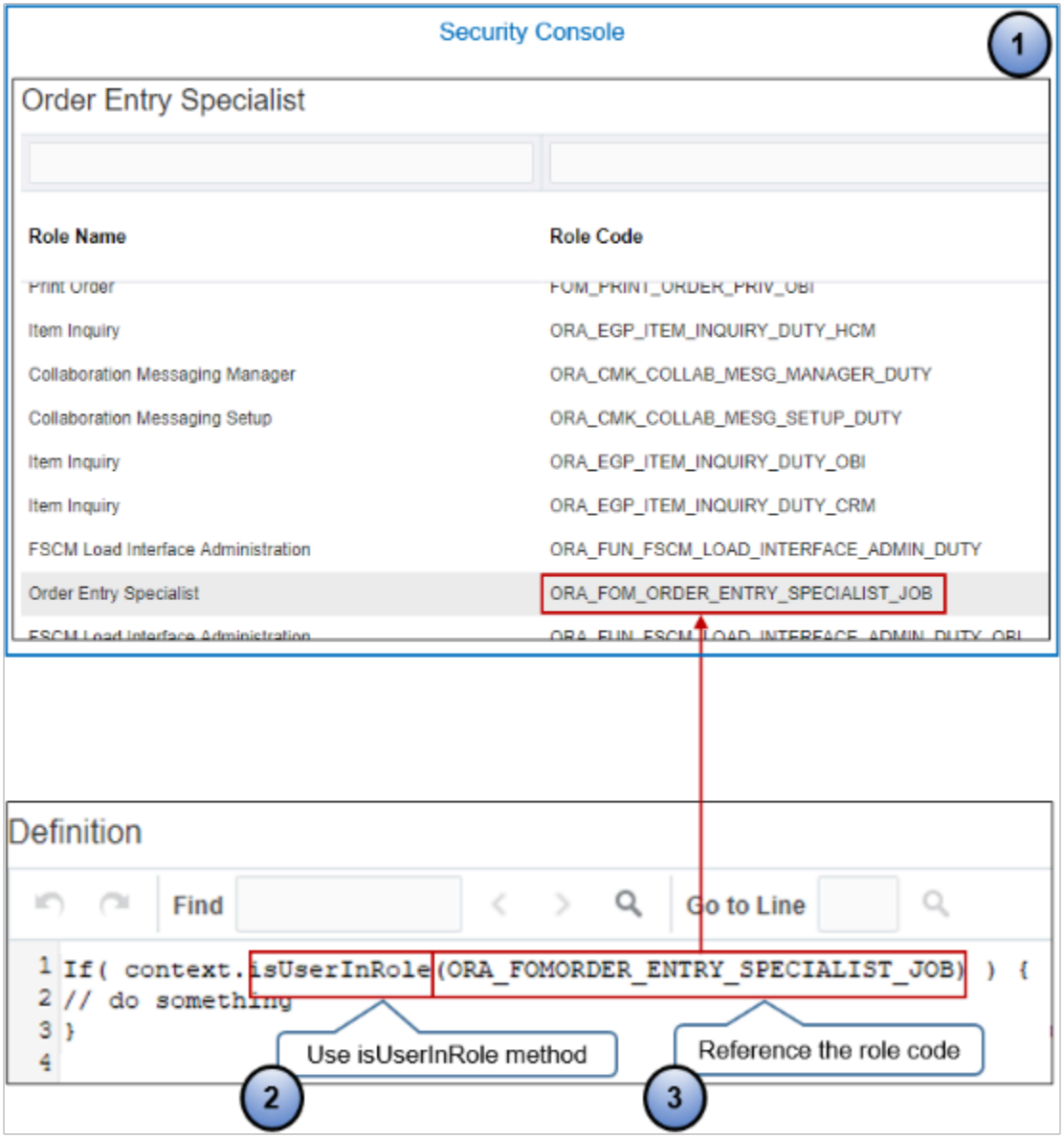
```
def lines = header.getAttribute("Lines"); //get the lines row set
while( lines.hasNext() ) {//determine whether more lines exist
 def line = lines.next();
 def charges = line.getAttribute("OrderCharges");
 while( charges.hasNext() ) {
 def charge = charges.next();
 def chargeComps = charge.getAttribute("ChargeComponents");
 while( chargeCompo.hasNext() ) {
 def chargeComp = chargeCompo.next();
 def currency = chargeComp.getAttribute("ChargeCurrencyCode");
 }
 }
}
```

Secure Your Extension

Write logic according to the job role. For example, write an extension that prevents an action or that sets default values on the sales order according to the job role that the Order Entry Specialist uses when signing into Order Management. This example references ORA_FOM_ORDER_ENTRY_SPECIALIST_JOB, which is the code for the Order Entry Specialist job role.

This topic uses predefined job roles. You must create your own job roles, depending on your security requirements.

For details about how to set up privileges, see *Security Reference for Order Management*.



Note

1. Use the Security Console to get details about the job roles and role code that your extension must reference. For details, see Overview of Security Console in Oracle ERP Cloud, Securing ERP.
2. Use the isUserInRole method.
3. In the first parameter for isUserInRole, reference the role code that the security console defines.

Don't Use Extensions and Transformation Rules Together

Don't use an order management extension and a product transformation rule to add order lines to the same sales order because the rule might remove or modify the lines that the extension added.

For example, if you use an extension to add Line 1 to sales order 65748, and then use a rule to add line 2 to the same sales order, then the rule will remove line 1 and add line 2 when you save, submit, or revise the order.

For another example, if you use an extension to add the AS54888 item with a quantity of 2 to line 1 on sales order 65748, and then use a rule to add the AS29888 with a quantity of 10 on line 1, then the rule will remove the AS54888 and the quantity before it adds the AS29888 with a quantity of 10.

Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Test Your Order Management Extension

Test your order management extension.

Validate Your Code

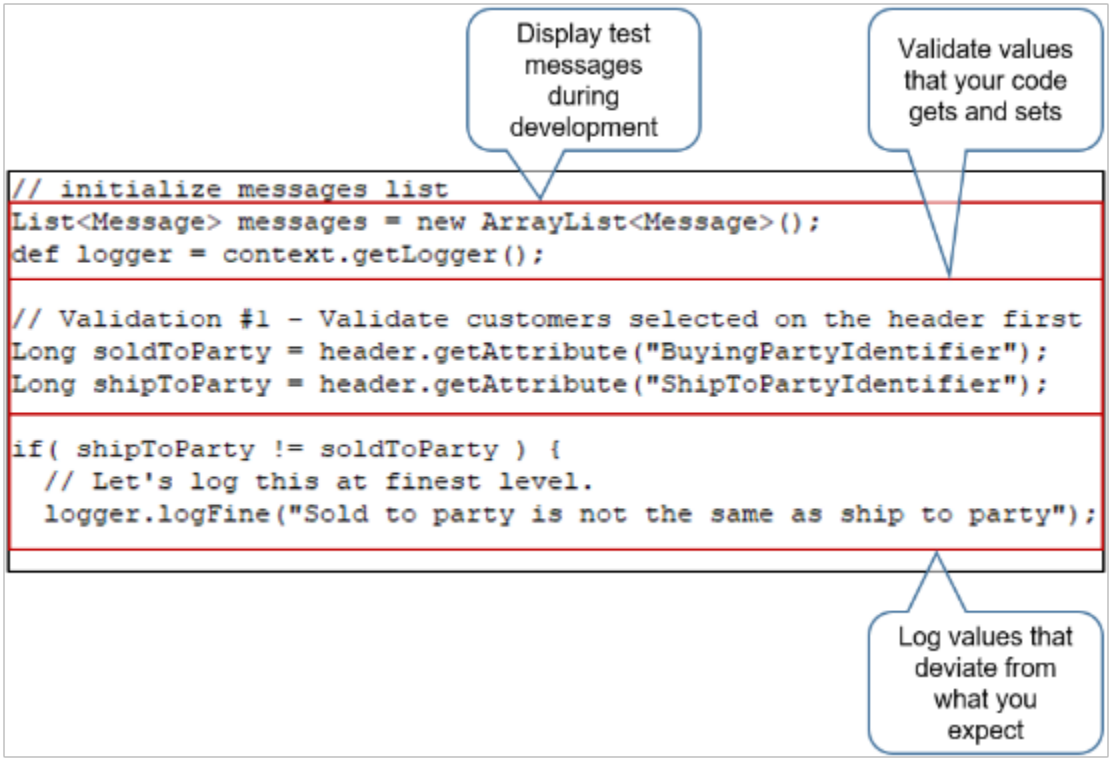
Enter your code into the Definition area of the Edit Extensions page, then click Validate to validate your design-time code. Order Management will make sure you correctly formatted your code, such as correct usage of parentheses, semicolons, and so on.

If an error happens at runtime, then Order Management handles the error in the same way it handles any other error. It displays the cause of failure, extension name, and event name. It displays these details in the Order Management work area and in the log files. It can handle these runtime errors.

- Reference to an attribute or method that isn't valid
- Incorrect reference between the message name and a token
- Incorrect reference to a web service name

Use Logging to Test Your Code

Include debugging tests during development to test your code and verify that it runs as expected.



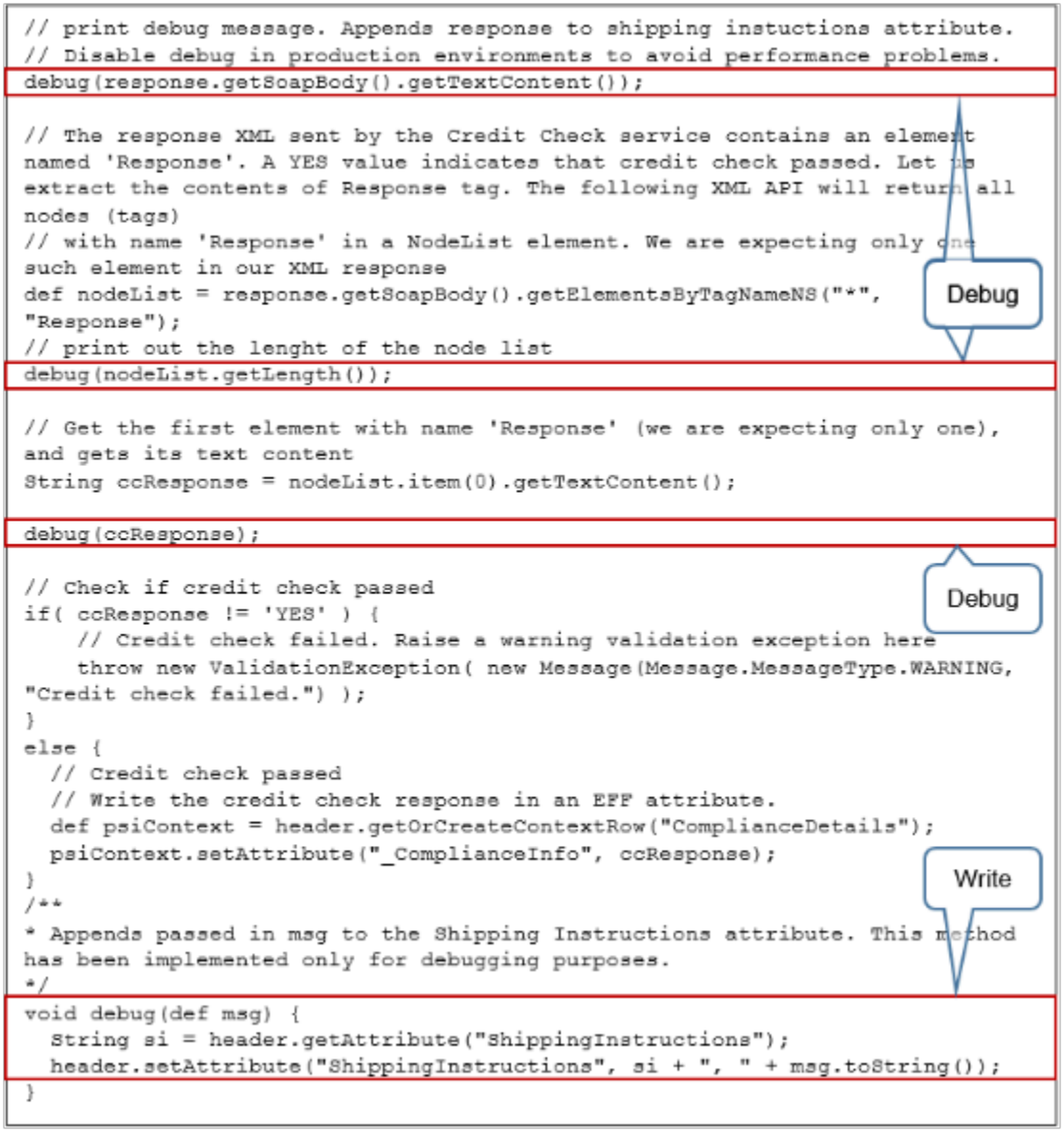
Note

- After you finish testing and are ready to deploy your extension to production, modify your debugging logic to write to a log file instead of commenting your test code. Writing to the log file can be helpful for future possible troubleshooting. If you write data to the log file in a production environment, then you must contact Oracle Support or a customer administrator to get view access to the log files that include application diagnostic data.
- Use the log files to evaluate performance. For example, if you write a lot of extension code, then examine and monitor performance to make sure your code doesn't impact performance in a negative way.

  For example, extension code typically performs a significant amount of validation immediately after the Order Entry Specialist clicks Submit. If Order Management requires a long amount of time to finish the submit, such as five minutes, then it might be necessary to look closely at your code to determine whether it contains some logic you can streamline.

Use the Debug Method

Use the Debug method at various points in your code during development to write variables and other values to an attribute that you can examine in work area Order Management. This example calls `debug` three times, then writes the contents of debug into the ShippingInstructions attribute.



You can then write your code so it displays these contents in the Shipping Instructions attribute in the Order Management work area, then examine this display output to determine the values of various attributes and variables that exist at various points in your code.

When you're ready to deploy to production, modify `void debug`. For example, assume you use this code to write your debug contents.

```
void debug(def msg)
 {String si = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", si + ", " + msg.toString());
```

Modify this code so it writes to the log. You replace `header.setAttribute` with `logger.logFine`. For example:

```
void debug(def msg)
 {String si = header.getAttribute("ShippingInstructions");
 logger.logFine("ShippingInstructions", si + ", " + msg.toString());
```

Use this approach so you can change only a single line in your code, which is useful if you have 10s of lines that call `debug`. You can also comment each line that calls the debug method, but commenting runs the risk of missing lines that call debug, or accidentally commenting a line that doesn't call debug but that's critical to your code logic.

Here's the complete code for this example.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");

if( poNumber != "CreditCheck" ) return;

// get attribute to populate in the payload
String customer = header.getAttribute("BillToCustomerName");
Long accountId = header.getAttribute("BillToCustomerIdentifier");
BigDecimal amount = new BigDecimal(1000);

// prepare the payload
String payLoad = "<ns1:creditChecking xmlns:ns1=\"http://xmlns.oracle.com/apps/financials/receivables/
creditManagement/creditChecking/creditCheckingService/types/\">" +
 "<ns1:request xmlns:ns2=\"http://xmlns.oracle.com/apps/financials/receivables/creditManagement/
creditChecking/creditCheckingService/\">" +
 "<ns2:CustomerName>" + customer + "</ns2:CustomerName>" +
 "<ns2:CustomerAccountNumber>" + accountId + "</ns2:CustomerAccountNumber>" +
 "<ns2:RequestType>Authorization</ns2:RequestType>" +
 "<ns2:PriceType>ONE_TIME</ns2:PriceType>" +
 "<ns2:RecurrencePeriod></ns2:RecurrencePeriod>" +
 "<ns2:RequestAuthorizationAmount currencyCode=\"USD\">" + amount + "</ns2:RequestAuthorizationAmount>" +
 "<ns2:RequestAuthorizationCurrency>USD</ns2:RequestAuthorizationCurrency>" +
 "<ns2:ExistingAuthorizationNumber></ns2:ExistingAuthorizationNumber>" +
 "<ns2:Requestor>ar_super_user</ns2:Requestor>" +
 "</ns1:request>" +
 "</ns1:creditChecking>";

// invoke the Check Check service using web service connector name 'CreditCheckService'. The connector is
 set up using task 'Manage External Interface Web Service Details'. Since this is an external service that
 is secured
// using message protection policy, we have registered the the https URL of the service
def response = context.invokeSoapService("CreditCheckService", payLoad);

// print a debug message. This appends the entire response to the shipping instuctions attribute.
// To avoid performance problems, you must comment out all debug statements in your production environment.
debug(response.getSoapBody().getTextContent());

// The response XML sent by the Credit Check service contains an element named 'Response'. A YES value
 indicates that credit check passed. Let us extract the contents of Response tag. The following XML API will
 return all nodes (tags)
// with name 'Response' in a NodeList element. We are expecting only one such element in our XML response
def nodeList = response.getSoapBody().getElementsByTagNameNS("*", "Response");

// print out the lenght of the node list
debug(nodeList.getLength());

// Get the first element with name 'Response' (we are expecting only one), and gets its text content
String ccResponse = nodeList.item(0).getTextContent();

debug(ccResponse);

// Check if credit check passed
if( ccResponse != 'YES' ) {
 // Credit check failed. Use a warning validation exception here.
 throw new ValidationException( new Message(Message.MessageType.WARNING, "Credit check failed.") );
}
else {
```

```
 // Credit check passed
 // Write the credit check response in an EFF attribute.
 def psiContext = header.getOrCreateContextRow("ComplianceDetails");
 psiContext.setAttribute("_ComplianceInfo", ccResponse);
}

/**
* Appends passed in msg to the Shipping Instructions attribute. This method has been implemented only for
 debugging purposes.
*/
void debug(def msg) {
 String si = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", si + ", " + msg.toString());
}
```

Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Examples

Extend Order Headers

Use these code examples to help you create order management extensions that manipulate data on the sales order header.

Many of these examples test a value for the purchase order number on the order header. This test isolates the extension and prevents it from affecting other developers who might also be running test code. For details, see *Use Proven Coding Techniques*.

To add and delete an attachment on an order header, see *Extend Attachments*.

Get Pricing Details for Order Header

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

if (!"PMC TEST".equals(header.getAttribute("CustomerPONumber"))) return;

List < Message > messages = new ArrayList < Message > ();

def payments = header.getAttribute("OrderTotals");
while (payments.hasNext()) {
 def payment = payments.next();
 messages.add(new Message(Message.MessageType.ERROR, "TotalAmount is " +
 payment.getAttribute("TotalAmount")));
 messages.add(new Message(Message.MessageType.ERROR, "OrderTotalId is " +
 payment.getAttribute("OrderTotalId")));
 messages.add(new Message(Message.MessageType.ERROR, "TotalCode is " + payment.getAttribute("TotalCode")));
 messages.add(new Message(Message.MessageType.ERROR, "CurrencyCode is " +
 payment.getAttribute("CurrencyCode")));
 messages.add(new Message(Message.MessageType.ERROR, "DisplayName is " +
 payment.getAttribute("DisplayName")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "TotalGroup is " +
 payment.getAttribute("TotalGroup")));
 messages.add(new Message(Message.MessageType.ERROR, "PrimaryFlag is " +
 payment.getAttribute("PrimaryFlag")));

 messages.add(new Message(Message.MessageType.ERROR, "EstimatedFlag is " +
 payment.getAttribute("EstimatedFlag")));
 messages.add(new Message(Message.MessageType.ERROR, "CreatedBy is " + payment.getAttribute("CreatedBy")));


 messages.add(new Message(Message.MessageType.ERROR, "HeaderId is " + payment.getAttribute("HeaderId")));
 messages.add(new Message(Message.MessageType.ERROR, "ObjectVersionNumber is " +
 payment.getAttribute("ObjectVersionNumber")));


}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Set the Billing Transaction Type According to Order Type

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String orderTypeCode = header.getAttribute("TransactionTypeCode");
def billingTxnTypeId = null;

//Message msg = new Message(Message.MessageType.WARNING, "Order Type: " + orderTypeCode);
//throw new ValidationException(msg);


//Case statement that tests the order type and calls function to get billing transaction type.

switch (orderTypeCode) {
 case ["AUTO RETURN - SHIP", "AUTO RETURN - SHIP", "AUTO RETURN - MXT SHIP", "Mosoi_test"]:
 billingTxnTypeId = getBillingTxnTypeId("MOTO_INVOICE");
 //msg = new Message(Message.MessageType.WARNING, "First Case " + billingTxnTypeId );
 break;
 case ["UnreferencedRMA"]:
 billingTxnTypeId = getBillingTxnTypeId("Invoice");
 //msg = new Message(Message.MessageType.WARNING, "Second Case " + billingTxnTypeId );
 break;
 default:
 billingTxnTypeId = null;
 //msg = new Message(Message.MessageType.WARNING, "Default " + billingTxnTypeId );
 break;
}

//throw new ValidationException(msg);

//update all order lines with the Billing Transaction Type.

def lines = header.getAttribute("Lines"); //get the lines row set
while (lines.hasNext()) { //if there are more order lines
 def line = lines.next();
 line.setAttribute("BillingTransactionTypeIdentifier", billingTxnTypeId);
}


//Function to get Billing Transaction Type

Long getBillingTxnTypeId(String billingTxnTypeName) {

 def txnTypePVO = context.getViewObject("oracle.apps.financials.receivables.publicView.TransactionTypePVO");
```

```
 //Create view criteria (where clause predicates)
 def vc = txnTypePVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();

 //Only return Billing Transaction Type for the common set that you must change.

 vcrow.setAttribute("Name", billingTxnTypeName);
 vcrow.setAttribute("SetName", "Common Set");

 //Run the view object query to find a matching row.
 def rowset = txnTypePVO.findByViewCriteriaWithBindVars(vc, 1, new String[0], new Object[0]);

 //See if we have a matching row.
 def row = rowset.first();

 Long txnTypeId = (Long) row.getAttribute("CustTrxTypeSeqId");

 return txnTypeId;
}
```

Set Payment Term on Order Header

Set the default value to use for the payment term on the order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

List < Message > messages = new ArrayList < Message > ();
def logger = context.getLogger();
def HeaderPayTerm = header.getAttribute("PaymentTerm");
def PName = header.getAttribute("BillToCustomerName");
def AId = header.getAttribute("BillToCustomerIdentifier");


def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def linePayTerm = line.getAttribute("PaymentTerm");
 def linetransactioncode = line.getAttribute("TransactionCategoryCode")
 def partyId = getBillToPartyId(PName);
 def termId = gettermID(AId, partyId);
 def termName = gettermName(termId);

 if (HeaderPayTerm == null && linePayTerm == null && linetransactioncode != 'RETURN') {
 if (termName != null)
 header.setAttribute("PaymentTerm", termName.getAttribute("Name"));
 }
}

Object getBillToPartyId(String partyName) {
 def partyId;
 def logger = context.getLogger();
 def custPVO =
 context.getViewObject("oracle.apps.cdm.foundation.parties.publicView.core.PartyPVO");
 def vc = custPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("PartyName", partyName);
 def rowset = custPVO.findByViewCriteria(vc, -1);
 def partyIdRowSet = rowset.first();
 if (partyIdRowSet != null)
 partyId = partyIdRowSet.getAttribute("PartyId");
 logger.logSevere("party id for Bill To customer", partyId);
 return partyId;
}
```

```
Object gettermID(Long AId, Long partyId) {
 def termId;
 def logger = context.getLogger();
 def custProfilePVO =
 context.getViewObject("oracle.apps.financials.receivables.publicView.analytics.CustomerProfilePVO");
 def vc1 = custProfilePVO.createViewCriteria();
 def vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("CustAcctProfileCustAccountId", AId);
 vcrow1.setAttribute("CustFinProfileSiteUseId", null);
 vcrow1.setAttribute("CustAcctProfilePartyId", partyId);
 def rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
 def profile = rowset1.first();
 if (profile != null)
 termId = profile.getAttribute("CustProfileStandardTerms");
 logger.logSevere("Term Id from customer profile VO", termId);
 return termId;
}

Object gettermName(Long termID) {
 def raTermPVO =
 ontext.getViewObject("oracle.apps.financials.receivables.publicView.TrxPaymentTermPVO");
 def vc2 = raTermPVO.createViewCriteria();
 def vcrow2 = vc2.createViewCriteriaRow();
 vcrow2.setAttribute("TermId", termID);
 def rowset2 = raTermPVO.findByViewCriteria(vc2, -1);
 def termName = rowset2.first();
 return termName;
}
```

Set Extensible Flexfield On Order Header

Set an attribute to the current date and time. Use an extensible flexfield on the order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");
//throw new ValidationException("An order with the Purchase Order Number " + poNumber + " already exists.");

if (poNumber == null || !poNumber.startsWith("SetDateEffAttributeOnSubmit")) return;

Date now = new Date();

//throw new ValidationException("An order with the Purchase Order Number " + now + " already exists.");
def complianceDetails = header.getOrCreateContextRow("ComplianceDetails");

complianceDetails.setAttribute("compliancedatetime", now);
```

Set Extensible Flexfield Values for Hazardous Items

If the HazardousMaterialFlag attribute equals Y, then set the value for attributes that use extensible flexfields on the order header and order lines.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber == null) return;

if (!poNumber.startsWith("SetEFFAttributeOnSave")) return;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
```

```
 def item = getItem(inventoryItemId, orgId);

 String hazardous = item.getAttribute("HazardousMaterialFlag");

 //throw new ValidationException("Item Hazardous Material Flag is " + hazardous);

 if ("Y".equals(hazardous)) {
 //get tow for fulfill line context PackShipInstruction
 def packShipInstruction = line.getOrCreateContextRow("PackShipInstruction");
 packShipInstruction.setAttribute("shippinginstruction", "Hazardous Handling Required.");
 }
}


Date now = new Date();

def complianceDetails = header.getOrCreateContextRow("ComplianceDetails");
complianceDetails.setAttribute("compliancedatetime", now);
complianceDetails.setAttribute("compliancereason", "This is a compliance reason.");

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 vc.add(vcrow);

 def rowset = itemPVO.findByViewCriteriaWithBindVars(vc, -1, new String[0], new String[0]);
 def item = rowset.first();

 return item;

}
```

Validate That Ship-to Site Belongs to Business Unit on Order Header

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

def VARShipToPartySiteIdentifier = header.getAttribute("ShipToPartySiteIdentifier");
def VARBusinessUnitIdentifier = header.getAttribute("BusinessUnitIdentifier");
def ShipToPartySite = getShipTo(VARShipToPartySiteIdentifier);
def RETShipToSetId = ShipToPartySite.getAttribute("SetId")
def VARBusinesseUnitSetId = 0

//Test each combination of the business unit and assignment set. Use this SQL:
//SELECT haotl.NAME ,
//haotl.organization_id,
//fsa.SET_ID
//FROM fusion.FND_SETID_ASSIGNMENTS fsa,
//fusion.HR_ORGANIZATION_UNITS_F_TL haotl
//WHERE reference_group_name LIKE 'HZ_CUSTOMER_ACCOUNT_SITE'
//AND determinant_value = haotl.organization_id
//AND haotl.NAME LIKE '&Business_Unit_Name%'
//AND haotl.LANGUAGE = USERENV('LANG')

//

if (300000017871360. equals(VARBusinessUnitIdentifier)) {
 VARBusinesseUnitSetId = 300000000002582;
}

if ((VARBusinesseUnitSetId).equals(RETShipToSetId)) {} else {
 //You can modify this validation error to match your business needs. This example includes details about
 the setIDs.
```

```
 Throw new ValidationException("BU Set Id does not match Customer Ship to SET ID - BU Set ID is " +
 VARBusinesseUnitSetId + " Ship to Set ID is :" + RETShipToSetId)
}

Object getShipTo(Long ShipToSiteId) {
 def ShipToPVO =
 context.getViewObject("oracle.apps.hed.campusCommunity.shared.shoppingCart.publicModel.view.AccountSitePVO");
 def vc = ShipToPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("PartySiteId", ShipToSiteId);
 def rowset = ShipToPVO.findByViewCriteria(vc, -1);
 def ShipTo = rowset.first();
 return ShipTo;
}
```

Validate Sales Credits for Salesperson from Order Header

Get the sales credit percent, then validate that Order Management allocates a percent greater than x to each salesperson.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;


def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber != null && poNumber.startsWith("PMC TEST")) {
 //def tokens = [EVENT_CODE: "Jason Carrier", EVENT_CRITERIA: "20"];
 //throw new ValidationException("ORA_MANAGE_EXTENSIONS", "DOO_CMN_ETP_INVALID_EVENT_DTLS", tokens);
}


def salesCredits = header.getAttribute("SalesCredits"); //Get the row set for sales credits that are
 specified on the order header.

while (salesCredits.hasNext()) {
 def salesCredit = salesCredits.next();
 if ("1".equals(salesCredit.getAttribute("SalesCreditTypeCode"))) {
 //we are dealing with revenue percent
 def percent = salesCredit.getAttribute("Percent");

 if (percent < 30) {
 def tokens = [EVENT_CODE: salesCredit.getAttribute("Salesperson"), EVENT_CRITERIA: percent];
 throw new ValidationException("ORA_MANAGE_EXTENSION", "DOO_CMN_ETP_INVALID_EVENT_DTLS", tokens);
 }
 }
}
```

Prevent Order Management from Deleting the Order Header

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
def reference = header.getAttribute("ReferenceHeaderId");
if (reference == null)
return;
def lines = header.getAttribute("Lines");
```

Prevent Duplicate Purchase Order Numbers on Order Header

Prevent Order Management from creating a duplicate of the purchase order number on the order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
//get the customer PO number, order number, and the buying party id from the order header of the sales order
 you're saving.
String customerPONumber = header.getAttribute("CustomerPONumber");
```

```
String buyingPartyIdentifier = header.getAttribute("BuyingPartyIdentifier");
String OrderNumber = header.getAttribute("OrderNumber");

//If the PO number is empty, then there's nothing to validate.
if (customerPONumber == null) return;

//Convert the PO number to upper case.
customerPONumber = customerPONumber.toUpperCase()
//Use the HeaderPVO to run a query according to customer PO number and sold to party's ID.
//Use a view criteria to set where clause.
def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HeaderPVO");
def vc = vo.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("CustomerPoNumber", customerPONumber);
vcrow.setAttribute("SoldToPartyId", buyingPartyIdentifier);
vc.add(vcrow);

//Use a view criteria to find one row. Even if we find one row, we know that we already have an order with
 the same PO number and bill to site ID
def rowset = vo.findByViewCriteria(vc, 1);

if (rowset.hasNext()) {
 //We found a sales order for the bill to site and the order number.
 //Now test to see if we alrady saved this sales order or if this is a revision.
 def vcSameOrder = vo.createViewCriteria();
 def vcSameOrderrow = vc.createViewCriteriaRow();
 vcSameOrderrow.setAttribute("CustomerPoNumber", customerPONumber);
 vcSameOrderrow.setAttribute("SoldToPartyId", buyingPartyIdentifier);
 vcSameOrderrow.setAttribute("OrderNumber", OrderNumber);
 vcSameOrder.add(vcSameOrderrow);
 def rowsetSameOrder = vo.findByViewCriteria(vcSameOrder, 1);
 if (rowsetSameOrder.hasNext()) {
 //We found a sales order that has the same order number and PO number as the order that we're already
 working on. this is a valid case. The order found is a revision or a previously saved order.
 header.setAttribute("CustomerPONumber", customerPONumber);
 } else
 //We found a different sales order for a different order number.
 //Convert the customerPO number to upper case.
 {
 throw new ValidationException("An order with the Purchase Order Number for this Customer already exists " +
 customerPONumber + " already Exists");
 }
} else
//We didn't find a sales order that has the same the bill to site id and po number, so we can save it.
//Covert the customerPO number to upper case.
{
 header.setAttribute("CustomerPONumber", customerPONumber);
}
```

Get Details for Related Sales Orders

Get details about the relationship between the sales order and other related sales orders, such as the original order of a copied order or a return order.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();
messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));
```

```
def docReferences = header.getAttribute("DocumentReferences");

messages.add(new Message(Message.MessageType.ERROR, "Header Doc"));
while (docReferences.hasNext()) {

 def docRef = docReferences.next();
 messages.add(new Message(Message.MessageType.ERROR, "Header Doc Refs"));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalNumber: " +
 docRef.getAttribute("DocumentAdditionalNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionaldentifier: " +
 docRef.getAttribute("DocumentAdditionaldentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentIdentifier: " +
 docRef.getAttribute("DocumentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineNumber: " +
 docRef.getAttribute("DocumentAdditionalLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineIdentifier" +
 docRef.getAttribute("DocumentAdditionalLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineIdentifier: " +
 docRef.getAttribute("DocumentLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineNumber: " +
 docRef.getAttribute("DocumentLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentReferenceType: " +
 docRef.getAttribute("DocumentReferenceType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineNumber: " +
 docRef.getAttribute("DocumentAdditionalSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineIdentifier: " +
 docRef.getAttribute("DocumentAdditionalSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineIdentifier: " +
 docRef.getAttribute("DocumentSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineNumbe: " +
 docRef.getAttribute("DocumentSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentNumber: " +
 docRef.getAttribute("DocumentNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineIdentifier: " +
 docRef.getAttribute("FulfillLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "LineIdentifier: " +
 docRef.getAttribute("LineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableId: " +
 docRef.getAttribute("OwnerTableId")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableName: " +
 docRef.getAttribute("OwnerTableName")));
 messages.add(new Message(Message.MessageType.ERROR, "TaskType: " + docRef.getAttribute("TaskType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSystemReferenceIdentifier: " +
 docRef.getAttribute("DocumentSystemReferenceIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + docRef.getAttribute("HeaderId")));
}

def lines = header.getAttribute("Lines");
messages.add(new Message(Message.MessageType.ERROR, "get lines"));
while (lines.hasNext()) {
 messages.add(new Message(Message.MessageType.ERROR, "A line"));
 def line = lines.next();
 def lineDocReferences = line.getAttribute("DocumentReferences");
 while (lineDocReferences.hasNext()) {
 messages.add(new Message(Message.MessageType.ERROR, "has doc references"));
 def lineDocRef = lineDocReferences.next();
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionaldentifier: " +
 lineDocRef.getAttribute("DocumentAdditionaldentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentIdentifier: " +
 lineDocRef.getAttribute("DocumentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineIdentifier" +
 lineDocRef.getAttribute("DocumentAdditionalLineIdentifier")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineIdentifier: " +
 lineDocRef.getAttribute("DocumentLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineNumber: " +
 lineDocRef.getAttribute("DocumentLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentReferenceType: " +
 lineDocRef.getAttribute("DocumentReferenceType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineIdentifier: " +
 lineDocRef.getAttribute("DocumentAdditionalSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineIdentifier: " +
 lineDocRef.getAttribute("DocumentSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineNumbe: " +
 lineDocRef.getAttribute("DocumentSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentNumber: " +
 lineDocRef.getAttribute("DocumentNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineIdentifier: " +
 lineDocRef.getAttribute("FulfillLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "LineIdentifier: " +
 lineDocRef.getAttribute("LineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableId: " +
 lineDocRef.getAttribute("OwnerTableId")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableName: " +
 lineDocRef.getAttribute("OwnerTableName")));
 messages.add(new Message(Message.MessageType.ERROR, "TaskType: " + lineDocRef.getAttribute("TaskType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSystemReferenceIdentifier: " +
 lineDocRef.getAttribute("DocumentSystemReferenceIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + lineDocRef.getAttribute("HeaderId")));
 }
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

Extend Order Lines

You can use an order management extension to create a new order line that is or isn't already related to an existing one.

Examine the Line's Status

Examine the order line's status before you update the line because you might not be able to update an attribute on an order line that's in the Shipped, Invoiced, Closed, or Cancelled status.

Assume your extension updates the Shipping Method attribute, then add a condition at the beginning of your extension that determines whether the line is already fulfilled and closed. If that condition is true, then it doesn't make sense to update the Shipping Method attribute, and you can exit the extension.

Create a New Order Line

Here's an example code snippet that uses the createNewLine method to create a new order line:

```
header.createNewLine
```

Note

- You can use the CreateLineParams() parameter with the createNewLine method.
- As an option, you can specify a condition according to the values of an existing line to determine whether to add a new line.
- Use the createNewLine method on the order header.
- Use the setAttribute method on the order line to set the values for order line attributes when you create the line. If you don't do this, then the order management extension will cascade values from the order header to the order line.
- If you use an order management extension to add a line, then the extension sets the AddedByExtensionFlag attribute to Y, and you can use it to determine whether an extension added the line.

Use the createNewLine method only:

- With lines that you price in Oracle Pricing. If you use createNewLine with a line that you price in a source system, you will encounter an error.
- With the On Save event. If you use createNewLine with any other event, you will encounter an error.
- To create a line that contains a standard item. If you use createNewLine to create a line that contains a subscription or coverage, you will encounter an error.

Here's an example that creates a new line for an order where the CustomerPONumber attribute contains the value `CreateStandaloneLine` and the line contains the AS54888 item.

```
//---
import oracle.apps.scm.doo.common.extensions.CreateLineParams;
 def poNumber = header.getAttribute("CustomerPONumber");
 if(poNumber != "CreateStandaloneLine") return;
 def createLineParams = new CreateLineParams(); // Initialize the new variable so we can send the required
 attributes.
 createLineParams.setProductNumber("AS54888"); // Add the AS54888 item to the new line.
 createLineParams.setOrderedUOM("Each"); // Set the unit of measure to Each.
 createLineParams.setOrderedQuantity(10); // Set the ordered quantity to 10.
 header.createNewLine(createLineParams); // Use the attribute values from the above lines to create the
 line. The extension will cascade the other ship to and bill to attribute values from the order header.
--//
```

Assume the product identifier is 2157 for the AS54888, and you want to use the identifier instead of the number. To specify the identifier instead of the number, replace this line:

```
createLineParams.setProductNumber("AS54888");
```

with this line:

```
createLineParams.setProductIdentifier(2157);
```

Create an Order Line That Isn't Related to Another Order Line

You can create, read, update, or cancel an order line that has a standard item or service item and you don't have to reference an existing line, but you must price the sales order in Order Management.

The extension automatically updates the AddedByExtensionFlag attribute to indicate that it added the line.

This example adds a freight line according to a condition:

- If the order line contains the AS54888 item, and if the quantity on the line is less than 5, then add a freight line.

It also adds the freight line's number to the original line.You can use that data to prevent adding another freight line during a subsequent save or revision.

If you copy the line that has the AS54888 after you add the freight line, then Order Management will also copy the extensible flexfield. Your code won't add a freight line for the copied line.

```
// Import the objects that you need to add an order line.
import oracle.apps.scm.doo.common.extensions.CreateLineParams;
// This code is for testing purposes. It makes sure the extension doesn't affect other users.
// It looks at the customer's purchase order number:
if (!"SNADDFREIGHT".equals(header.getAttribute("CustomerPONumber"))) return;

def lines = header.getAttribute("Lines");
while(lines.hasNext()){
 def line = lines.next();
 def float orderQuantity = line.getAttribute("OrderedQuantity")
 def product = line.getAttribute("ProductNumber");
 def linenumber = line.getAttribute("LineNumber");
 // Use this code to debug the extension:
 debug(" Orig Line " + linenumber + " " + orderQuantity.toString() + product +
 line.getAttribute("AddedByExtensionFlag"));

 def MinimumOrdQty = 5
 if (product == "AS54888") {
 //If the ordered quantity is more than 5, then continue to next code line.
 if (orderQuantity > MinimumOrdQty) continue;
 else {
 /* Get the extensible flexfield's context for the line. If the context isn't available, then it means the
 extension didn't add the new line, so you use getContext instead of getOrCreate.
 */
 def effContext = line.getContextRow("FulfillLineContext3");
 //If the extensible flexfield's context is available, and if the segment contains data, then go to next
 code line.
 if (effContext != null && effContext.getAttribute("_FL3AttributeChar1") != null) continue;
 //If your code reaches this point, then the extension didn't add the freight line.
 def createLineParams = new CreateLineParams();
 createLineParams.setProductNumber("Freight_GST_QPMapper");
 createLineParams.setOrderedUOM("Each");
 createLineParams.setOrderedQuantity(1);
 /* The extension will automatically use header.createNewLine to create the new line.
 This line will include the price just as it would if you added it in the Order Management work area. The
 extension sets the AddedByExtensionFlag attribute to Y on the new line.
 If you want to add a free item, then use line.createNewLine instead of header.createNewLine. You can't edit
 the free line, the free line won't contain a price, and the extension won't set the AddedByExtensionFlag
 attribute on the free line to Y.*/
 fline = header.createNewLine(createLineParams);
 // Use code line to debug this extension:
 debug("Freight " + fline.getAttribute("LineNumber"));
 // Now that you added the freight line, you can add those details to the extensible flexfield on the
 original line:
 def effContext1 = line.getOrCreateContextRow("FulfillLineContext3");
 effContext1.setAttribute("_FL3AttributeChar1",fline.getAttribute("LineNumber"));
 }
 }
}

void debug(String msg) {
 header.setAttribute("ShippingInstructions", header.getAttribute("ShippingInstructions") + msg + "\n");
}
```

Manage an Internal Material Transfer

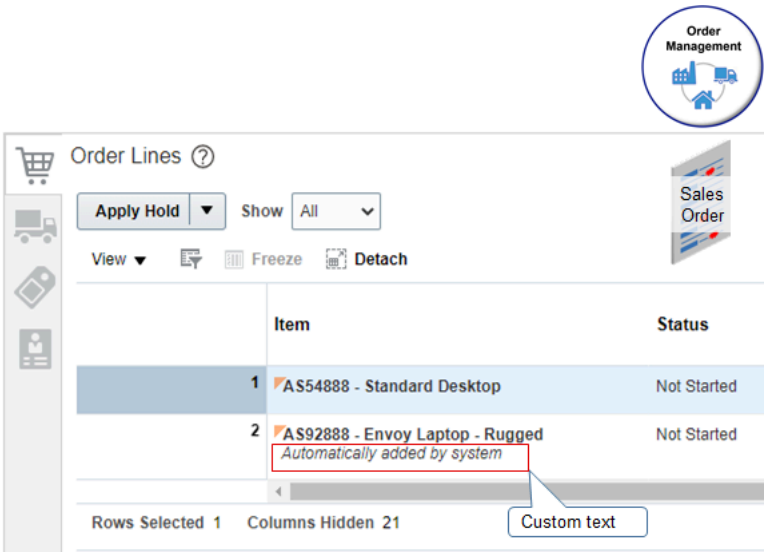
You can't use the createNewLine method with a sales order that includes an internal material transfer because the pricing for an internal material transfer is frozen.

Use this condition in your extension to skip adding a line when the order includes an internal material transfer:

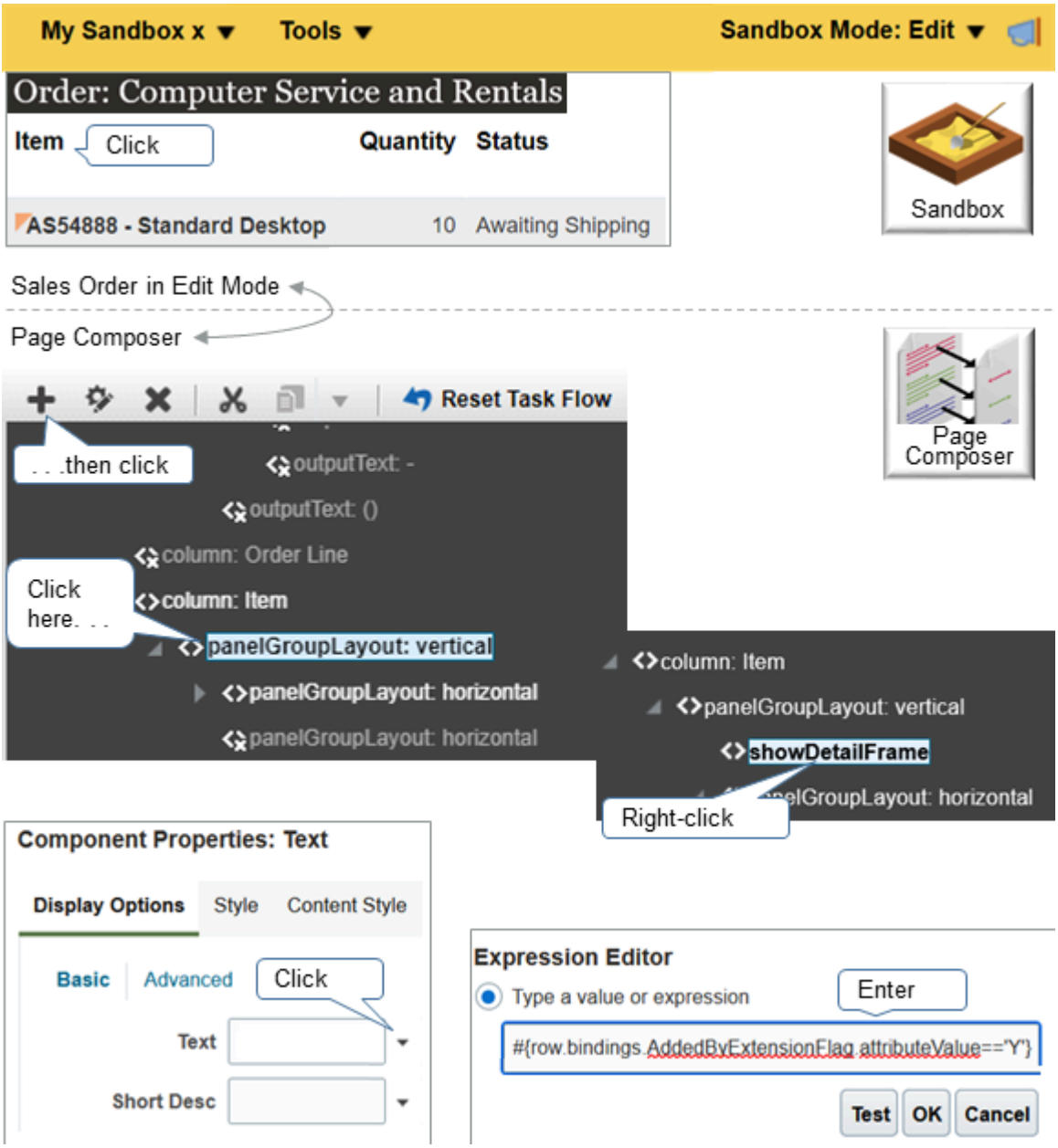
```
If (header.getAttribute("TransactionDocumentTypeCode") == "TO") return;
```

Indicate Whether an Extension Automatically Added an Item

You can customize the Order Management work area to indicate whether an Order Management Extension automatically added an item. Assume you want to display the text Automatically added by system in the description of the item on the order line.



Here's the general flow that you will use to set it up.

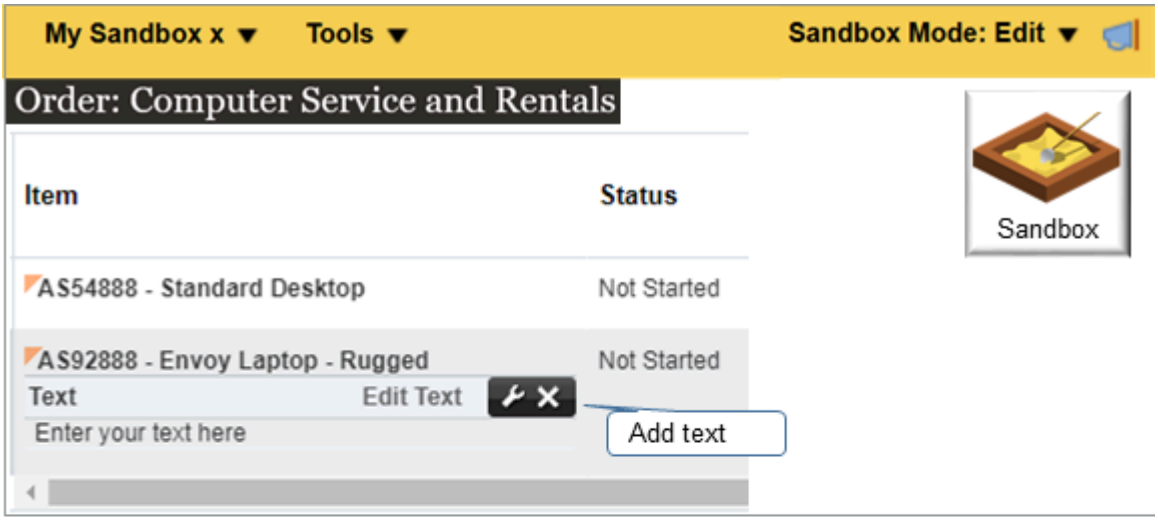

My Sandbox x
Tools
Sandbox Mode: Edit
Order: Computer Service and Rentals
Item
Click
Quantity
Status
AS54888 - Standard Desktop
10
Awaiting Shipping
Sandbox
Sales Order in Edit Mode
Page Composer
Page Composer
Reset Task Flow
. . .then click
outputText: -
outputText: ()
column: Order Line
Click here. . .
column: Item
panelGroupLayout: vertical
panelGroupLayout: horizontal
panelGroupLayout: horizontal
column: Item
panelGroupLayout: vertical
showDetailFrame
Right-click
Component Properties: Text
Display Options
Style
Content Style
Basic
Advanced
Click
Text
Short Desc
Expression Editor
Type a value or expression
Enter
#{row.bindings.AddedByExtensionFlag.attributeValue=='Y'}
Test
OK
Cancel

Try it.

1. Open Page Composer.
    - Create a sandbox, and enable Page Composer. For details, see Change How Order Management Displays Attributes.
    - Go to the Order Management work area, then click **Tasks > Manage Orders**.
    - On the Manage Orders page, search for, then open any sales order.
    - On the Order page, in the upper-right corner, click your **login name**.
    - In the Settings and Actions menu, click **Edit Pages**.

      Page Composer opens in a separate user interface that sits like a frame around the outside of the Order Management work area. You can now use Page Composer to modify the user interface for the Order Management work area in real time.
2. Access the sales order's structure.
    - Click Page Composer's Structure tab.
    - Toggle the Hide Structure, Show Structure button so you can see where the Structure window is. Its usually at the bottom of the page.
    - In the Structure window, click **Dock > Right**.
    - Expand the Structure window until you can examine it's contents.
    - On the sales order, on the order line, click the **Item column header**.
    - Notice that the Structure window automatically scrolls to the `column: Item` component.
3. Add a text component to the sales order.
    - You're going to add text inside the `panelGroupLayout: vertical` component, so click **panelGroupLayout: vertical**.
    - At the top of the structure window, click the **plus icon** (+).
    - In the Add Content Dialog, click **Components**.
    - Click **Add** in the Text row.
    - Notice that Page Composer added a new `showDetailFrame: Text` component immediately below the `panelGroupLayout: vertical` component in the structure window.
    - In the Add Content Dialog, click **Close**.
4. Add an order management extension.
    - Right-click the new **showDetailFrame: Text** component, then click **Edit**.
    - In the Component Properties dialog, on the Display Options tab, in the Basic section, click the **down arrow** next to the Text attribute, then click **Expression Builder**.
    - In the Expression Editor dialog, select the **Type a Value or Expression** option, then enter the value.
      ```
      #{row.bindings.AddedByExtensionFlag.attributeValue=='Y'}
      ```
    - Click **OK**.
    - In the Component Properties dialog, click **Apply > OK**.

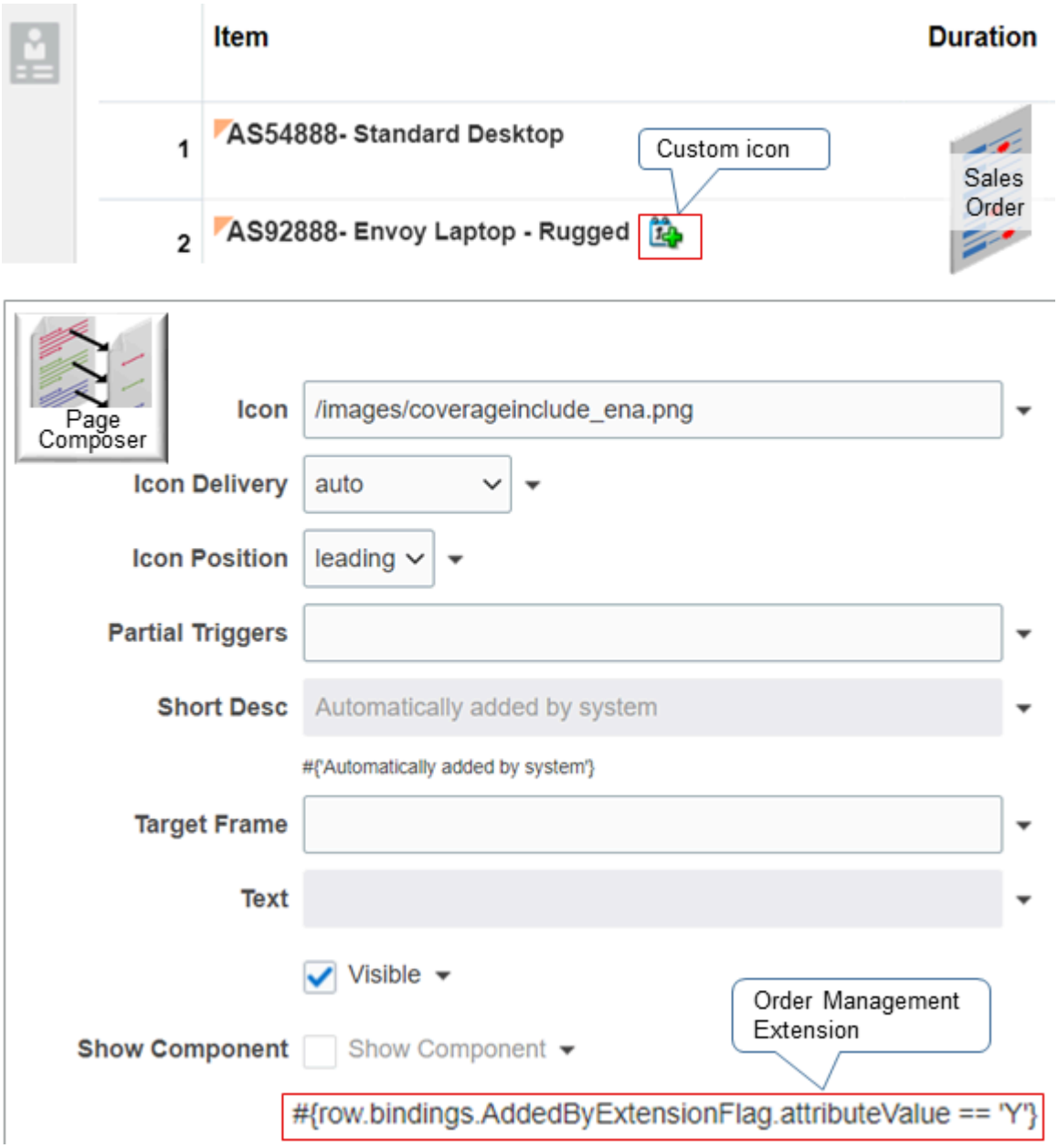
5. Add the text that you want to display.



- Notice that the Item column in the sales order now includes an area where you can enter text.
- Click **Edit Text**, then add the text that you want to display, such as *Automatically added by system*.

Display an Icon

Assume you have an icon name coverageinclude_ena.png, and you want to display it as part of the item. You can use an order management extension in Page Composer to add it:



Add a Free Item

You can add a free item to an existing item according to a condition. The line will have a price of 0.00 and you can't update it. The extension automatically updates the tranformedFrom attribute to indicate that it added this line to an existing item. For example:

- If the order line contains the AS54888 item, and if the quantity on the line is more than 10, then add a free item with a quantity 1.

You use the On Save event:

```
// import the objects that you need to add an order line
import oracle.apps.scm.doo.common.extensions.CreateLineParams;
// This code is for testing purposes. It makes sure the extension doesn't affect other users.
// It looks at the customer's purchase order number:
```

```
if (!"SNADDFREE".equals(header.getAttribute("CustomerPONumber"))) return;

def lines = header.getAttribute("Lines");
while(lines.hasNext()){
 def line = lines.next();
 def float orderQuantity = line.getAttribute("OrderedQuantity")
 def product = line.getAttribute("ProductNumber");
 def linenumber = line.getAttribute("LineNumber");
 // Use this code to debug the extension:
 debug(" Orig Line " + linenumber + " " + orderQuantity.toString() + product +
 line.getAttribute("AddedByExtensionFlag"));

 if (product == "AS54888") {
 //If the ordered quantity is less than or equal to 10, then continue to the next code line.
 if (orderQuantity <= 10) continue;
 else {
 def isClosed = line.isClosed()
 def isCanceled = line.isCanceled()
 def isTransformed = line.isTransformed();
 debug("closed " + isClosed + " canceled " + isCanceled + " transformed " + isTransformed);
 //Make sure the current order line isn't closed, cancelled, or transformed.
 if (isClosed || isCanceled || isTransformed) continue;
 //Make sure you haven't already added a free item to the order line.
 def freeLines = line.getTransformedLines();
 // The above code line gives you a list of lines that you have added for the order line.
 debug(" associated lines " + freeLines);
 //If the above code returns an empty list, then the freeLines variable will be false
 // If you need to check the count of items in the list, then use the size()method.
 if (freeLines && freeLines.size() > 0) continue;
 //If your code reaches this point, then the extension didn't add the free order line.
 def createLineParams = new CreateLineParams();
 createLineParams.setProductNumber("AS54888"); //Freight_GST_QPMapper");
 createLineParams.setOrderedUOM("Each");
 createLineParams.setOrderedQuantity(orderQuantity);
 /* The extension will automatically use line.createNewLine to add the order line that has the free item.
 This order line will have a price of 0 and you can't update it.
 If you want to add an independent line that's priced and that you can update, then use header.createNewLine
 instead of line.createNewLine.
 */
 fline = line.createNewLine(createLineParams);
 // Use this code line to debug this extension:
 debug("Free " + fline.getAttribute("LineNumber"));
 }
 }
}
```

Get Manual Price Adjustments from Order Lines

```
Iimport oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

if (!"PMC TEST".equals(header.getAttribute("CustomerPONumber"))) return;

List < Message > messages = new ArrayList < Message > ();

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def MPAS = line.getAttribute("ManualPriceAdjustments");

 while (MPAS.hasNext()) {
 def mpa = MPAS.next();
 messages.add(new Message(Message.MessageType.ERROR, "AdjustmentElementBasis is " +
 mpa.getAttribute("AdjustmentElementBasis")));
 messages.add(new Message(Message.MessageType.ERROR, "AdjustmentTypeCode is " +
 mpa.getAttribute("AdjustmentTypeCode")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "ChargeDefinitionCode is " +
 mpa.getAttribute("ChargeDefinitionCode")));
 messages.add(new Message(Message.MessageType.ERROR, "ChargeRollupFlag is " +
 mpa.getAttribute("ChargeRollupFlag")));


 messages.add(new Message(Message.MessageType.ERROR, "Comments is " + mpa.getAttribute("Comments")));
 messages.add(new Message(Message.MessageType.ERROR, "ManualPriceAdjustmentId is " +
 mpa.getAttribute("ManualPriceAdjustmentId")));

 messages.add(new Message(Message.MessageType.ERROR, "ParentEntityCode is " +
 mpa.getAttribute("ParentEntityCode")));

 messages.add(new Message(Message.MessageType.ERROR, "ParentEntityId is " +
 mpa.getAttribute("ParentEntityId")));


 messages.add(new Message(Message.MessageType.ERROR, "PricePeriodicityCode is " +
 mpa.getAttribute("PricePeriodicityCode")));
 messages.add(new Message(Message.MessageType.ERROR, "ReasonCode is " + mpa.getAttribute("ReasonCode")));
 messages.add(new Message(Message.MessageType.ERROR, "SequenceNumber is " +
 mpa.getAttribute("SequenceNumber")));

 messages.add(new Message(Message.MessageType.ERROR, "SourceManualPriceAdjustmentIdentifier is " +
 mpa.getAttribute("SourceManualPriceAdjustmentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "ValidationStatusCode is " +
 mpa.getAttribute("ValidationStatusCode")));

 messages.add(new Message(Message.MessageType.ERROR, "ChargeDefinition is " +
 mpa.getAttribute("ChargeDefinition")));
 messages.add(new Message(Message.MessageType.ERROR, "AdjustmentType is " +
 mpa.getAttribute("AdjustmentType")));
 messages.add(new Message(Message.MessageType.ERROR, "AdjustmentElementBasisCode is " +
 mpa.getAttribute("AdjustmentElementBasisCode")));

 }
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Get Fulfillment Line Attributes

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

if (!"PMC TEST".equals(header.getAttribute("CustomerPONumber"))) return;

List < Message > messages = new ArrayList < Message > ();

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def flinedetails = line.getAttribute("FulfillLineDetails");
 while (flinedetails.hasNext()) {
 def flinedetail = flinedetails.next();
 messages.add(new Message(Message.MessageType.ERROR, "ActualDeliveryDate is " +
 flinedetail.getAttribute("ActualDeliveryDate")));


 messages.add(new Message(Message.MessageType.ERROR, "AvailabilityShipDate is " +
 flinedetail.getAttribute("AvailabilityShipDate")));
 messages.add(new Message(Message.MessageType.ERROR, "BillOfLadingNumber is " +
 flinedetail.getAttribute("BillOfLadingNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "BillingTransactionNumber is " +
 flinedetail.getAttribute("BillingTransactionNumber")));
```

```
messages.add(new Message(Message.MessageType.ERROR, "DeliveryName is " +
flinedetail.getAttribute("DeliveryName")));
messages.add(new Message(Message.MessageType.ERROR, "ExceptionFlag is " +
flinedetail.getAttribute("ExceptionFlag")));

messages.add(new Message(Message.MessageType.ERROR, "Category is " +
flinedetail.getAttribute("Category")));
messages.add(new Message(Message.MessageType.ERROR, "CreatedBy is " +
flinedetail.getAttribute("CreatedBy")));
messages.add(new Message(Message.MessageType.ERROR, "CreationDate is " +
flinedetail.getAttribute("CreationDate")));
messages.add(new Message(Message.MessageType.ERROR, "CustomerTrxLineId is " +
flinedetail.getAttribute("CustomerTrxLineId")));



messages.add(new Message(Message.MessageType.ERROR, "FulfillLineDetailId is " +
flinedetail.getAttribute("FulfillLineDetailId")));
messages.add(new Message(Message.MessageType.ERROR, "FulfillLineId is " +
flinedetail.getAttribute("FulfillLineId")));
messages.add(new Message(Message.MessageType.ERROR, "LastUpdateDate is " +
flinedetail.getAttribute("LastUpdateDate")));
messages.add(new Message(Message.MessageType.ERROR, "LastUpdateLogin is " +
flinedetail.getAttribute("LastUpdateLogin")));
messages.add(new Message(Message.MessageType.ERROR, "LastUpdatedBy is " +
flinedetail.getAttribute("LastUpdatedBy")));


messages.add(new Message(Message.MessageType.ERROR, "ObjectVersionNumber is " +
flinedetail.getAttribute("ObjectVersionNumber")));
messages.add(new Message(Message.MessageType.ERROR, "Quantity is " +
flinedetail.getAttribute("Quantity")));
messages.add(new Message(Message.MessageType.ERROR, "RmaReceiptDate is " +
flinedetail.getAttribute("RmaReceiptDate")));
messages.add(new Message(Message.MessageType.ERROR, "RmaReceiptLineNumber is " +
flinedetail.getAttribute("RmaReceiptLineNumber")));
messages.add(new Message(Message.MessageType.ERROR, "RmaReceiptNumber is " +
flinedetail.getAttribute("RmaReceiptNumber")));
messages.add(new Message(Message.MessageType.ERROR, "RmaReceiptTransactionId is " +
flinedetail.getAttribute("RmaReceiptTransactionId")));
messages.add(new Message(Message.MessageType.ERROR, "Status is " + flinedetail.getAttribute("Status")));
messages.add(new Message(Message.MessageType.ERROR, "StatusAsofDate is " +
flinedetail.getAttribute("StatusAsofDate")));


messages.add(new Message(Message.MessageType.ERROR, "TaskType is " +
flinedetail.getAttribute("TaskType")));
messages.add(new Message(Message.MessageType.ERROR, "TrackingNumber is " +
flinedetail.getAttribute("TrackingNumber")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceCode is " +
flinedetail.getAttribute("TradeComplianceCode")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceExplanation is " +
flinedetail.getAttribute("TradeComplianceExplanation")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceResultCode is " +
flinedetail.getAttribute("TradeComplianceResultCode")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceTypeCode is " +
flinedetail.getAttribute("TradeComplianceTypeCode")));

messages.add(new Message(Message.MessageType.ERROR, "WaybillNumber is " +
flinedetail.getAttribute("WaybillNumber")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceName is " +
flinedetail.getAttribute("TradeComplianceName")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceCode is " +
flinedetail.getAttribute("TradeComplianceCode")));
messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceResultName is " +
flinedetail.getAttribute("TradeComplianceResultName")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "TradeComplianceTypeName is " +
 flinedetail.getAttribute("TradeComplianceTypeName")));


 }

}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Prevent Updates on Order Lines

Prevent Order Management from updating order lines that it has fulfilled, canceled, closed, backordered, or split. This extension doesn't prevent updates to lines that Order Management has canceled.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

/*
def poNumber = header.getAttribute("OrderNumber");
if (poNumber == null || !poNumber.equals("520956"))
return;
*/
def reference = header.getAttribute("ReferenceHeaderId");

if(reference != null){
 def lines = header.getAttribute("Lines");

 if (!lines.hasNext()){
 throw new ValidationException("We need more time to process the order.");
 }

 Set<Long> splitFlinesSet = null;
 def vo =
 context.getViewObject("oracle.apps.scm.doo.processOrder.publicModel.partyMerge.view.FulfillLinePVO");
 def vc1 = vo.createViewCriteria();
 def vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("HeaderId", header.getAttribute("HeaderId"));
 vcrow1.setAttribute("FulfillLineNumber", " > 1 ");
 vcrow1.setAttribute("FulfillmentSplitRefId", " is not null ");
 vcrow1.setAttribute("FulfilledQty", " is null ");
 rowset1 = vo.findByViewCriteria(vc1, -1);

 rowset1.reset();
 while (rowset1.hasNext()) {
 if (splitFlinesSet == null) {
 splitFlinesSet = new TreeSet<Long>();
 }
 def line1 = rowset1.next();
 splitFlinesSet.add(line1.getAttribute("FulfillLineId"));
 }

 if (splitFlinesSet == null) {
 return;
 }

 while (lines.hasNext()) {
 def line = lines.next();
 Long fLineId = line.getAttribute("FulfillmentLineIdentifier");
 if (!(splitFlinesSet.contains(fLineId))) {
 continue;
 }
```

```
 if('Y'.equals(line.getAttribute("ModifiedFlag")) &&
 line.getAttribute("ReferenceFulfillmentLineIdentifier") != null && !(0 ==
 line.getAttribute("OrderedQuantity")) ){
 throw new ValidationException("Backordered Split line can't be updated. DisplayLineNumber: " +
 line.getAttribute("DisplayLineNumber"));
 }
 }
}
```

Prevent Updates on Order Lines That Are on Backorder

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String orderType = header.getAttribute("TransactionTypeCode");
def excludeStatuses = ["CANCELED", "CLOSED", "PARTIAL_CLOSE"] as Set;
def forOrderTypes = ["DOMESTIC", "EXPORT", "DOMESTICPPD", "CONSIGN", "EXPORTC", "EXPORTD", "EXPORTF",
 "HWIPARTNER", "HWIPARTNERC", "HWIPARTNERD", "HWIPARTNERF", "RETURN", "VAS"] as Set;

if (orderType == null) return;
if (!forOrderTypes.contains(orderType)) return;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();

 Long lineId = line.getAttribute("LineId");
 boolean backUpdateFlag = readFlineStatus(lineId);

 if (backUpdateFlag) //if not back order line
 {
 if (!excludeStatuses.contains(line.getAttribute("StatusCode"))) {

 def BillingTransactionTypeId_Current = line.getAttribute("BillingTransactionTypeIdentifier");
 def BillingTransactionTypeId_New = line.getAttribute("BillingTransactionTypeIdentifier");

 if (orderType.equals("DOMESTIC")) {
 BillingTransactionTypeId_New = 300000034980382;
 }
 if (orderType.equals("EXPORT")) {
 BillingTransactionTypeId_New = 300000034980386;
 }
 if (orderType.equals("DOMESTICPPD")) {
 BillingTransactionTypeId_New = 300000039619347;
 }
 if (orderType.equals("CONSIGN")) {
 BillingTransactionTypeId_New = 300000034980380;
 }
 if (orderType.equals("EXPORTC")) {
 BillingTransactionTypeId_New = 300000034988180;
 }
 if (orderType.equals("EXPORTD")) {
 BillingTransactionTypeId_New = 300000034988182;
 }
 if (orderType.equals("EXPORTF")) {
 BillingTransactionTypeId_New = 300000034989759;
 }
 if (orderType.equals("HWIPARTNER")) {
 BillingTransactionTypeId_New = 300000084209774;
 }
 if (orderType.equals("HWIPARTNERC")) {
 BillingTransactionTypeId_New = 300000036925861;
 }
 if (orderType.equals("HWIPARTNERD")) {
 BillingTransactionTypeId_New = 300000036925863;
 }
```

```
 if (orderType.equals("HWIPARTNERF")) {
 BillingTransactionTypeId_New = 300000036925870;
 }
 if (orderType.equals("RETURN")) {
 BillingTransactionTypeId_New = 300000034980378;
 }
 if (orderType.equals("VAS")) {
 BillingTransactionTypeId_New = 300000034980370;
 }

 if (BillingTransactionTypeId_Current != BillingTransactionTypeId_New) {
 line.setAttribute("BillingTransactionTypeIdentifier", BillingTransactionTypeId_New);
 }
 }//Condition skipping lines of particular status codes
 }//Back order line check condition
}

/*Method that determines whether the line is shipped or backordered. If the line shipped, the method returns
 true. If the line is backordered, the method returns false.

*/
def readFlineStatus(Long lineId) {

 def lines = header.getAttribute("Lines");

 if (!lines.hasNext()) {
 throw new ValidationException("In readFlineStatus failing to read lines");
 }

 while (lines.hasNext()) {

 def line = lines.next();
 def eachLineId = line.getAttribute("LineId");
 def quantity = line.getAttribute("OrderedQuantity");

 if (lineId.equals(eachLineId)) {

 def flineDetails = line.getAttribute("FulfillLineDetails");
 if (!flineDetails.hasNext()) {
 continue;
 }

 while (flineDetails.hasNext()) {
 def eachFlineDetails = flineDetails.next();
 def status = eachFlineDetails.getAttribute("Status");
 if ("BACKORDERED".equals(status) || "SHIPPED".equals(status)) {
 return false;
 }

 }
 }
 }
 return true;
}
```

Skip Order Lines According to Status and Set the Requested Ship Date

Skip order lines that are in Fulfilled, Closed, or Canceled status, and set the Requested Ship Date on the order line to the same value that the RequestShipDate attribute contains on the order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
/* Exclude orchestration processes that you have set up where the statuses are equivalent to SHIPPED or
 FULFILLED.*/
Set excludeStatusesSet = ["CANCELED", "CLOSED", "SHIPPED", "BACKORDERED", "AWAIT_BILLING", "BILLED" ];
```

```
/*
def poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null || !poNumber.equals("CSR_2720"))
return;
*/
def requestedDate = header.getAttribute("RequestShipDate");
def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
def line = lines.next();
def statusCode = line.getAttribute("StatusCode");
if (statusCode != null && excludeStatusesSet.contains(statusCode)) {
continue;
}
/*
def comments = line.getAttribute("Comments");
if (comments == null)
comments = "TEST:";
line.setAttribute("Comments", comments + header.getAttribute("ChangeVersionNumber"));
*/
/*set the RequestedShipDate attribute on the order line to the value of the requestedDate variable.
 requestedDate contains the value of the RequestShipDate order header attribute.
line.setAttribute("RequestedShipDate",requestedDate);
line.setAttribute("ScheduleShipDate",requestedDate);
}
```

Update the ModifiedFlag Attribute for Closed Order Lines

```
if(!header.getAttribute("OrderNumber").equals("100002054")) {
 return;
}

def lines = header.getAttribute("Lines");
while(lines.hasNext()) {
 def line = lines.next();

 if(line.getAttribute("ModifiedFlag").equals("Y") && line.getAttribute("StatusCode").equals("CLOSED")) {
 line.setAttribute("ModifiedFlag", "N");
 }
}
```

Manipulate Dates on Order Lines

Here's an example that illustrates how to include dates in your extension. This extension uses the On Save extension point.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import java.sql.Date;
import java.sql.Timestamp;

if( !"SNDATE".equals( header.getAttribute("CustomerPONumber" ) ) ) return;

def pattern = "MM/dd/YYYY hh:mm:ss a";
def date = header.getAttribute("TransactionOn");
/* You can use these formats:
Letter Date or Time Type Example
G Era designator Text AD
y Year Year 1996; 96
Y year Year 2009; 09
M Month in year Month Jul
MMMM Month in year Month July
w Week in year Number 27
W Week in month Number 2
D Day in year Number 189
d Day in month Number 10
```

```
E Day name in week Text Tue
EEEE Day name in week Text Tuesday
u Day number of week Number !1 (1 = Monday, ..., 7 = Sunday)
a Am/pm marker Text PM
H Hour in day (0-23) Number 0
k Hour in day (1-24) Number 24
K Hour in am/pm (0-11) Number 0
h Hour in am/pm (1-12) Number 12
m Minute in hour Number 30
s Second in minute Number 55
S Millisecond Number 978
z Time zone Text PST
zzzz Time zone Text Pacific Standard Time;
Z Time zone RFC 822 time zone -0800
X Time zone ISO 8601 time zone -08; -0800; -08:00
*/

debug( "Order Date "+ date.format(pattern));

def headerRSD = header.getAttribute("RequestShipDate");

def headerRSD_Time = headerRSD.getTime();
def vPSD = new java.sql.Date(headerRSD_Time - getTimeInMillis(2));
def vPAD = new java.sql.Date(headerRSD_Time + getTimeInMillis(7));

def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();
 line.setAttribute("PromiseShipDate", vPSD);
 line.setAttribute("PromiseArrivalDate", vPAD);
}

public static long getTimeInMillis(long days) {
 return days * 24 * 60 * 60 * 1000;
}


void debug(String msg) {
 String text = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", text + ". " + msg);
}
```

Remove the System Date from the Contract Start Date on the Order Line

Order Management sets the Contract Start Date attribute to the system date when you copy a sales order. Use this extension to remove the system date from the Contract Start Date.

```
def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {
 def line = lines.next();
 def VARTotalContractAmount = line.getAttribute("TotalContractAmount");
 def refFlineId=line.getAttribute("ReferenceFulfillmentLineIdentifier");
 if( VARTotalContractAmount == null && refFlineId==null) {
 line.setAttribute("ContractStartDate1",null);
 }

}
```

Set the Default Value for the Unit of Measure on the Order Line

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

def poNumber = header.getAttribute("CustomerPONumber");;
def reference = header.getAttribute("ReferenceHeaderId");
```

```
def headerId = header.getAttribute("HeaderId");
def sourceOrderId = header.getAttribute("SourceTransactionIdentifier");

//If you use the Order Management work area to create the sales order, then don't run this extension.
if (headerId == sourceOrderId)
 return;
//If its an order revision, then don't run this extension. Remove this condition to run the extension for
 revisions.
if (reference != null)
 return;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def parentFLineId = line.getAttribute("RootParentLineReference");
 if (parentFLineId != null) {
 def item = getItem(inventoryItemId, orgId);
 def uomCode = item.getAttribute("PrimaryUomCode");
 if (uomCode != null) {
 line.setAttribute("OrderedUOMCode", uomCode);
 }
 }

}

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 def rowset = itemPVO.findByViewCriteria(vc, -1);
 def item = rowset.first();
 return item;
}
```

Get the Rate for a Unit of Measure

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

def serviceInvoker = context.getServiceInvoker();

String payLoad = "<ns1:invUomConvert xmlns:ns1=\"http://xmlns.oracle.com/apps/scm/inventory/uom/
uomPublicService/types/\">"+
 "<ns1:pInventoryItemId>141</ns1:pInventoryItemId>"+
 "<ns1:pFromQuantity>1</ns1:pFromQuantity>"+
 "<ns1:pFromUomCode>Ea</ns1:pFromUomCode>"+
 "<ns1:pToUomCode>Ea</ns1:pToUomCode>"+
 "<ns1:pFromUnitOfMeasure/>"+
 "<ns1:pToUnitOfMeasure/>"+
 "</ns1:invUomConvert>";

def responseBody = serviceInvoker.invokeSoapService("UOM_RATE", payLoad).getSoapBody().getTextContent();
//throw new ValidationException("response final_Uom_Rate : "+responseBody);
```

Identify Order Lines That You Partially Shipped

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String customerPONumber = header.getAttribute("CustomerPONumber");
if (customerPONumber != "SNPART") return;
List<Message> messages = new ArrayList<Message>();
```

```
// Get all of the sales order's order lines.
def Lines = header.getAttribute("Lines");
while (Lines.hasNext()) {
 def line = Lines.next();
 def flineID = line.getAttribute("FulfillmentLineIdentifier");
 debug("Fulfill Line Id " + flineID.toString());
 // Find out whether the FulfillmentSplitReferenceId attribute contains a value.
 def splitFrom = line.getAttribute("FulfillmentSplitReferenceId")
 if (splitFrom != null) {
 // Find out whether the order line is split because of a partial shipment.
 if (line.getAttribute("ShippedQuantity")== null) {
 //This is the open order line.
 debug("Line " + flineID + " is split from " + splitFrom);
 //Add your logic here.
 }
 }
}

void debug(String msg) {
 //messages.add(new Message(Message.MessageType.WARNING,msg));
 header.setAttribute("PackingInstructions", header.getAttribute("PackingInstructions") + "" + msg);
}
```

Cascade Values Between Order Lines in Configuration Models

This example cascades an attribute value from the top parent of a configuration model to all of that parent's child lines.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

def poNumber = header.getAttribute("CustomerPONumber");;

//Condition that specifies to call the extension.
if(poNumber != null && poNumber.equals("CASCADE_CONTRACT_SDATE")){
 def lines=header.getAttribute("Lines");
 long currentTime = new Date().getTime();
 def date = new java.sql.Date(currentTime);

 //Iteration all lines of the order
 while( lines.hasNext() ) {
 def line=lines.next();

 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def parentLineReference=line.getAttribute("ParentLineReference");
 def rootParentLineReference=line.getAttribute("RootParentLineReference");
 def item = getItem(inventoryItemId, orgId); // Getting item details for the line
 def itemTypeCode=item.getAttribute("BomItemType");
 def pickComponentsFlag=item.getAttribute("PickComponentsFlag");
 def parentstartdate=line.getAttribute("ContractStartDate1");
 def flineId=line.getAttribute("FulfillmentLineIdentifier");
 //Passing root line for PTO ,Hybrid and ATO Items
 if(1==itemTypeCode && flineId==rootParentLineReference && rootParentLineReference== parentLineReference &&
 parentstartdate!=null){
 updateChildLines(line);
 }
 //Passing root line For KIT Items
 else if(4==itemTypeCode && "Y".equals(pickComponentsFlag) && parentstartdate!=null){
 updateChildLines(line);
 }
 }
}

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
```

```
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 def rowset = itemPVO.findByViewCriteria(vc, -1);
 def item = rowset.first();
 return item;
}

//Iterate through all lines. Set the child lines's contract start date and end date to the parent's contract
 start date and end date.
void updateChildLines(def rootfline){
 def lines=header.getAttribute("Lines");
 def flineId=rootfline.getAttribute("FulfillmentLineIdentifier");
 def parentstartdate=rootfline.getAttribute("ContractStartDate1");
 def parentenddate=rootfline.getAttribute("ContractEndDate1");
 while( lines.hasNext() ) {
 def line=lines.next();
 def rootFlineId=line.getAttribute("RootParentLineReference");
 if(flineId==rootFlineId){
 line.setAttribute("ContractStartDate1",parentstartdate);
 line.setAttribute("ContractEndDate1",parentenddate);
 }
 }
}
```

Set the Fulfillment Organization and Shipping Instructions

If the item is shippable, then set the fulfillment organization for the order line and shipping instructions on the line to the default shipping organization that the item references.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber == null) return;

if (!poNumber.startsWith("DefaultShippingOrg")) return;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def item = getItem(inventoryItemId, orgId);

 String shippable = item.getAttribute("ShippableItemFlag");

 if ("Y".equals(shippable)) {
 Long defaultOrgId = item.getAttribute("DefaultShippingOrg");

 //msg = new Message(Message.MessageType.WARNING, "default: " + inventoryItemId + " - " + orgId + " - " +
 shippable+ " - " + defaultOrgId);
 //throw new ValidationException(msg);

 line.setAttribute("FulfillmentOrganizationIdentifier", defaultOrgId);
 line.setAttribute("FulfillmentOrganizationIdentifier", 1234);
 line.setAttribute("ShippingInstructions", "Ship From Org :" + 999);
 }
}

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
```

```
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 vc.add(vcrow);

 def rowset = itemPVO.findByViewCriteriaWithBindVars(vc, -1, new String[0], new String[0]);
 def item = rowset.first();

 return item;

}
```

Add Packing Instruction for Hazardous Items

If the item is hazardous, then set the packing instructions on the order line.

```
def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber == null) return;

if (!poNumber.startsWith("HazardousMaterial")) return;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def item = getItem(inventoryItemId, orgId);

 String hazardous = item.getAttribute("HazardousMaterialFlag");

 if ("Y".equals(hazardous)) {
 //get row for fulfill line context PackShipInstruction
 line.setAttribute("PackingInstructions", "Hazardous Handling Required.");
 }
}

Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 vc.add(vcrow);

 def rowset = itemPVO.findByViewCriteriaWithBindVars(vc, -1, new String[0], new String[0]);
 def item = rowset.first();

 return item;

}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

Extend Order Revisions

Use an order management extension to extend your order revisions.

See If It's a Revision

An order management extension doesn't distinguish between a new order and an order revision, by default. You can inspect the ReferenceHeaderId attribute on the header entity and the ReferenceFulfillmentLineIdentifier attribute on the Line entity to determine whether the sales order is new or a revision.

This extension uses the On Save extension point.

```
/*
 This extension compares some of the attribute of the revision line to the existing sales order.
 */
import oracle.apps.scm.doo.common.extensions.ValidationException;
import java.util.Date;

def postShippedStatuses = ["AWAIT_BILL","ASSET_COMPLETED","BILLED","SHIPPED","CLOSED"];
def startTime = context.getCurrentTime();
if (!"SNCHECKREV".equals(header.getAttribute("CustomerPONumber"))) return;

Integer changeVersionNumber = header.getAttribute("ChangeVersionNumber");
Long runningHeaderId = header.getAttribute("ReferenceHeaderId");

debug(changeVersionNumber.toString() + " ref " + runningHeaderId.toString());
//If this sales order isn't a revision, then don't run the extension.
if (changeVersionNumber == null || changeVersionNumber <= 1 || runningHeaderId == null) return;

try {

 // Create an array of order lines from the existing sales order.
 Map linesFromRunningOrder = getLinesFromRunningOrder(runningHeaderId);
 debug("Line from running order" + linesFromRunningOrder.size());

 def lines = header.getAttribute("Lines");

 while (lines.hasNext()) {

 def line = lines.next();

 //Skip order lines that are already cancelled or are being cancelled as part of this revision.
 if (line.isCanceled() || line.getAttribute("OrderedQuantity") == 0) continue;

 Long referenceFlineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");
 if (referenceFlineId == null) {
 // Nothing to do here. This is a new order line in the current version of the order.
 debug("No Fline Ref");
 continue;
 }

 // Get the corrosponding order line from existing sales order.
 def runningLine = linesFromRunningOrder.get(referenceFlineId);

 if (runningLine == null) {
 // we have an error. The extension can't find a fulfillment line that has referenceFlineId. We should
 ideally never get here.
 throw new ValidationException("Something's not right. The extension can't find a fulfillment line that has
 referenceFlineId.");
 }

 Date priorRequestedShipDate = runningLine.getAttribute("FulfillLineRequestShipDate");
 Date priorRequestedArrivalDate = runningLine.getAttribute("FulfillLineRequestArrivalDate");
 def priorShippedQuantity = runningLine.getAttribute("FulfillLineShippedQty");
```

```
 def priorOrderedQty = runningLine.getAttribute("FulfillLineOrderedQty");

 def lineStatus = line.getAttribute("StatusCode");
 debug("StatusCode: " + lineStatus + priorRequestedShipDate + " " + priorRequestedArrivalDate);

 //Skip order lines that already shipped.
 if (postShippedStatuses.contains(lineStatus)) continue;

 // Skip order lines that already shipped.
 if (priorShippedQuantity != null && priorShippedQuantity > 0) continue;

 // Skip order lines that are already cancelled.
 if (priorOrderedQty != null && priorOrderedQty == 0) continue;

 Date lineRequestedShipDate = line.getAttribute("RequestedShipDate");
 Date lineRequestedArrivalDate = line.getAttribute("RequestedArrivalDate");

 def changeInRequestDateAttributes = (areDifferent(priorRequestedShipDate, lineRequestedShipDate)
 || areDifferent(priorRequestedArrivalDate, lineRequestedArrivalDate));

 /*
 If the requested ship date or the requested arrival date changed, then do something.
 */
 if (changeInRequestDateAttributes) {
 //Put your code here.
 debug("One of the dates is diff");
 }
 }
} finally {
 def endTime = context.getCurrentTime();
 long diff = endTime.getTime() - startTime.getTime();
 debug("Time taken by extension " + context.getExtensionName() + ";" + diff.toString());
}


/*
 * Return a boolean value that indicates whether the two incoming dates are same or different.
 */
boolean areDifferent(def v1, def v2) {
 if (v1 == null || v2 == null) return v1 != v2;

 return v1.compareTo(v2) != 0;
}


/*
 * Use the fline id to return a map of the sales order's order lines.
 */
Map getLinesFromRunningOrder(Long runningHeaderId) {

 //Create an Instance of the FulfillLinePVO public view object.
 def flinePVO = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");

 // Create a view criteria object.
 def vc = flinePVO.createViewCriteria();

 // Create a view criteria row.
 def vcrow = vc.createViewCriteriaRow();

 // Set query conditions on the view criteria row.
 vcrow.setAttribute("FulfillLineHeaderId", runningHeaderId);
 vc.add(vcrow);

 def rowset = flinePVO.findByViewCriteriaWithBindVars(vc, -1, new String [0] , new String [0] );

 Map linesMap = new HashMap();
```

```
 while (rowset.hasNext()) {
 def fline = rowset.next();

 Long fLineId = fline.getAttribute("FulfillLineId");
 linesMap.put(fLineId, fline);
 }

 return linesMap;
}

void debug(String msg) {
 String text = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", text + ". " + msg);
}
```

Run Extension Only for Order Revisions

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
def reference = header.getAttribute("ReferenceHeaderId");
List<Message> messages = new ArrayList<Message>();
 if(reference!=null){//firing for revisions
 //Include logic here that you want to run only for an order revision.
 }
```

Validate Revision on Drop Ship Order

Make sure the purchase order isn't on hold when you revise a sales order in a drop ship flow.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber == null) return;

if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();

messages.add(new Message(Message.MessageType.ERROR, "In Code"));

def lines = header.getAttribute("Lines");

def headerPVO =
 context.getViewObject("oracle.apps.scm.doo.processOrder.publicModel.partyMerge.view.HeaderPVO");

def vc = headerPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("SourceOrderNumber", header.getAttribute("SourceTransactionNumber"));
vcrow.setAttribute("SourceOrderSystem", header.getAttribute("SourceTransactionSystem"));
vcrow.setAttribute("StatusCode", "OPEN");

def rowset = headerPVO.findByViewCriteria(vc, -1);
def headerPRow = rowset.first();
def headerId = null;

if (headerPRow != null)
 headerId = headerPRow.getAttribute("HeaderId");
else {
 //ValidationException ex = new ValidationException(messages);
 //throw ex;
 return;
}
```

```
while (lines.hasNext()) {
 def line = lines.next();

 def linePVO =
 context.getViewObject("oracle.apps.scm.doo.processOrder.publicModel.partyMerge.view.FulfillLinePVO");

 def vcLine = linePVO.createViewCriteria();
 def vcrowLine = vcLine.createViewCriteriaRow();
 vcrowLine.setAttribute("HeaderId", headerId);
 vcrowLine.setAttribute("SourceLineId", line.getAttribute("SourceTransactionLineIdentifier"));

 def rowsetLine = linePVO.findByViewCriteria(vcLine, -1);
 def linePRow = rowsetLine.first();
 def flineId = null;

 if (linePRow != null)
 flineId = linePRow.getAttribute("FulfillLineId");
 else
 continue;

 def docRefPVO =
 context.getViewObject("oracle.apps.scm.doo.common.pricing.integration.publicView.DocumentReferencePVO");

 def vcDr = docRefPVO.createViewCriteria();
 def vcrowDr = vcDr.createViewCriteriaRow();
 vcrowDr.setAttribute("HeaderId", headerId);
 vcrowDr.setAttribute("FulfillLineId", flineId);
 vcrowDr.setAttribute("DocRefType", "DROPSHIP_PO_REFERENCE");

 def rowsetDr = docRefPVO.findByViewCriteria(vcDr, -1);
 def drPRow = rowsetDr.first();
 def poHeaderId = null;

 if (drPRow != null)
 poHeaderId = drPRow.getAttribute("DocId");
 else
 continue;


 def poPVO = context.getViewObject("oracle.apps.prc.po.publicView.PurchasingDocumentHeaderPVO");

 def vcPo = poPVO.createViewCriteria();
 def vcrowPo = vcPo.createViewCriteriaRow();
 vcrowPo.setAttribute("PoHeaderId", poHeaderId);

 def rowsetPo = poPVO.findByViewCriteria(vcPo, -1);
 def poPRow = rowsetPo.first();

 if (poPRow != null) {
 if ("Y".equals(poPRow.getAttribute("FrozenFlag")) || "ON
 HOLD".equals(poPRow.getAttribute("DocumentStatus")))
 throw new ValidationException("PO is frozen. OM Change cannot be submitted");

 def pvPVO = context.getViewObject("oracle.apps.prc.po.publicView.PurchasingDocumentVersionPVO");

 def vcPv = pvPVO.createViewCriteria();
 def vcrowPv = vcPv.createViewCriteriaRow();
 vcrowPv.setAttribute("PoHeaderId", poHeaderId);
 vcrowPv.setAttribute("ChangeOrderStatus", "INCOMPLETE");

 def rowsetPv = pvPVO.findByViewCriteria(vcPv, -1);
 def pvPRow = rowsetPv.first();

 if (pvPRow != null)
```

```
 messages.add(new Message(Message.MessageType.ERROR, "PO Change is in progress. OM Change cannot be
 submitted"));

 } else
 continue;
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Copy an Attribute from a Previous Revision

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def po = header.getAttribute("CustomerPONumber");
def reference = header.getAttribute("ReferenceHeaderId");

previousLinesMap = [:];
if (reference == null)
return;

def lines = header.getAttribute("Lines");
populateReferenceLines(reference);
 while (lines.hasNext()) {
 def line = lines.next();
 def sourceLineId=line.getAttribute("SourceTransactionLineIdentifier");
 def originalLine=previousLinesMap.get(sourceLineId);
 def currentTaxCode=line.getAttribute("TaxClassificationCode");
 if(originalLine!=null && currentTaxCode==null){
 def orginalTaxCode=originalLine.getAttribute("FulfillLineTaxClassificationCode");
 line.setAttribute("TaxClassificationCode",orginalTaxCode);
 }
}

void populateReferenceLines(Long headerId) {
 def vo =context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 def vc1 = vo.createViewCriteria();
 def vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("FulfillLineHeaderId", headerId);
 rowset1 = vo.findByViewCriteria(vc1, -1);

 while (rowset1.hasNext()) {
 def originalFLine=rowset1.next();

 previousLinesMap.put(originalFLine.getAttribute("FulfillLineSourceLineId"),originalFLine);
 }
}
```

Extend Return Orders

Use an order management extension to manage your return orders.

Copy Extensible Flexfield Values from the Original Fulfillment Line to Return Order

This example copies extensible flexfield values from the original fulfillment line to a return order.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import java.util.logging.Level;

Long lineId = 0;
Long headerId = 0;
def lines = header.getAttribute("Lines");
```

```
while( lines.hasNext() ) {
 def line = lines.next();

 String categoryCode = line.getAttribute("TransactionCategoryCode");

 if ("RETURN" != categoryCode) continue;
 def docRefs = line.getAttribute("DocumentReferences");
 lineId = 0;
 while(docRefs.hasNext() & lineId == 0) {

 def docRef = docRefs.next();
 String docRefType = docRef.getAttribute("DocumentReferenceType");
 if( "ORIGINAL_ORCHESTRATION_ORDER" == docRefType ) {
 // We found the document reference that references the original order.
 // The DocumentSubLineIdentifier attribute identifies the fulfillment line on the original order.

 lineId = new Long(docRef.getAttribute("DocumentSubLineIdentifier"));

 if(headerId == 0) {
 // The DocumentIdentifier attribute identifies the header of the original order.
 headerId = new Long(docRef.getAttribute("DocumentIdentifier"));
 if (headerId !=0){
 def voRow = getContextHeaderRow(headerId, "HeaderBIEffBComplianceDetailsBIVO");
 def complianceInfo = voRow.getAttribute("_ComplianceInfo");

 def context = header.getOrCreateContextRow("ComplianceDetails");
 context.setAttribute("_ComplianceInfo",complianceInfo);
 }
 }
 }
 }
 if ( lineId != 0) {
 // throw new ValidationException(lineId.toString());
 /*
 def oLine = getLine(lineId);
 if (oLine == null) continue;

 def Ocontext = oLine.getAttribute("FulfillLineBIEffEFFBIFlattened");
 def oContextRow = Ocontext.first();
 def oattr1 = oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1_EffLineId");
 def oattr = oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1__FL1AttributeNum1");
 */

 def voRow = getContextVORow(lineId, "FulfillLineBIEffBPackShipInstructionBIVO");
 def contNum1 = voRow.getAttribute("_PackingInstruction");

 def context = line.getOrCreateContextRow("PackShipInstruction");
 context.setAttribute("_PackingInstruction",contNum1);
 // context.setAttribute("_FL1AttributeNum1",23323);



 }
 else{
 throw new ValidationException("We couldn't find a value.");

 }

// def effValues = line.getOrCreateContextRow("SODetails");
// effValues.setAttribute("cogs",12);

}

def Object getLine(Long lineId) {

def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
```

```
 Object[] rows = vo.findByKey(lineId, 1);
 def originalFLine = rows[0];

 return originalFLine;
}

/**
 * Returns a row for the context on the order header.
 * @param contextVOName identifies the name of the context for the virtual object. To get this value, use
 the context code or look it up in the extensible flexfield archive.
 * under oracle/apps/publicFlex/scm/doo/headerContextsB/analytics/view
 */
Object getContextVORow(Long headerId, String contextVOName) {
 def contextVO = context.getViewObject("oracle.apps.publicFlex.scm.doo.fulfillLineContextsB.analytics.view."
 + contextVOName);
 def vc = contextVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("FulfillLineId", headerId);
 def rowset = contextVO.findByViewCriteria(vc, -1);
 def effRow = rowset.first();
 return effRow;
}

Object getContextHeaderRow(Long headerId, String contextVOName) {
 def contextVO = context.getViewObject("oracle.apps.publicFlex.scm.doo.headerContextsB.analytics.view." +
 contextVOName);
 def vc = contextVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("HeaderId", headerId);
 def rowset = contextVO.findByViewCriteria(vc, -1);
 def effRow = rowset.first();
 return effRow;
}
```

Copy Extensible Flexfield from Original Order to Return Order

This example copies extensible flexfield data from the original order to a return order that includes a return material authorization (RMA).

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import java.util.logging.Level;

//Following lines are for applying this extension only to specific cases
String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (poNumber != "PMC TEST") return;


Long lineId = 0;
Long headerId = 0;

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();

 //get the category code specified on the line. A category code of RETURN means it is a return line
 String categoryCode = line.getAttribute("TransactionCategoryCode");

 if ("RETURN" != categoryCode) continue;

 //we have a return line. Now let's find the original line
 //we will go through the document references on this line to locate the document reference
 //that identifies the original order line.
```

```
def docRefs = line.getAttribute("DocumentReferences");
lineId = 0;
while (docRefs.hasNext() & lineId == 0) {

def docRef = docRefs.next();
String docRefType = docRef.getAttribute("DocumentReferenceType");
if ("ORIGINAL_ORCHESTRATION_ORDER" == docRefType) {
//We found the document reference that points to the original order.
//The DocumentSubLineIdentifier attribute is the fulfillline id of the original order fulfillment line

lineId = new Long(docRef.getAttribute("DocumentSubLineIdentifier"));

if (headerId == 0) {
//The DocumentIdentifier attribute is the header id of the original order header
headerId = new Long(docRef.getAttribute("DocumentIdentifier"));
}
}
}
/*
if (lineId != 0) {

//call method getline to get the original line a PVO
def oLine = getLine(lineId);
if (oLine == null) continue;
//Define the context variable that you want to copy value to
def context1 = line.getOrCreateContextRow("FulfillLineContext1");
//Define the next context variable that you want to copy value to
def packShipContext = line.getOrCreateContextRow("PackShipInstruction");
//Define the next context variable that you want to copy value to
def itemMaterialContext = line.getOrCreateContextRow("Item Material");
//get Context form the oldline
def Ocontext = oLine.getAttribute("FulfillLineBIEffEFFBIFlattened");
//get to the first row of the context for the original order fulfillment line
def oContextRow = Ocontext.first();
//Attribute name will be like DOO_FULFILL_LINES_ADD_INFO_<contextname>_<segmentname>
//You can download the Flexfield archives and navigate to oracle\apps\scm\doo\processOrder\flex
\fulfillLineCategories\view\
//and look for j_ExtendedDeclarativeprivateVO.xml. This is where you will find all the flex attribute names
in above fashion
def attr1 = oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1__FL1AttributeNum1");
context1.setAttribute("_FL1AttributeNum1", attr1);
context1.setAttribute("_FL1AttributeChar1",oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1
context1.setAttribute("_FL1AttributeDate1",oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1
itemMaterialContext.setAttribute("lineMaterialSgmt",oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_Item__Mate
itemMaterialContext.setAttribute("lineMaterialText",oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_Item__Mate
packShipContext.setAttribute("_PackingInstruction",oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_PackShipIns
}
}
*/

if (headerId != 0) {

def oHeader = getHeader(headerId)

if (oHeader == null) continue;
//get Context from the original header
def context = oHeader.getAttribute("HeaderBIEffEFFBIFlattened");

//get to the first row of the context for the original header
def contextRow = context.first();

//Attribute name will be like DOO_HEADERS_ADD_INFO_<contextname>_<segmentname>
//You can download the Flexfield archives and navigate to oracle\apps\scm\doo\processOrder\flex
\headerCategories\view\
```

```
 //and look for j_ExtendedDeclarativeprivateVO.xml. This is where you will find all the flex attribute names
 in above fashion
 def complInfo = contextRow.getAttribute("DOO_HEADERS_ADD_INFO_PMC__Pricing_partyid");

 //Define the context variable that you want to copy value to - header is the header of the return oorder
 which is currently being worked on
 def complianceDetails = header.getOrCreateContextRow("PMC Pricing");
 complianceDetails.setAttribute("partyid", complInfo);

 /* This SQL will be useful in identifying the correct values to set:

select
fdsb.context_code
,fdsb.segment_code
,fdsb.SEGMENT_IDENTIFIER
from
fnd_df_segments_tl fdst,
fnd_df_segments_b fdsb
where
fdst.APPLICATION_ID = fdsb.APPLICATION_ID and
fdst.ENTERPRISE_ID = fdsb.ENTERPRISE_ID and
fdst.DESCRIPTIVE_FLEXFIELD_CODE = fdsb.DESCRIPTIVE_FLEXFIELD_CODE and
fdst.CONTEXT_CODE = fdsb.CONTEXT_CODE and
fdst.SEGMENT_CODE = fdsb.SEGMENT_CODE and
fdst.language = 'US' and
fdst.descriptive_flexfield_code = 'DOO_HEADERS_ADD_INFO'
order by
fdst.CONTEXT_CODE,
fdsb.SEQUENCE_NUMBER

In this instance the Sql returned

CONTEXT_CODE SEGMENT_CODE SEGMENT_IDENTIFIER

PMC Pricing PartyID partyid

The value DOO_HEADERS_ADD_INFO_PMC__Pricing_partyid - is derived from :

Context_code = PMC Pricing , note the space is replaced by __
Segment_identifier = partyid

*/


 }
}

def Object getLine(Long lineId) {

 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");

 Object[] rows = vo.findByKey(lineId, 1);
 def originalFLine = rows[0];

 return originalFLine;
}

def Object getHeader(Long headerId) {
 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HeaderPVO");

 Object[] rows = vo.findByKey(headerId, 1);
 def originalHeader = rows[0];
 return originalHeader;
}

void debug(String msg) {
```

```
 header.setAttribute("ShippingInstructions", header.getAttribute("ShippingInstructions") + ", " + msg);
}
```

Get Values from the Original Order

A return order doesn't include values from the original order on some order line attributes. For example, the return line doesn't include the original value for the sales credit or purchase order number. You can use an extension to get the value from the original order line. The following examples in this topic get values from the original order when you use a return material authorization (RMA).

Get Order Type from Original Order

This example gets the order type from the original sales order and uses it to set the value of the order type of a return order.

```
// Extension : DefaultOrderTypeFromOrigOrder
//
//============================================================
import oracle.apps.scm.doo.common.extensions.ValidationException;

// Go through all the order lines. We're interested in return lines. As soon as we find a return line, we
 will use the document
// reference on the line to try to locate the original order.
def lines=header.getAttribute("Lines");

while (lines.hasNext()) {

 def line=lines.next();

 // Get the line type specified on the line. A line type code of ORA_RETURN means its a return line.
 String lineTypeCode=line.getAttribute("TransactionLineTypeCode");

 if ("ORA_RETURN"==lineTypeCode) {
 // We have a return line. Now let's find the original order.
 // We will go through the document references on this line to locate the document reference
 // that identifies the original order.
 def docRefs=line.getAttribute("DocumentReferences");

 while (docRefs.hasNext()) {
 def docRef=docRefs.next();
 String docRefType=docRef.getAttribute("DocumentReferenceType");

 if ("ORIGINAL_ORCHESTRATION_ORDER"==docRefType) {
 // We found the document reference that points to the original order.
 // The DocumentIdentifier attribute is the header id of the original order.
 Long headerId=new Long(docRef.getAttribute("DocumentIdentifier"));
 // Call the getOrderType method to get the order type of the original order using a PVO
 String orderTypeCode=getOrderType(headerId);

 debug("Original Order Order Type: "+ orderTypeCode);

 // If order type isn't null, set it on the current order, else use a validation exception.
 if (orderTypeCode !=null) {
 header.setAttribute("TransactionTypeCode", orderTypeCode);
 break;
 }

 else {
 throw new ValidationException("Order type isn't specified on the original order.");
 }
 }
 }
 }
}
```

```
/**
* Returns the order type of the indicated order.
*/
String getOrderType(Long headerId) {
 debug("finding order by headerId: "+ headerId);

 def vo=context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HeaderPVO");

 Object[] rows=vo.findByKey(headerId, 1);
 def originalHeader=rows[0];

 return originalHeader.getAttribute("OrderTypeCode");

void debug(String msg) {
// Print a debug message. This code appends the string passed in as parameter to the shipping instructions
 attribute.
// To avoid performance problems, you must comment out this debug statement in your production environment.
 header.setAttribute("ShippingInstructions", header.getAttribute("ShippingInstructions") + msg + "\n");
}
```

Get Purchase Order from Original Order

This example gets purchase order details from the order header and order line of the original order, then copies them to the return order and return order line.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import java.util.logging.Level;
Long lineId = 0;
Long headerId = 0;
def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 String categoryCode = line.getAttribute("TransactionCategoryCode");

 if ("RETURN" != categoryCode) continue;
 def docRefs = line.getAttribute("DocumentReferences");
 lineId = 0;
 while (docRefs.hasNext() & lineId == 0) {
 def docRef = docRefs.next();
 String docRefType = docRef.getAttribute("DocumentReferenceType");

 if ("ORIGINAL_ORCHESTRATION_ORDER" == docRefType) {
 // We found the document reference that points to the original order.
 // The DocumentSubLineIdentifier attribute is the fulfillline id of the original order fulfillment line.
 lineId = new Long(docRef.getAttribute("DocumentSubLineIdentifier"));

 if (headerId == 0) {
 // The DocumentIdentifier attribute is the header id of the original order header.
 headerId = new Long(docRef.getAttribute("DocumentIdentifier"));
 }
 }
 }
 if (lineId != 0) {
 // throw new ValidationException(lineId.toString());
 def oLine = getLine(lineId);

 if (oLine == null) continue;
 def oLinePO = oLine.getAttribute("FulfillLineCustomerPoNumber");
 def oHeaderPO = oLine.getAttribute("HeaderCustomerPoNumber");
 line.setAttribute("CustomerPONumber", oLinePO);
 header.setAttribute("CustomerPONumber", oHeaderPO);
 } else {
 throw new ValidationException("Value not found");
```

```
 }
}
 def Object getLine(Long lineId) {
 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 Object[] rows = vo.findByKey(lineId, 1);
 def originalFLine = rows[0];
 return originalFLine;
}
```

Get Extensible Flexfields from Original Order

This example gets extensible flexfields from the order line of the original order, then copies it to a return order line.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import java.util.logging.Level;
Long lineId = 0;
Long headerId = 0;
def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();

 String categoryCode = line.getAttribute("TransactionCategoryCode");

 if ("RETURN" != categoryCode) continue;
 def docRefs = line.getAttribute("DocumentReferences");
 lineId = 0;
 while (docRefs.hasNext() & lineId == 0) {
 def docRef = docRefs.next();
 String docRefType = docRef.getAttribute("DocumentReferenceType");
 if ("ORIGINAL_ORCHESTRATION_ORDER" == docRefType) {
 // We found the document reference that references the original order.
 // The DocumentSubLineIdentifier attribute identifies the fulfillment line in the original order.

 lineId = new Long(docRef.getAttribute("DocumentSubLineIdentifier"));
 if (headerId == 0) {
 // The DocumentIdentifier attribute identifies the header in the original order.
 headerId = new Long(docRef.getAttribute("DocumentIdentifier"));
 }
 }
 }
 if (lineId != 0) {
 // throw new ValidationException(lineId.toString());
 def oLine = getLine(lineId);
 if (oLine == null) continue;

 def Ocontext = oLine.getAttribute("FulfillLineBIEffEFFBIFlattened");
 def oContextRow = Ocontext.first();
 def oattr1 = oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1_EffLineId");
 def oattr = oContextRow.getAttribute("DOO_FULFILL_LINES_ADD_INFO_FulfillLineContext1__FL1AttributeNum1");

 def context = line.getOrCreateContextRow("FulfillLineContext1");
 context.setAttribute("_FL1AttributeNum1", oattr);
 } else {
 throw new ValidationException("Value not found");

 }
}
def Object getLine(Long lineId) {

 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 Object[] rows = vo.findByKey(lineId, 1);
 def originalFLine = rows[0];

 return originalFLine;
```

```
}
```

Make Sure Business Unit on Return Matches Business Unit on Original Order

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

/*
def poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null || !("VS_BU_TEST".equals(poNumber))) {
 return;
}
*/

Set<Long> referencedReturnLinesHeaderIds = null;

def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {

 def line = lines.next();

 def categoryCode = line.getAttribute("TransactionCategoryCode");
 if (categoryCode == null || !(categoryCode.equals("RETURN"))) {
 continue;
 }

 def docRefs = line.getAttribute("DocumentReferences");
 Long headerId = null;
 while(docRefs.hasNext()) {

 def docRef = docRefs.next();
 String docRefType = docRef.getAttribute("DocumentReferenceType");
 if( "ORIGINAL_ORCHESTRATION_ORDER".equals(docRefType) ) {
 if (referencedReturnLinesHeaderIds == null) {
 referencedReturnLinesHeaderIds = new HashSet<Long>();
 }
 headerId = new Long(docRef.getAttribute("DocumentIdentifier"));
 if (!(referencedReturnLinesHeaderIds.contains(headerId))) {
 referencedReturnLinesHeaderIds.add(headerId);
 }
 break;
 }
 }
}

if (referencedReturnLinesHeaderIds != null && referencedReturnLinesHeaderIds.size()>0) {

 def buId = header.getAttribute("BusinessUnitIdentifier");

 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.HeaderPVO");
 def vc = vo.createViewCriteria();

 def vcrow = vc.createViewCriteriaRow();
 Object inQuery = " in " + referencedReturnLinesHeaderIds;
 inQuery = inQuery.replace('[', '(');
 inQuery = inQuery.replace(']', ')');
 vcrow.setAttribute("HeaderId", inQuery);

 def rowset = vo.findByViewCriteria(vc, -1);
 def row = null;
 if(rowset.hasNext()){
 row = rowset.next();
 def returnBuId = row.getAttribute("BUBusinessUnitId");
 if (returnBuId != buId) {
 throw new ValidationException("The request failed. Make sure the business unit of this return order matches
 the business unit on the original order, then resubmit your sales order. ");
 }
```

```
  }
}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

Extend Attachments

Use an order management extension to create, read, update, or delete an attachment.

Overview

Use a text or XML attachment file. Don't use a binary file.

Note

1. Use get methods to get attachment details from an attachment that already exists, such as getText to get all the text that the attachment contains, use getPklValue to get the sales order or order line that references the attachment, getFileName to get the name of the attachment file name, and so on. Here's some of the methods you can use.
    - getText
    - getPklValue
    - getEntityName
    - getDataTypeCode
    - getTitle
    - getDescription

- getFileName
- getUrl
- getCategoryName

2. Use the createAttachment method to create an attachment, and use `set` methods to specify how to create it. You can specify a new attachment file of type File, Text, or URL. For example, use setFileName to specify the name of the attachment file. Here are some of the methods you can use.
   - setDataTypeCode
   - setTitle
   - setDescription
   - setFileName
   - setFileContentType
   - setFileContent
3. Use the deleteAttachment method to delete an attachment. In this example, the code uses a `for` statement to iterate through all the attachments that a sales order contains and deletes all of them.

Get Attachments from Order Headers

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Attachment;
import oracle.jbo.domain.BlobDomain;

String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();
//messages.add(new Message( Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));


//////////////////Read header attachments
def attachments = header.getAttachments();
for (int i = 0; i < attachments.size(); ++i) {
 def attachment = attachments[i];
 messages.add(new Message(Message.MessageType.ERROR, "Pk1value:" + attachment.getPk1Value()));
 messages.add(new Message(Message.MessageType.ERROR, "Entityname:" + attachment.getEntityName()));
 messages.add(new Message(Message.MessageType.ERROR, "DatatypeCode:" + attachment.getDatatypeCode()));
 messages.add(new Message(Message.MessageType.ERROR, "Title:" + attachment.getTitle()));
 messages.add(new Message(Message.MessageType.ERROR, "Description:" + attachment.getDescription()));
 messages.add(new Message(Message.MessageType.ERROR, "FileName:" + attachment.getFileName()));
 messages.add(new Message(Message.MessageType.ERROR, "File Content Type:" +
 attachment.getFileContentType()));
 def blobDomainData = attachment.getFileContent();
 messages.add(new Message(Message.MessageType.ERROR, "File Content:" + blobDomainData.toString()));
 messages.add(new Message(Message.MessageType.ERROR, "Url:" + attachment.getUrl()));
 messages.add(new Message(Message.MessageType.ERROR, "Text:" + attachment.getText()));
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Get Attachments from Order Lines

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String poNumber = header.getAttribute("CustomerPONumber");
```

```
if (poNumber == null) return;
if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();
//messages.add(new Message( Message.MessageType.ERROR, "Enter Attachment On Save"));
//messages.add(new Message( Message.MessageType.ERROR, "CustomerPONumber: " + poNumber));
Long headerId = header.getAttribute("HeaderId");
messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + headerId));

def lines = header.getAttribute("Lines");

while (lines.hasNext()) {

 def line = lines.next();
 def attachments = line.getAttachments();

 for (int i = 0; i < attachments.size(); ++i) {
 def attachment = attachments[i];
 messages.add(new Message(Message.MessageType.ERROR, "Pk1value:" + attachment.getPk1Value()));
 messages.add(new Message(Message.MessageType.ERROR, "Entityname:" + attachment.getEntityName()));
 messages.add(new Message(Message.MessageType.ERROR, "DatatypeCode:" + attachment.getDatatypeCode()));
 messages.add(new Message(Message.MessageType.ERROR, "Text:" + attachment.getText()));
 messages.add(new Message(Message.MessageType.ERROR, "Title:" + attachment.getTitle()));
 messages.add(new Message(Message.MessageType.ERROR, "Description:" + attachment.getDescription()));
 messages.add(new Message(Message.MessageType.ERROR, "FileName:" + attachment.getFileName()));
 messages.add(new Message(Message.MessageType.ERROR, "Url:" + attachment.getUrl()));
 messages.add(new Message(Message.MessageType.ERROR, "Url:" + attachment.getCategoryName()));
 }
}
ValidationException ex = new ValidationException(messages);
throw ex;
```

Add Attachment to Order Header

The example adds an attachment to an order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Attachment;
import oracle.jbo.domain.BlobDomain;
String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("WriteAttachmentsHeaderOnEndSubmit_run_extension")) return;

List < Message > messages = new ArrayList < Message > ();
//messages.add(new Message( Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));

////create header attachments
//Create TEXT attachment
Attachment newAttachment1 = new Attachment();
newAttachment1.setDatatypeCode("TEXT");
newAttachment1.setTitle("This is a text type title on submit");
newAttachment1.setText("this is some text");
newAttachment1.setDescription("this is some description");
header.createAttachment(newAttachment1);

//Create URL attachment
Attachment newAttachment2 = new Attachment();
newAttachment2.setDatatypeCode("WEB_PAGE");
newAttachment2.setTitle("This URL points to google server on submit");
newAttachment2.setUrl("http://www.google.com/");
newAttachment2.setDescription("Used for searching stuffs");
header.createAttachment(newAttachment2);

//Create a file attachment
```

```
Attachment newAttachment3 = new Attachment();
newAttachment3.setDatatypeCode("FILE");
newAttachment3.setTitle("file type title");
newAttachment3.setDescription("this file contains some random data on submit");
newAttachment3.setFileName("APITextFile.txt");
//newAttachment3.setFileContentType("application/text");
newAttachment3.setFileContentType("text/plain");
//newAttachment3.setFileContent(new BlobDomain("This is a test creation using Bytes".getBytes()));
newAttachment3.setFileContent("This is a test for creation file from api using bytes".getBytes());
header.createAttachment(newAttachment3);

/*delete all header attachments
//messages.add(new Message( Message.MessageType.ERROR, "Delete all header attachments"));
def attachments1 = header.getAttachments();
for (int i; i< attachments1.size(); ++i) {
def attachment1 = attachments1[i];
//header.deleteAttachment(attachment1);
}*/

ValidationException ex = new ValidationException(messages);
throw ex;
```

Delete Attachment from Order Header

The example deletes an attachment from the order header.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Attachment;
import oracle.jbo.domain.BlobDomain;

String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("DeleteAttachmentsHeaderOnSubmit_run_extension")) return;

//delete all header attachments
//messages.add(new Message( Message.MessageType.ERROR, "Delete all header attachments"));
def attachments1 = header.getAttachments();

for (int i; i < attachments1.size(); ++i) {
 //throw new ValidationException("An order with the Purchase Order ddddd Number " + attachments1 + " already
 exists.");
 def attachment1 = attachments1[i];
 header.deleteAttachment(attachment1);
}
```

Add Attachment to Order Line

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Attachment;
import oracle.jbo.domain.BlobDomain;

String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("WriteAttachmentsLineOnEndSubmitOnly_run_extension")) return;

List < Message > messages = new ArrayList < Message > ();
//messages.add(new Message( Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));
def lines = header.getAttribute("Lines");

while (lines.hasNext()) {

 def line = lines.next();
```

```
 /////////////////create header attachments
 //Create TEXT attachment
 Attachment newAttachment1 = new Attachment();
 newAttachment1.setDatatypeCode("TEXT");
 newAttachment1.setTitle("This is a text type title");
 newAttachment1.setText("this is some text");
 newAttachment1.setDescription("this is some description");

 line.createAttachment(newAttachment1);

 //Create URL attachment
 Attachment newAttachment2 = new Attachment();
 newAttachment2.setDatatypeCode("WEB_PAGE");
 newAttachment2.setTitle("This URL points to google server");
 newAttachment2.setUrl("http://www.google.com/");
 newAttachment2.setDescription("Used for searching stuffs");
 line.createAttachment(newAttachment2);

 //Create a file attachment
 Attachment newAttachment3 = new Attachment();
 newAttachment3.setDatatypeCode("FILE");
 newAttachment3.setTitle("file type title");
 newAttachment3.setDescription("this file contains some random data");
 newAttachment3.setFileName("APITextFile.txt");
 //newAttachment3.setFileContentType("application/text");
 newAttachment3.setFileContentType("text/plain");
 //newAttachment3.setFileContent(new BlobDomain("This is a test creation using Bytes".getBytes()));
 newAttachment3.setFileContent("This is a test for creation file from api using bytes".getBytes());
 line.createAttachment(newAttachment3);

}
/*delete all header attachments
//messages.add(new Message( Message.MessageType.ERROR, "Delete all header attachments"));
def attachments1 = header.getAttachments();
for (int i; i< attachments1.size(); ++i) {
 def attachment1 = attachments1[i];
 //header.deleteAttachment(attachment1);
}*/

ValidationException ex = new ValidationException(messages);
throw ex;
```

Delete Attachment from Order Line

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Attachment;
import oracle.jbo.domain.BlobDomain;
String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("DeleteAttachmentsLineOnSubmitOnly_run_extension")) return;

List < Message > messages = new ArrayList < Message > ();
//messages.add(new Message( Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));
def lines = header.getAttribute("Lines");

while (lines.hasNext()) {

 def line = lines.next();

 //delete all attachments from the order header
 //messages.add(new Message( Message.MessageType.ERROR, "Delete all header attachments"));
 def attachments1 = line.getAttachments();
 for (int i; i < attachments1.size(); ++i) {
 def attachment1 = attachments1[i];
 line.deleteAttachment(attachment1);
```

```
 }

}
```

Extend Billing and Payment

Use an order management extensions to manage billing and payment data on your sales order.

Some of these examples test a value for the purchase order number on the order header. This test isolates the extension and prevents it from affecting other developers who might also be running test code. For details, see *Overview of Creating Order Management Extensions*.

Some examples include methods that you can use with an extensible flexfield, such as getOrCreateContextRow. For details about them, see *Guidelines for Setting Up Extensible Flexfields in Order Management*.

Get Details for Billing Plans from Order Lines

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def billingPlans = line.getAttribute("BillingPlans");
 while (billingPlans.hasNext()) {
 def billingPlan = billingPlans.next();
 messages.add(new Message(Message.MessageType.ERROR, "BillingPlanTypeCode is " +
 billingPlan.getAttribute("BillingPlanTypeCode")));
 messages.add(new Message(Message.MessageType.ERROR, "BillingPlanId is " +
 billingPlan.getAttribute("BillingPlanId")));
 messages.add(new Message(Message.MessageType.ERROR, "ObjectVersionNumber is " +
 billingPlan.getAttribute("ObjectVersionNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "BillingTrxDate is " +
 billingPlan.getAttribute("BillingTrxDate")));
 messages.add(new Message(Message.MessageType.ERROR, "CancellationEffectiveDate is " +
 billingPlan.getAttribute("CancellationEffectiveDate")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineId is " +
 billingPlan.getAttribute("FulfillLineId")));

 messages.add(new Message(Message.MessageType.ERROR, "OverridePeriod is " +
 billingPlan.getAttribute("OverridePeriod")));
 messages.add(new Message(Message.MessageType.ERROR, "OverridePeriodAmount is " +
 billingPlan.getAttribute("OverridePeriodAmount")));
 messages.add(new Message(Message.MessageType.ERROR, "PeriodicityCode is " +
 billingPlan.getAttribute("PeriodicityCode")));
 messages.add(new Message(Message.MessageType.ERROR, "PeriodicityName is " +
 billingPlan.getAttribute("PeriodicityName")));
 }
}

ValidationException ex = new ValidationException(messages);
throw ex;.
```

Get Details for Payment Methods from Order Lines

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
```

```
if (!"PMC TEST".equals(header.getAttribute("CustomerPONumber"))) return;

List < Message > messages = new ArrayList < Message > ();

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();

 def payments = line.getAttribute("Payments");

 messages.add(new Message(Message.MessageType.ERROR, "PaymentMethodCode is " + payments.hasNext()));

 while (payments.hasNext()) {
 def payment = payments.next();


 messages.add(new Message(Message.MessageType.ERROR, "PaymentMethodCode is " +
 payment.getAttribute("PaymentMethodCode")));
 messages.add(new Message(Message.MessageType.ERROR, "PaymentTransactionIdentifier is " +
 payment.getAttribute("PaymentTransactionIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "PaymentSetIdentifier is " +
 payment.getAttribute("PaymentSetIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "SourceTransactionPaymentIdentifier is " +
 payment.getAttribute("SourceTransactionPaymentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "ReceiptMethodId is " +
 payment.getAttribute("ReceiptMethodId")));
 messages.add(new Message(Message.MessageType.ERROR, "PaymentMethod is " +
 payment.getAttribute("PaymentMethod")));
 messages.add(new Message(Message.MessageType.ERROR, "PaymentTypeCode is " +
 payment.getAttribute("PaymentTypeCode")));
 messages.add(new Message(Message.MessageType.ERROR, "PaymentType is " +
 payment.getAttribute("PaymentType")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineId is " +
 payment.getAttribute("FulfillLineId")));
 messages.add(new Message(Message.MessageType.ERROR, "HeaderId is " + payment.getAttribute("HeaderId")));
 messages.add(new Message(Message.MessageType.ERROR, "ObjectVersionNumber is " +
 payment.getAttribute("ObjectVersionNumber")));

 messages.add(new Message(Message.MessageType.ERROR, "AuthorizationStatus is " +
 payment.getAttribute("AuthorizationStatus")));
 messages.add(new Message(Message.MessageType.ERROR, "InstrumentAssignmentIdentifier is " +
 payment.getAttribute("InstrumentAssignmentIdentifier")));
 }

}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Set the Billing Transaction ID

```
String orderTypeCode = header.getAttribute("TransactionTypeCode");

def billingTxnTypeId = null;

if( orderTypeCode.equals("STD") ) { // Get the order type.
 billingTxnTypeId = getBillingTxnTypeId("Credit Memo");
}
else if ( orderTypeCode.equals("MIX") ){
 billingTxnTypeId = getBillingTxnTypeId("Invoice");
}

def lines = header.getAttribute("Lines");// Get the lines for the row set.

while( lines.hasNext() ) {// If more order lines exist...
 def line = lines.next();
```

```
 line.setAttribute("BillingTransactionTypeIdentifier", billingTxnTypeId);
}

Long getBillingTxnTypeId(String billingTxnTypeName) {

 def txnTypePVO = context.getViewObject("oracle.apps.financials.receivables.publicView.TransactionTypePVO");

 // Create view criteria (where clause predicates)
 def vc = txnTypePVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("Name", billingTxnTypeName);


 // Execute the view object query to find a matching row
 def rowset = txnTypePVO.findByViewCriteriaWithBindVars(vc, 1, new String[0], new Object[0]);

 // check if we have a matching row
 def row = rowset.first();

 Long txnTypeId = (Long) row.getAttribute("CustTrxTypeSeqId");
 //header.setAttribute("ShippingInstructions", txnTypeId); // debug statement

 return txnTypeId;
}
```

Prevent Updates on the Billing Transaction Type

If Order Management already sent the order line to accounts receivable, then don't allow Order Management to update the billing transaction type on the order line.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def po = header.getAttribute("CustomerPONumber");
if (po == null || !"VS_TEST".equals(po))
 return;

def reference = header.getAttribute("ReferenceHeaderId");
if (reference == null) return;

previousLinesMap = [:];
IsBillingTrxTypeIdUpdateable(reference);

if(previousLinesMap.size() == 0) {
 return;
}


def billTrxTypeId = null;
def origBillTrxTypeId = null;
def refFlineId = null;

def lines = header.getAttribute("Lines");
def line = null;
while (lines.hasNext()) {

 line = lines.next();

 refFlineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");

 if (refFlineId != null && previousLinesMap.containsKey(refFlineId)) {
 billTrxTypeId = line.getAttribute("BillingTransactionTypeIdentifier");
 origBillTrxTypeId = previousLinesMap.get(refFlineId);
 if( (billTrxTypeId == null && origBillTrxTypeId == null) || (billTrxTypeId != null && origBillTrxTypeId !=
 null && billTrxTypeId == origBillTrxTypeId)) {
```

```
 continue;
 }
 throw new ValidationException("Billing Transaction Type (aka Receivable Transaction Type) can't be
 updated :: Line#" + line.getAttribute("DisplayLineNumber") + " [ NEW:" + billTrxTypeId + " - OLD:" +
 origBillTrxTypeId + " ]");
 }
}

void IsBillingTrxTypeIdUpdateable(def referenceHeaderId) {

 def statusesSet = ["AWAIT_BILLING", "BILLED", "CANCELED", "CLOSED"] as Set;

 def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 def vc = vo.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("FulfillLineHeaderId", referenceHeaderId);
 rowset = vo.findByViewCriteria(vc, -1);

 def line = null;
 def fldItr = null;
 def fld = null;
 while (rowset.hasNext()) {

 line = rowset.next();

 if (statusesSet.contains(line.getAttribute("FulfillLineStatusCode"))) {
 previousLinesMap.put(line.getAttribute("FulfillLineId"), line.getAttribute("FulfillLineBillingTrxTypeId"));
 continue;
 }

 fldItr = line.getAttribute("FulfillLineDetail");
 while(fldItr.hasNext()) {
 fld = fldItr.next();
 if("Invoice".equals(fld.getAttribute("FulfillLineDetailTaskType"))) {
 previousLinesMap.put(line.getAttribute("FulfillLineId"), line.getAttribute("FulfillLineBillingTrxTypeId"));
 break;
 }
 }
 }
}
```

Set the Default Value for the Payment Term

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

List < Message > messages = new ArrayList < Message > ();
def logger = context.getLogger();

//def PName = header.getAttribute("BillToCustomerName");
//def custActId = header.getAttribute("BillToCustomerIdentifier");
//def siteUseId = header.getAttribute("BillToAccountSiteUseIdentifier");

def HeaderPayTerm = header.getAttribute("PaymentTermCode");
def lines = header.getAttribute("Lines");
if (!lines.hasNext()){
 throw new ValidationException("More time is needed to process the order");
}

while (lines.hasNext()) {

 def line = lines.next();
 def custActId = line.getAttribute("BillToCustomerIdentifier");
 def siteUseId = line.getAttribute("BillToAccountSiteUseIdentifier");
 def linePayTerm = line.getAttribute("PaymentTermCode");
 def linetransactioncode = line.getAttribute("TransactionCategoryCode");
```

```
 //def termId = getBillToPartyIdTermId(PName);
 def termId = getTermID(custActId, siteUseId);
 def termName = gettermName(termId);
 logger.logSevere("Term Name TrxPaymentTermPVO : ", termName);
 if (HeaderPayTerm == null && linePayTerm == null && linetransactioncode != 'RETURN') {
 if (termId != null){
 header.setAttribute("PaymentTerm", termName);
 line.setAttribute("PaymentTerm", termName);
 }
 }
}

Object getTermID(Long custActId, Long siteUseId) {
 def termId;
 def logger = context.getLogger();
 def custProfilePVO =
 context.getViewObject("oracle.apps.financials.receivables.publicView.analytics.CustomerFinancialProfilePVO");
 def vc1 = custProfilePVO.createViewCriteria();
 def vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("CustProfileCustAccountId", custActId);
 vcrow1.setAttribute("CustProfileSiteUseId", siteUseId);
 vcrow1.setAttribute("CustProfileStandardTerms", "> -1");

 def rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
 def profile = rowset1.first();
 if (profile != null) {

 termId = profile.getAttribute("CustProfileStandardTerms");
 header.setAttribute("ShippingInstructions", "TERM=" + termId);
 logger.logSevere("Term Id from CustomerFinancialProfilesPVO : ", termId);
 }
 else {
 vc1 = custProfilePVO.createViewCriteria();
 vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("CustProfileCustAccountId", custActId);
 vcrow1.setAttribute("CustProfileSiteUseId", "is null");
 vcrow1.setAttribute("CustProfileStandardTerms", "> -1");

 rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
 profile = rowset1.first();
 if (profile != null) {

 termId = profile.getAttribute("CustProfileStandardTerms");
 header.setAttribute("ShippingInstructions", "TERM=" + termId);
 logger.logSevere("Term Id from CustomerFinancialProfilesPVO : ", termId);
 }
 else {
 header.setAttribute("ShippingInstructions", termId);
 throw new ValidationException("No Payment Term found for the BillToCustomer 2 :: "+siteUseId);
 }
 }
 return termId;
}

Object gettermName(Long termID) {
 def raTermPVO =
 context.getViewObject("oracle.apps.financials.receivables.publicView.TrxPaymentTermPVO");
 def vc2 = raTermPVO.createViewCriteria();
 def vcrow2 = vc2.createViewCriteriaRow();
 vcrow2.setAttribute("TermId", termID);
 def rowset2 = raTermPVO.findByViewCriteria(vc2, -1);
 def termName = rowset2.first();
 def paymentName = termName.getAttribute("Name");
 header.setAttribute("PackingInstructions", paymentName);
 return paymentName;
```

```
}
```

Set the Default Value for the Payment Term Before You Submit the Sales Order

```
//preSubmit_PAYMENT_TERM_DEFAULTING//
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;


//Return if not a test order based on PO number if
(!"TERMS".equals(header.getAttribute("CustomerPONumber")))
return;
List < Message > messages = new ArrayList <
Message > (); def logger = context.getLogger();
//def PName =
header.getAttribute("BillToCustomerName");
//def custActId = header.getAttribute("BillToCustomerIdentifier");
//def siteUseId =
header.getAttribute("BillToAccountSiteUseIdentifier");
def HeaderPayTerm =
header.getAttribute("PaymentTermCode");

def lines = header.getAttribute("Lines"); if
(!lines.hasNext()){ throw new ValidationException("More time is needed to
process the order"); } while (lines.hasNext()) {
def line = lines.next();


def custActId =
line.getAttribute("BillToCustomerIdentifier");

def siteUseId =
line.getAttribute("BillToAccountSiteUseIdentifier");

def linePayTerm =
line.getAttribute("PaymentTermCode");

def linetransactioncode =
line.getAttribute("TransactionCategoryCode");


//def termId = getBillToPartyIdTermId(PName);
def termId = getTermID(custActId, siteUseId);
def termName = gettermName(termId);
logger.logSevere("Term Name TrxPaymentTermPVO : ", termName);
if(HeaderPayTerm == null && linePayTerm == null && linetransactioncode !='RETURN') {

 if (termId != null){
 header.setAttribute("PaymentTerm", termName);
 line.setAttribute("PaymentTerm", termName);
 }
 }
}

Object getTermID(Long custActId, Long siteUseId) { def
termId; def logger = context.getLogger(); def custProfilePVO =
context.getViewObject("oracle.apps.financials.receivables.publicView.analytics.CustomerFinancialProfilesPVO");

def vc1 = custProfilePVO.createViewCriteria();

def vcrow1 = vc1.createViewCriteriaRow();
vcrow1.setAttribute("CustProfileCustAccountId", custActId);
//vcrow1.setAttribute("CustProfileSiteUseId", siteUseId);
vcrow1.setAttribute("CustProfileStandardTerms", "> -1");

//vcrow1.setAttribute("PartyId", partyId); def
```

```
rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
def profile =rowset1.first();
 if (profile != null) {
 termId =profile.getAttribute("CustProfileStandardTerms");
 logger.logSevere("Term Id from CustomerFinancialProfilesPVO: ", termId);
 }
 else throw new ValidationException("No Payment Term found for the BillToCustomer :: "+custActId); return
 termId; }

Object gettermName(Long termID) {
def raTermPVO = context.getViewObject("oracle.apps.financials.receivables.publicView.TrxPaymentTermPVO");
def vc2 = raTermPVO.createViewCriteria(); def vcrow2 =
vc2.createViewCriteriaRow(); vcrow2.setAttribute("TermId", termID);
def rowset2 = raTermPVO.findByViewCriteria(vc2, -1); def termName =
rowset2.first(); def paymentName = termName.getAttribute("Name")
return paymentName;
}
```

Set the Default Value of the Payment Term According to CustAccountId and Date Range

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

List < Message > messages = new ArrayList < Message > ();
def logger = context.getLogger();

//def PName = header.getAttribute("BillToCustomerName");
//def custActId = header.getAttribute("BillToCustomerIdentifier");
//def siteUseId = header.getAttribute("BillToAccountSiteUseIdentifier");

def HeaderPayTerm = header.getAttribute("PaymentTermCode");
def lines = header.getAttribute("Lines");
if (!lines.hasNext()) {
 throw new ValidationException("More time is needed to process the order");
 //header.setAttribute("ShippingInstructions","More time is needed to process the order");
}


while (lines.hasNext()) {

 def line = lines.next();
 def custActId = line.getAttribute("BillToCustomerIdentifier");
 def siteUseId = line.getAttribute("BillToAccountSiteUseIdentifier");
 def linePayTerm = line.getAttribute("PaymentTermCode");
 def linetransactioncode = line.getAttribute("TransactionCategoryCode");

 //def termId = getBillToPartyIdTermId(PName);
 def termId = getTermID(custActId, siteUseId);
 def termName = gettermName(termId);
 logger.logSevere("Term Name TrxPaymentTermPVO : ", termName);
 if (HeaderPayTerm == null && linePayTerm == null && linetransactioncode != 'RETURN') {
 if (termId != null) {
 header.setAttribute("PaymentTerm", termName);
 line.setAttribute("PaymentTerm", termName);
 }
 }
}


Object getTermID(Long custActId, Long siteUseId) {
 def termId;
 def logger = context.getLogger();
 def custProfilePVO =
 context.getViewObject("oracle.apps.financials.receivables.publicView.analytics.CustomerPrfilePVO");

 java.sql.Date sqlDate = new java.sql.Date((new Date()).getTime());
```

```
 def vc1 = custProfilePVO.createViewCriteria();
 def vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("CustomerProfileCustAccountId", custActId);
 vcrow1.setAttribute("CustomerProfileSiteUseId", siteUseId);
 vcrow1.setAttribute("CustomerProfileStandardTerms", "> -1");
 vcrow1.setAttribute("CustomerProfileEffectiveEndDate", ">= "+sqlDate);
 vcrow1.setAttribute("CustomerProfileEffectiveStartDate", " <= "+sqlDate);

 def rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
 def profile = rowset1.first();
 if (profile != null) {

 termId = profile.getAttribute("CustomerProfileStandardTerms");
 //header.setAttribute("ShippingInstructions", "TERM=" + termId);
 logger.logSevere("Term Id from CustomerFinancialProfilesPVO : ", termId);
 } else {
 vc1 = custProfilePVO.createViewCriteria();
 vcrow1 = vc1.createViewCriteriaRow();
 vcrow1.setAttribute("CustomerProfileCustAccountId", custActId);
 vcrow1.setAttribute("CustomerProfileSiteUseId", "is null");
 vcrow1.setAttribute("CustomerProfileStandardTerms", "> -1");
 vcrow1.setAttribute("CustomerProfileEffectiveEndDate", ">= "+sqlDate);
 vcrow1.setAttribute("CustomerProfileEffectiveStartDate", " <= "+sqlDate);

 rowset1 = custProfilePVO.findByViewCriteria(vc1, -1);
 profile = rowset1.first();
 if (profile != null) {

 termId = profile.getAttribute("CustomerProfileStandardTerms");
 //header.setAttribute("ShippingInstructions", "TERM=" + termId);
 logger.logSevere("Term Id from CustomerFinancialProfilesPVO : ", termId);
 } else {
 //header.setAttribute("ShippingInstructions", termId);
 throw new ValidationException("No Payment Term found for the BillToCustomer 2 :: " + siteUseId);
 }
 }
 return termId;
}

Object gettermName(Long termID) {
 def raTermPVO =
 context.getViewObject("oracle.apps.financials.receivables.publicView.TrxPaymentTermPVO");
 def vc2 = raTermPVO.createViewCriteria();
 def vcrow2 = vc2.createViewCriteriaRow();
 vcrow2.setAttribute("TermId", termID);
 def rowset2 = raTermPVO.findByViewCriteria(vc2, -1);
 def termName = rowset2.first();
 def paymentName = termName.getAttribute("Name");
 header.setAttribute("PackingInstructions", paymentName);
 return paymentName;
}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Guidelines for Setting Up Extensible Flexfields in Order Management

Extend Shipping

Use these code examples to help you create order management extensions that manage shipping.

Some of these examples test a value for the purchase order number on the order header. This test isolates the extension and prevents it from affecting other developers who might also be running test code. For details, see *Overview of Creating Order Management Extensions*.

Validate Ship-to, Bill-to, and Sold-to Attributes

Get values for the Ship-to, Bill-to, and Sold-to attributes from the order header. Compare them to the customer account details and validate that they're all for the same customer.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber == null) return;

if (!poNumber.startsWith("PMC TEST")) return;

List < Message > messages = new ArrayList < Message > ();

boolean relationExists = false;

def soldTo = header.getAttribute("BuyingPartyIdentifier");
def shipTo = header.getAttribute("ShipToPartyIdentifier");
def billTo = header.getAttribute("BillToCustomerIdentifier");

messages.add(new Message(Message.MessageType.ERROR, "Sold to is " + soldTo));
messages.add(new Message(Message.MessageType.ERROR, "Ship to is " + shipTo));
messages.add(new Message(Message.MessageType.ERROR, "Bill to is " + billTo));

if (header.getAttribute("BuyingPartyName") == header.getAttribute("BillToCustomerName"))
 relationExists = true;

messages.add(new Message(Message.MessageType.ERROR, "Buying Party Name is " +
 (header.getAttribute("BuyingPartyName"))));
messages.add(new Message(Message.MessageType.ERROR, "Bill to Customer Name is " +
 header.getAttribute("BillToCustomerName")));


def CustPVO =
 context.getViewObject("oracle.apps.cdm.foundation.parties.publicView.customerAccounts.CustomerAccountRelationshipPV
def vc = CustPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("RelatedCustAccountId", soldTo);
def rowSet = CustPVO.findByViewCriteria(vc, -1);

messages.add(new Message(Message.MessageType.ERROR, "Found a row: " + rowSet.hasNext()));

while (rowSet.hasNext()) {
 def row = rowSet.next();
 def billtorelation = row.getAttribute("CustAccountId");
 if (billtorelation == billTo) {
 relationExists = true;
 }
}

//header.setAttribute("ShippingInstructions", header.getAttribute("ShippingInstructions") + ", " +
 relationExists);
```

```
/*
if( !relationExists) {

 messages.add(new Message( "Bill To Customer is not related with Sold To"));
}
*/
ValidationException ex = new ValidationException(messages);
throw ex;
```

Set the Default Value for Ship-To Address According to Business Unit

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

List<Message> messages = new ArrayList<Message>();

def buNumber = header.getAttribute("BusinessUnitIdentifier");

def lines = header.getAttribute("Lines");
if(!lines.hasNext()){
 messages.add(new Message( Message.MessageType.ERROR,"No lines available"));
}
while( lines.hasNext() ) {
 def line = lines.next();
//Add more business units according to your requirements.
 if(buNumber==204){
 setShipToPartySiteId(line,messages,1220);//Have to pass the ShipToPartySiteId
 }
 if(buNumber==500){
 setShipToPartySiteId(line,messages,1222);
 }
}//end of while

if(!messages.isEmpty()) {
 ValidationException ex = new ValidationException(messages);
 throw ex;
 }


def setShipToPartySiteId(def line,List<Message> messages, def partySiteId){
 def partySiteUsgFlag = getPartySiteUssage(partySiteId);
 if(!partySiteUsgFlag){
 messages.add(new Message( Message.MessageType.ERROR,"The Ship-to Address isn't valid. Examine your
 setup."));
 return null;
 }//shipBeginDate null check
 else{
 header.setAttribute("ShipToPartySiteIdentifier",partySiteId);
 line.setAttribute("ShipToPartySiteIdentifier",partySiteId);
 }
 return null;
}

def getPartySiteUssage(Long partySiteId) {
def PartySiteUsePVO =
 context.getViewObject("oracle.apps.cdm.foundation.parties.publicView.analytics.PartySiteUsePVO");
 def vc = PartySiteUsePVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();

 vcrow.setAttribute("PartySiteId", partySiteId);
 vcrow.setAttribute("SiteUseType", "SHIP_TO");
 def rowset = PartySiteUsePVO.findByViewCriteria(vc, -1);
 if(rowset.hasNext()){
 def row = rowset.next();
 return true;
 }
```

```
 else
 return null;
}
```

Cascade the Shipping Method from the Order Header to Order Lines

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
def poNumber = header.getAttribute("CustomerPONumber");
List<Message> messages = new ArrayList<Message>();

if(poNumber != "shipmethod") return;
def ServiceCode=header.getAttribute("ShipClassOfServiceCode");
def Carrier=header.getAttribute("CarrierId");
def ShipModeOfTransport=header.getAttribute("ShipModeOfTransportCode");

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 line.setAttribute("ShippingInstructions","ship_method_test");
 line.setAttribute("ShippingCarrierCode",Carrier);
 line.setAttribute("ShippingServiceLevelCode",ServiceCode);
 line.setAttribute("ShippingModeCode",ShipModeOfTransport);
}
```

Populate Shipping Instructions on Order Lines

Run this extension when the original order doesn't contain shipping instructions and you're revising the order.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");
def orderNumber = header.getAttribute("SourceTransactionNumber");

if(orderNumber != "BMIQ-05022018-248408") return;
if(changeVN < 2) return;

def lines = header.getAttribute("Lines");
 while (lines.hasNext()) {
 def line = lines.next();
 def lineStatus = line.getAttribute("StatusCode");
 def shipInst = line.getAttribute("ShippingInstructions");
 if("CLOSED".equals(lineStatus) && shipInst != null){
 def refFlineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");
 if(refFlineId == null){
 continue;
 }
 def orginalFlinePRow = getFLineForRefFLine(refFlineId);
 if(orginalFlinePRow!=null) {
 def lineStatus1 = orginalFlinePRow.getAttribute("FulfillLineStatusCode");
 def shipInst1 = orginalFlinePRow.getAttribute("FulfillLineShippingInstructions");
 if("CLOSED".equals(lineStatus1) && shipInst1 == null){
 line.setAttribute("ShippingInstructions", null);
 }
 }
 }
}

def Object getFLineForRefFLine(Long refFlineId) {
def vo = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 Object[] rows = vo.findByKey(refFlineId, 1);
 def originalFLine = rows[0];

 return originalFLine;
}
```

Make Sure Site Usage is Ship To

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");

if(poNumber != null && poNumber.equals("ship")) {

def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();
 def partySiteId = line.getAttribute("ShipToPartySiteIdentifier");
 //throw new ValidationException("partySiteUseId :"+partySiteUseId);

 if(!(getPartySiteUssage(partySiteId)))
 throw new ValidationException("siteUssage is not shipto");
 }

}

Boolean getPartySiteUssage(Long partySiteId) {
def PartySiteUsePVO =
 context.getViewObject("oracle.apps.cdm.foundation.parties.publicView.analytics.PartySiteUsePVO");
 def vc = PartySiteUsePVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();

 vcrow.setAttribute("PartySiteId", partySiteId);
 vcrow.setAttribute("SiteUseType", "SHIP_TO");
 def rowset = PartySiteUsePVO.findByViewCriteria(vc, -1);

 if(rowset.hasNext())
 return true;
 else
 return false;

}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

Extend Extensible Flexfields

Use order management extensions with extensible flexfields as a handy and powerful way to customize your deployment.

Access and Update Extensible Flexfields

You can use any extension point to access or update an extensible flexfield that's on the order header or the order line. To access a flexfield segment, your extension creates a row for a flexfield context. You can then use the getAttribute method and the setAttribute method to read and write to the segment.

For more, see *Use Order Management Extensions to Get Values from Extensible Flexfields*.

Take Action According to the Value of an Extensible Flexfield

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
```

```
def poNumber = header.getAttribute("CustomerPONumber");

if(poNumber != null && poNumber.equals("test") ){
def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
def line = lines.next();
def context = line.getOrCreateContextRow("VS_Context");
 def effVal = context.getAttribute("eligibleforprime");

// throw new ValidationException("Eff value effVal:: "+effVal);
 if(effVal.equals("Y")){
 line.setAttribute("OrderedQuantity",0);
 }

}
}
```

Copy Extensible Flexfield Values to Data Values

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");

if(poNumber != null && poNumber.equals("TEST_CO2") ){
def docRefs = header.getAttribute("DocumentReferences");
 if (!docRefs.hasNext()){
 throw new ValidationException("We need more time to get the document reference.");
}
while(docRefs.hasNext()) {
def docRef = docRefs.next();
String docRefType = docRef.getAttribute("DocumentReferenceType");
if(!'COPY_REF_ORDER'.equals(docRefType))continue;
def cntxRow = header.getOrCreateContextRow("HeaderContext1");
cntxRow.setAttribute("_H1AttributeChar2",docRef.getAttribute("DocumentNumber"));
}
}
```

Update the Ship-To Address

This example updates the ship-to address. If the customer relationship type is Single, then set the value of the Ship-To Address according to the value of an extensible flexfield.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

List < Message > messages = new ArrayList < Message > ();
def po = header.getAttribute("CustomerPONumber");
//Condition that specifies when to call the extension. You can can also use an extensible flexfield on the
 order header.
if (!"TEST_MDF".equals(po))
 return;

def line2AddrMap = [: ];
def lines = header.getAttribute("Lines");

if (!lines.hasNext()) {
 throw new ValidationException("Order Management hasn't saved the order line details. Save the sales order,
 then try again.");
}
while (lines.hasNext()) {
 def line = lines.next();
 def lineNo = line.getAttribute("DisplayLineNumber");
 line2AddrMap.put(lineNo, line.getAttribute("ShipToPartySiteIdentifier"));
}

lines.reset();
```

```
while (lines.hasNext()) {
 def line = lines.next();
 def context = line.getContextRow("FulfillLineContext1"); //Add your own extensible flexfield context that
 identifies the address for the ship-to line that you must copy.
 def copyFromLineNo = context != null ? context.getAttribute("_FL1AttributeChar1") : null; //Update this
 value with the display line number of the source line.
 if (copyFromLineNo != null && !copyFromLineNo.isEmpty()) {
 line.setAttribute("ShipToPartySiteIdentifier", line2AddrMap.get(copyFromLineNo));
 }
}

if (!messages.isEmpty()) {
 ValidationException ex = new ValidationException(messages);
 throw ex;
}
```

More

Topic	Description
Use Extensions to Get Data from Oracle Applications	Set the value in a context segment.
Update the Order's Submit Date	Use an extensible flexfield to update the order submit date.
Use Extensions to Cancel Order Lines	Use an extensible flexfield to cancel order lines in a sales order.
See the Get Extensible Flexfields from Original Sales Order subtopic in *Extend Return Orders*.	Get extensible flexfields from the order line of the original order, then copy it to a return order line,
Extend Order Headers	See these subtopics: • Set Extensible Flexfield On Order Header. Set an attribute to the current date and time. Use an extensible flexfield on the order header. • Set Extensible Flexfield Values for Hazardous Items. If the HazardousMaterialFlag attribute equals Y, then set the value for attributes that use extensible flexfields on the order header and order lines.
Extend Return Orders	See the Copy Extensible Flexfield from Original Order to an RMA subtopic. Copy extensible flexfield data from the original order to a return order that includes a return material authorization (RMA).

Verify Data That Users Enter

Create an order management extension that determines whether a purchase order exists for the purchase order number that the Order Entry Specialist enters in the Purchase Order attribute.

It calls a public view object to get data from Oracle Procurement.

For demonstration purposes, this example hard codes some values, such as HW INTERNAL. Your environment will likely require different variable values.

This topic uses example values. You might need different values, depending on your business requirements.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Extensions
2. On the Manage Order Management Extensions page, on the On Start of Submission Request tab, create a new extension.

Attribute	Value
Name	Verify That the Purchase Order Exists
Description	Determine whether a purchase order exists for the purchase order number that the Order Entry Specialist enters in the Purchase Order attribute.

3. In the Definition area, add code.

```
//Import classes for validation exceptions and messages from Oracle Trading Community Architecture.
import oracle.apps.scm.doo.common.extensions.ValidationException;

def orderType = header.getAttribute("TransactionTypeCode");

//Determine whether the sales order is internal.
if(orderType != null && orderType.contains("HW INTERNAL")) {

 //Determine whether the purchase order exists.
 String poNumber = header.getAttribute("CustomerPONumber");

 boolean poExists = false;
 if( poNumber != null ) {

//Get the PVO you need to access purchase orders.
def poPVO = context.getViewObject("oracle.apps.prc.po.publicView.PurchasingDocumentHeaderPVO");

//Create the view criteria. Use where clause predicates.
def vc = poPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("Segment1", poNumber);

//Query the view object to find a matching row.
def rowset = poPVO.findByViewCriteriaWithBindVars(vc, 1, new String [0], new String [0]);

//Determine whether a matching row exists.
poExists = rowset.hasNext();
}

//If a matching row does not exist, then the purchase order that the user entered does not exist. Create
 a validation error and stop the sales order submit.
 if( !poExists ) {
throw new ValidationException("ORA_MANAGE_EXTENSIONS", "DOO_EXT_HW_INTERNAL_PO_REQD", null);
 }
}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Update the Order's Submit Date

Create an extension that uses an extensible flexfield to update the order submit date.

This topic uses example values. You might need different values, depending on your business requirements.

1. Create an extensible flexfield.

Attribute	Value
Name	H1AttributeDateTime1
Category	Additional Header Information
Context	HeaderContext1

   For details about how to create an extensible flexfield, see *Set Up Extensible Flexfields in Order Management*.
2. In the Setup and Maintenance work area, go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Management Extensions
3. On the Manage Order Management Extensions page, notice the elements.
   - Three tabs, one for each extension point.
   - A Sequence column. Order Management runs the extensions for each extension point sequentially according to the values in the Sequence column. For example, if the first row contains a value of 10, the second row a value of 30, and the third row a value of 20, then Order Management will run row one, row three, and then row two.
4. On the Manage Order Management Extensions page, click **Actions > Create New**.
5. In the Create Extension dialog, accept the default values, then click **OK**.

   The Use Existing option creates a reference to an extension that already exists. It doesn't create a copy. For example, assume an extension named My_Extension already exists on the On Save extension point. Assume you set Use Existing to My_Extension when you create an extension on the On Start of Submission Request extension point. If you modify My_Extension from On Start of Submission Request at any point in the future, then Order Management will automatically update My_Extension for On Save with your modification. If

you modify My_Extension for On Save, then Order Management will update My_Extension for On Start of Submission Request.

6. On the Create Extension page, on the On Start of Submission Request tab, enter values.

Attribute	Value
Name	Update Order Submit Date
Description	This extension sets the value of an extensible flexfield to the date and time when Order Management submitted the sales order.

7. In the Definition area, add code.

```
// Make sure the extension runs only for your test sales order. If more than one developers use your
 test environment, then this condition makes sure the code updates only your sales order. You must
 remove this condition in a production environment.
def poNumber = header.getAttribute("CustomerPONumber");
if( !"UpdateOrderSubmissionDate".equals(poNumber) ) return;

// Get the current time, and then create an instance of java.sql.Date, and set it to the current time.
long currentTime = new Date().getTime();
def date = new java.sql.Date(currentTime);

// Get the row for the flexfield context named HeaderContext1.
def flexfieldContext = header.getOrCreateContextRow("HeaderContext1");

// Set the date on the attribute named _H1AttributeDateTime1 to the current date. Use flexfieldContext
 to identify the flexfield context where _H1AttributeDateTime1 resides, which is HeaderContext1.
flexfieldContext.setAttribute("_H1AttributeDateTime1", date);
```

8. Click **Validate > Save and Close**.

Test Your Set Up

1. Navigate to the Order Management work area, create a new sales order, then click **Submit**.

Attribute	Value
Purchase Order	UpdateOrderSubmitDate_run_extension

2. Click **Actions > View Additional Information**.
3. In the Additional Information dialog, click **HeaderEFFDetails**, then verify that the dialog displays the value.

Attribute	Value
H1AttributeDateTime1	The current date and time. For example: 04/01/17 6:25 PM

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

Validate Relationships Between Attributes

Create an order management extension that makes sure the sales order includes a relationship between the sold-to customer and the ship-to customer, and between the sold-to customer and bill-to customer.

If one of these relationships doesn't exist when the user attempts to submit the sales order, then the extension stops the sales order from proceeding and displays an error message.

This topic uses example values. You might need different values, depending on your business requirements.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Management Extensions
2. On the Manage Order Management Extensions page, on the On Start of Submission Request tab, create a new extension.

Attribute	Value
Name	Validate Relationships Between Attributes
Description	Make sure a relationship exists between the sold-to customer and the ship-to customer, and between the sold-to customer and the bill-to customer. If one of these relationships doesn't exist, then create an error and stop the sales order from proceeding.

3. In the Definition area, add code.

```
//import classes for validation exceptions and messages from Oracle Trading Community Architecture.
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scom.doo.comm.extensions.Message;

// Make sure the extension runs only for your test sales order. If more than one developer uses your
 test environment, then this condition makes sure the code updates only your sales order. You must
 remove this condition in a production environment.
def poNumber = header.getAttribute("CustomerPONumber" );
if(poNumber == null) return;
if( !poNumber.startsWith("MatchRelationship") ) return;
boolean relationExists=false;

//define the variables you will use to store the identifiers.
def soldTo = header.getAttribute("BuyingPartyIdentifier")
def billTo = header.getAttribute("BillToCustomerIdentifier")
```

```
//if the relationship exists, then further validation is not necessary, so save the sales order, and
 then exit.
if (header.getAttribute("BuyingPartyName")==header.getAttribute("BillToCustomerName"))
 relationExists=true;

//determine what relationship currently exists between the bill-to customer and the sold-to customer.
//reference the view object that stores the relationship. In this example, the
 CustomerAccountRelationship view object stores this relationship.
def CustPVO =
 context.getViewObject("oracle.apps.cdm.foundation.parties.publicView.customerAccounts.CustomerAccountRelationsh

//create the view criteria.
def vc = CustPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
//RelatedCustAccountId is an attribute that stores the sold-to relationship, so you set vcrow to
 reference RelatedCustAccountId. You will examine it to determine whether the sold-to customer is
 related to the bill-to customer.
vcrow.setAttribute("RelatedCustAccountId", soldTo);
//Query the view object according to the criteria that you set in the previous line of code.
def rowSet = CustPVO.findByViewCriteria(vc, -1);

//Read through the row set. If a row exists that contains a relationship, then you have determined that
 a relationship exists between sold-to and bill-to, so save the sales order, and then exit.
while (rowSet.hasNext()) {
 def row = rowSet.next();
 def billtorelation=row.getAttribute("CustAccountId")
 if (billtorelation == billTo)
 {
 relationExists=true;
 }
}
//Create an exception when a relationship does not exist and display an error message.
//header.setAttribute("ShippingInstructions", header.getAttribute("ShippingInstructions") + ", "
 relationExists)

if( !relationExists) {
throw new ValidationException("The order submit failed because the bill-to customer is not related to
 the sold-to customer.");
}
```

This example is only for demonstration purposes in a development environment. It hard codes the message to display in the Order Management work area in the English language as `The order submit failed because the bill-to customer is not related to the sold-to customer`. To avoid problems with translation to other languages, don't code the message in a production environment.

Instead, here's some code you can use to reference a message from the messaging framework.

```
throw new ValidationException("lookup_code", "message name", token_values);
```

where

- **lookup_code** determines where and how to display the message in the Order Management work area. For example, you can reference more than one lookup code to display messages in different pie slices on the infolets in the Order Management work area, according to lookup code.
- **message name** identifies the name of the message that exists in the messaging framework.

- **token_values** specifies a list of the values to use in the tokens that the message contains. If the message doesn't contain any tokens, then use null.

For example, here's some code that displays the contents of the FOM_CMN_INV_BILL_TO message.

```
throw new ValidationException("ORA_MANAGE_EXTENSIONS", "FOM_CMN_INV_BILL_TO", null);
```

Learn how to use a lookup_code with the ValidationException method. For details, see *Methods That You Can Use with Order Management Extensions*.

Test Your Set Up

1. Navigate to the Order Management work area, create a new sales order, then click **Submit**.

Attribute	Value
Customer	Computer Service and Rentals
Bill-to Customer	Business World Inc.
Purchase Order	ValidateRelationshipsBetweenAttributes _run_extension

2. Verify that the error dialog displays the message you coded.

   Computer Service and Rentals is the sold-to customer, Business World Inc is the bill-to customer, they don't match, so Order Management displays an error.
3. Set the value, then click **Submit**.

Attribute	Value
Bill-to Customer	Computer Service and Rentals

4. Verify that Order Management submits the sales order and sets the status to Processing.

   Computer Service and Rentals is the sold-to customer and the bill-to customer, they match, so Order Management processes the sales order.

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Use Extensions to Cancel Order Lines

Create an extension that uses an extensible flexfield to cancel order lines in a sales order.

Assume your customer places a sales order for items you can't fulfill because the items have become obsolete and your supplier no longer supplies the item. You can use an order management extension to cancel lines that contain a value, such as Obsolete, in an extensible flexfield.

This topic uses example values. You might need different values, depending on your business requirements.

1. Create the extensible flexfield.
    - Add it to the Fulfillment Line Information flexfield. Use these values.

Attribute	Value
Name	CancelLine
Category	Additional Fulfillment Line Information
Context	ProductObsoleteContext

    - Set the values on the Create Segment page.

Attribute	Value
Name	CancelLine
Code	CancelLine
API Name	`cancelline`
Enabled	Contains a check mark.
Data type	Character
Table Column	ATTRIBUTE_CHAR2
Value Set	10 Characters
Prompt	Cancel Line

Attribute	Value
Display Type	Text Box

- Publish and deploy your flexfield.

For details, see *Set Up Extensible Flexfields in Order Management*.

2. Create your extension.
   - In the Setup and Maintenance work area, go to the task.
     - Offering: Order Management
     - Functional Area: Orders
     - Task: Manage Order Management Extensions
   - On the Create Extension page, on the On Start of Submission Request tab, enter values.

Attribute	Value
Name	Cancel Order Quantity
Description	This extension examines the value of an extensible flexfield on an order line to determine whether to cancel the line.

   - In the Definition area, add code.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");
if( !"CancelOrderQuantity".equals(poNumber) )
return;
def lines = header.getAttribute("Lines");
if(lines == null)
return;
while(lines.hasNext()){
 def line = lines.next();
 //Get the row for the flexfield context named ProductObsoleteContext.
 //Use the code value for the context as the argument to get the context row.
 def context = line.getContextRow("ProductObsoleteContext");
 if(context == null)
 throw new ValidationException("Context ProductObsoleteContext was null");
 def cancelLineEffVal = context.getAttribute("cancelline");
 if("Y".equals(cancelLineEffVal)){
 line.setAttribute("OrderedQuantity",0);
 }
}
```

   - Click **Validate > Save and Close**.

Test Your Setup

Create a test sales order for customer Computer Service and Rentals.

1. Create the sales order.
    - Go to the Order Management work area and create a new sales order.

      Set attributes on the order header.

Attribute	Value
Customer	Computer Service and Rentals
Order Type	Standard Orders

    - Add an order line that contains an item that isn't configured.
    - Click **Submit**.
2. Wait for the order line status to change to Awaiting Shipping. Click **Refresh** to update the status.
3. Revise your sales order.
    - On the Manage Orders page, click **Actions > Create Revision**
    - On the Create Revision page, set the attribute on the order header.

Attribute	Value
Purchase Order	CancelOrderQuantity

    - Click an **order line**, then click **Update Lines**.
    - On the Update Lines page, move the Additional Information attribute to the Selected window, then click **Next**.
    - In the Specify values area, click **Update Additional Information**.
    - In the dialog that opens, click **CancelLine**.
    - In the ProductObsoleteContext area, in the window next to CancelLine, enter `"Y"`.

      You must include the double quotation marks.
    - Click **OK**, then click **Update**.
4. Verify that Order Management changed the quantity on the order line to 0, and canceled the line.

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Set Up Extensible Flexfields in Order Management

External

Guidelines for Calling Web Services in Order Management Extensions

Apply these guidelines when your extension calls a web service.

Process Calls to Your Web Services

If you use the invokeSoapService method more than one time in your code, then make sure you finish processing the response from the first call before you make the second call.

Correct Code

Line	Code
1	`def serviceInvoker = context.getServiceInvoker();`
2	`def response1 = serviceInvoker.invokeSoapService("connector1", payload).getSoapBody();`
3	`String tagStorageLimitDays = response1.getElementsByTagNameNS("*", "tagStorageLimitDays").item(0)?.getTextContent()?: 0;`
4	`def response2 = serviceInvoker.invokeSoapService("webservice2", payload).getSoapBody();`
5	`String status = response2.getElementsByTagNameNS("*", "status").item(0)?.getTextContent()?: "failure";`

This code processes the response from webservice1 before it calls webservice2. This is how you should write your code.

- Line 2 calls web service webservice1 and line 3 processes the response from webservice1.
- Line 4 calls web service webservice2 and line 5 processes the response from web service webservice2.

Incorrect Code

Line	Code
1	`def serviceInvoker = context.getServiceInvoker();`

Line	Code
2	`def response1 = serviceInvoker.invokeSoapService("webservice1", payload).getSoapBody();`
3	`def response2 = serviceInvoker.invokeSoapService("webservice2", payload).getSoapBody();`
4	`String tagStorageLimitDays = response1.getElementsByTagNameNS("*", "tagStorageLimitDays").item(0)?.getTextContent()?: 0;`
5	`String status = response2.getElementsByTagNameNS("*", "status").item(0)?.getTextContent()?: "failure";`

This code will fail because line 3 calls webservice2 before line 4 has a chance to process the response from webservice1. This happens because the response from webservice1 is no longer available.

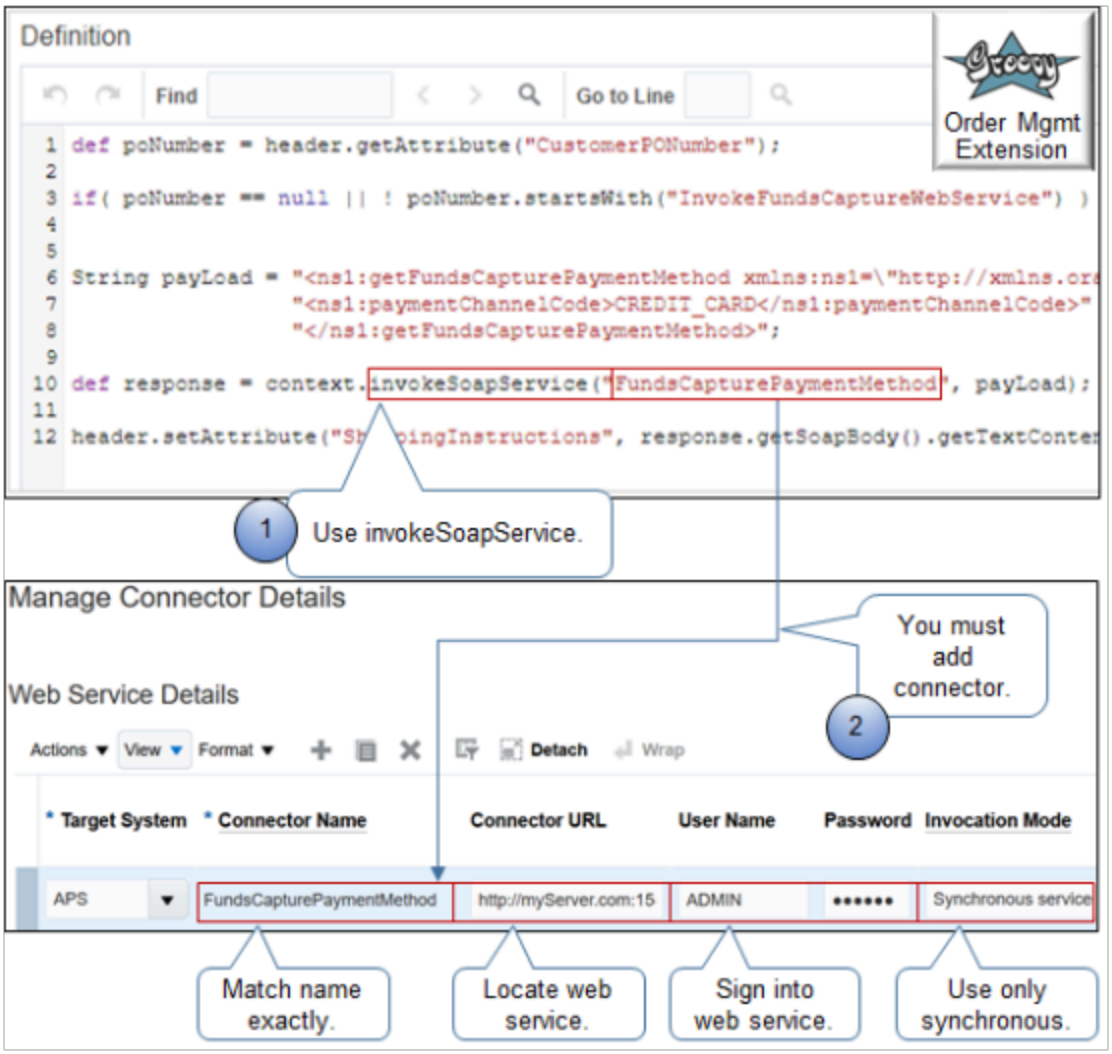
To avoid this problem, process the response from the first call before you make the second call.

As an alternative, you can add the extractContentAsDocument method to make sure the extension finishes extracting the response from the first call. For example:

Line	Code
1	`def serviceInvoker = context.getServiceInvoker();`
2	`def response1 = serviceInvoker.invokeSoapService("webservice1", payload).getSoapBody().extractContentAsDocument();`
3	`def response2 = serviceInvoker.invokeSoapService("webservice2", payload).getSoapBody();`
4	`String tagStorageLimitDays = response1.getElementsByTagNameNS("*", "tagStorageLimitDays").item(0)?.getTextContent()?: 0;`
5	`String status = response2.getElementsByTagNameNS("*", "status").item(0)?.getTextContent()?: "failure";`

Line 2 calls the extractContentAsDocument method in the SOAPBody object to make sure the extension finishes extracting the response from webservice1 before it calls webservice2.

Use Method and Create Connector



Note

1. Use the invokeSoapService method in your extension to call the web service.

   You can call an Oracle Application web service or some other service that resides outside of Oracle Applications. You can call only a SOAP service. You can't call REST API. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then click **Sales Orders for Order Hub**.

2. Set up the connector.

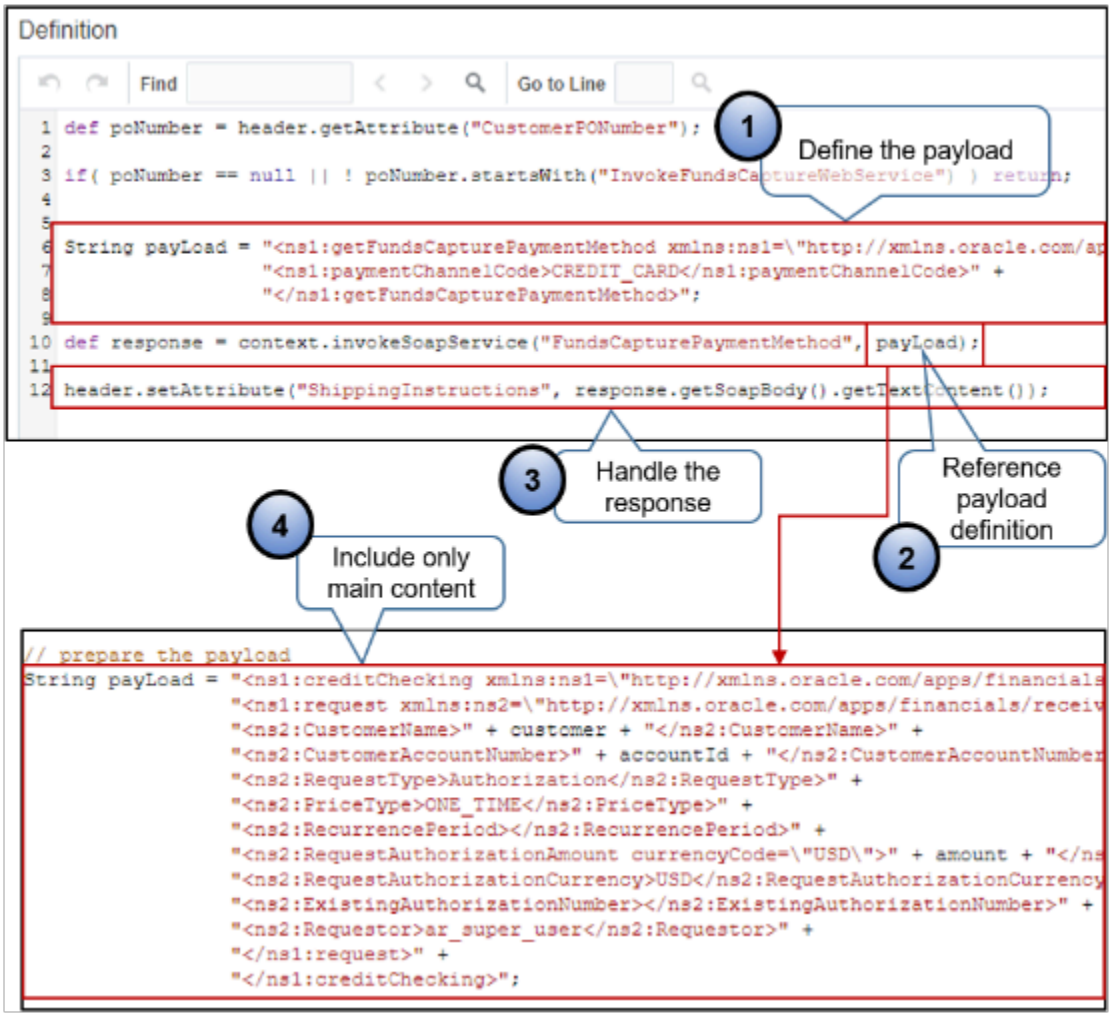
> **Note:** You must use the Manage Connector Details page to set up a connector for the web service that you reference in your extension.

Set these values.

Attribute	Value
Connector Name	Use the exact same name that you use in the extension. For example: `invokeSoapService("FundsCapturePaymentMethod")` For this example, set Connector Name to **`FundsCapturePaymentMethod`**.
Connector URL	Enter the URL that locates the web service address that you deploy on the system that resides outside of Order Management.
User Name	Enter the user name you use to sign into the web service.
Password	Enter the password you use to sign into the web service.
Invocation Mode	You must use Synchronous Service.

For details, see *Manage Connector Details Between Order Management and Your Fulfillment System*.

Create the Payload



Note

1. Create the payload. This example uses **`String payLoad`** to **`hard code`** the payload as a string. You can also use other techniques to meet your requirements, such as defining variables in the payload.
2. Reference your payload definition in a parameter of the invokeSoapService method. This example references string **`payLoad`**.
3. Use the getSoapBody method to handle the response that the web service sends in reply to your request. Your extension can receive the response, parse it, then extract the details it requires from the response.
4. Create your payload so it includes only the main content. Don't include envelope or body details. invokeSoapService expects only the main content. It will add envelope and body details.

Extract Details from Web Service Response

The getSoapBody() call on the response object in a web service response is an instance of Java class for a SOAP body (Simple Object Access Protocol). For details, see *Interface SOAPBody*. Use the methods that are available on these interfaces to extract details from a web service response.

This example order management extension uses methods from these interfaces to extract data from the response.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
```

```
def poNumber = header.getAttribute("CustomerPONumber");

if( poNumber != "CreditCheck" ) return;

// get attribute to populate in the payload
String customer = header.getAttribute("BillToCustomerName");
Long accountId = header.getAttribute("BillToCustomerIdentifier");
BigDecimal amount = new BigDecimal(1000);

// prepare the payload
String payLoad = "<ns1:creditChecking xmlns:ns1=\"http://xmlns.oracle.com/apps/financials/receivables/
creditManagement/creditChecking/creditCheckingService/types/\">" +
 "<ns1:request xmlns:ns2=\"http://xmlns.oracle.com/apps/financials/receivables/creditManagement/
creditChecking/creditCheckingService/\">" +
 "<ns2:CustomerName>" + customer + "</ns2:CustomerName>" +
 "<ns2:CustomerAccountNumber>" + accountId + "</ns2:CustomerAccountNumber>" +
 "<ns2:RequestType>Authorization</ns2:RequestType>" +
 "<ns2:PriceType>ONE_TIME</ns2:PriceType>" +
 "<ns2:RecurrencePeriod></ns2:RecurrencePeriod>" +
 "<ns2:RequestAuthorizationAmount currencyCode=\"USD\">" + amount + "</ns2:RequestAuthorizationAmount>" +
 "<ns2:RequestAuthorizationCurrency>USD</ns2:RequestAuthorizationCurrency>" +
 "<ns2:ExistingAuthorizationNumber></ns2:ExistingAuthorizationNumber>" +
 "<ns2:Requestor>ar_super_user</ns2:Requestor>" +
 "</ns1:request>" +
 "</ns1:creditChecking>";

// Use the CreditCheckService web service connector to the Check Check service. You use the Manage External
 Interface Web Service Details task to set up the connector. Since this is a service that is secured
// using message protection policy, we have registered the the https URL of the service
def response = context.invokeSoapService("CreditCheckService", payLoad);

// print a debug message. This appends the entire response to the shipping instuctions attribute.
// To avoid performance problems, you must comment out all debug statements in your production environment.
debug(response.getSoapBody().getTextContent());

// The response XML sent by the Credit Check service contains an element named 'Response'. A YES value
 indicates that credit check passed. Let us extract the contents of Response tag. The following XML API will
 return all nodes (tags)
// with name 'Response' in a NodeList element. We are expecting only one such element in our XML response
def nodeList = response.getSoapBody().getElementsByTagNameNS("*", "Response");

// print out the lenght of the node list
debug(nodeList.getLength());

// Get the first element with name 'Response' (we are expecting only one), and gets its text content
String ccResponse = nodeList.item(0).getTextContent();

debug(ccResponse);

// Check if credit check passed
if( ccResponse != 'YES' ) {
 // Credit check failed. Raise a warning validation exception here
 throw new ValidationException( new Message(Message.MessageType.WARNING, "Credit check failed.") );
}
else {
 // Credit check passed
 // Write the credit check response in an EFF attribute.
 def psiContext = header.getOrCreateContextRow("ComplianceDetails");
 psiContext.setAttribute("_ComplianceInfo", ccResponse);
}

/**
* Appends passed in msg to the Shipping Instructions attribute. This method has been implemented only for
 debugging purposes.
*/
```

```
void debug(def msg) {
 String si = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", si + ", " + msg.toString());
}
```

Inspect Payloads When You Use Web Services or Public View Objects

It might be difficult to examine the response from a web service or from the view criteria of a public view object. Use the setAttribute method during development to get values for the attributes that your code references, display them in the Order Management work area, then examine these values to verify your code returns the values you expect.

For example, assume your extension references shipping instructions. You can use this code to write the contents of ShippingInstructions into the response from the web service.

```
header.setAttribute("ShippingInstructions", response.getSoapBody().getTextContext());
```

You can write the same data into the response from a public view object.

Make sure you convert your test code to log files or comment it (//) before you deploy to production.

You can also write a message to achieve a similar result. However, a message stops or pauses processing. Use a payload so processing can continue without interruption, and so you can view attribute values in the context of how they display in the Order Management work area, such as the Shipping Instructions attribute on the order line.

Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Call Web Services from Order Management Extensions

Use an order management extension to get data from a source that resides outside of Order Management before you do default logic or validation logic.

Use ServiceInvoker in your extension code to call a web service and get the data. ServiceInvoker is available from the context object. You use the ExecutionContext method.

You will create an example extension that:

- Allocates sales credits to a salesperson for order lines that include an item that satisfies a condition.
- References the purchase order number and item number when it calls a web service so it can get the name of the salesperson and the percent sales credit to allocate.
- Extracts the salesperson name and percent allocation from the web service response, then creates a new row in the sales credit row set that it gets from the order line.

Summary of the Steps

1. Register the web service.
2. Call the web service.
3. Determine item of interest.

This topic uses example values. You might need different values, depending on your business requirements.

Register the Web Service

1. Go to the Manage Connector Details page, then set the value.

Attribute	Value
Invocation Mode	Synchronous Service You must use Synchronous Service with an order management extension.

   For details, see *Manage Connector Details Between Order Management and Your Fulfillment System*.
2. Go to the Manage Order Management Extensions page and create a new extension.
3. Set up the call.

Code	Description
`def lines = header.getAttribute("Lines");`	Get the row set for the order lines.
`while( lines.hasNext() )`	Determine whether more lines exist that we must process.

`def line = lines.next();`
 	
`def itemNumber = line.getAttribute("ProductNumb`
	Get the item number that the line specifies.

`if( itemOfInterest(itemNumber)`	Determine whether the item number satisfies a condition.
`allocateSalesCredits(line);`	Call the method that allocates sales credits for this line.

Call the Web Service

Add code that calls the web service so it can get the sales credit allocation.

Code	Description

`void allocateSalesCredits(def line, String itemNumber) {`
 	
`def serviceInvoker = context.getServiceInvoker();`
	Get the handle that you use to call the web service.

`def poNumber = header.getAttribute("CustomerPONu`	Get the value of attribute Purchase Order Number for the customer from the order header.

`String payLoad = "<ns1:GetSalesCreditAllocation xmlns:ns1=\"http://www.your_company.com/SalesCreditWS/\">" +`
 	
`"<ns1:poNumber>" + poNumber + "</ns1:poNumber>" +`
 	
`"<ns1:itemNumber>" + itemNumber + "</ns1:itemNumber>" +`
 	
`"</ns1:GetSalesCreditAllocation>";`
	Concatenate the strings so you can prepare the payload. As an option, you can also use an XML API to concatenate the string.

`def responseBody = serviceInvoker.invokeSoapService( payLoad).getSoapBody();`	Use the interface name of the web service and the request payload to call the web service.

Code	Description
	You must specify the interface name in the Connector Name attribute on the Manage External Interface Web Service Details page. For this example, you specify SalesCreditAllocationService as the connector name. You must not register a connector that uses an Oracle internal web service or an Oracle host. For more, see *Connect Order Management to Your Fulfillment System*.
`def salesPerson = //;`	Extract the salesperson name from the response payload.
`def percent = //`	Extract the percent allocation from the response payload.
`def salesCredits = line.getAttribute("SalesCredits")`	Get the row set for the sales credit from the current line.
`def salesCredit = salesCredits.createRow();`	Create a sales credit row.
`sc.setAttribute("Salesperson", salesPerson);`	Set the value for the salesperson attribute.
`sc.setAttribute("Percent", new BigDecimal(percent));`	Set the value for the percent allocation attribute.
`sc.setAttribute("SalesCreditTypeC 1L);`	Set the sales credit type. In this example, assume that 1 equals Revenue credits.

`salesCredits.insertRow(salesCredi`
 	
`}`
	Insert the new row in the row set.

The code uses parameters.

- `param line`. Line where we will allocate the sales credit.
- `param itemNumber`. Number of the item that you specify on the line.

Determine Item of Interest

Add code that returns a boolean value to indicate whether we are interested in the item that the web service sent.

Code	Description
`boolean itemOfInterest(String itemNumber) {`	For brevity, this example doesn't include details about how to implement this method. The logic you use is specific to your business requirement.
if **condition** `return true;`	**condition** is code you write. It decides whether the item number is of interest.

Code	Description

else
 	
return false
 	
}	

Entire Code

Here's the code for this example.

```
//get the lines row set.
def lines = header.getAttribute("Lines");

//if more lines exist.
while( lines.hasNext() ) {

 def line = lines.next();

//then get the item number that the line specifies.
 def itemNumber = line.getAttribute("ProductNumber");

//determine whether the item number satisfies the condition.
 if( itemOfInterest(itemNumber) ) {

//call method to allocate sales credits for this line.
 allocateSalesCredits(line);
 }
}

/**
 * Call a web service that gets sales credit allocation for the order line.
 * @param line identifies the line where you allocate sales credit.
 * @param itemNumber is the number of the item that the line specifies.
 */
void allocateSalesCredits(def line, String itemNumber) {

//get a handle for the method that calls the web service.
 def serviceInvoker = context.getServiceInvoker();

//get the customer attribute for the purchase order number from the order header.
 def poNumber = header.getAttribute("CustomerPONumber");

String payLoad = "<ns1:GetSalesCreditAllocation

//concatenate the strings to prepare the payload.
xmlns:ns1=\"http://your.address/SalesCreditWS/\">" +
//As an alternative, you can also use this code:
 "<ns1:poNumber>" + poNumber + "</ns1:poNumber>" +
 //XML APIs
 "<ns1:itemNumber>" + itemNumber + "</ns1:itemNumber>" +
 "</ns1:GetSalesCreditAllocation>";

 def responseBody =

//use the web service name and the constructed payload to call the service.
serviceInvoker.invokeSoapService("SalesCreditAllocationService",
payLoad).getSoapBody();

//get the salesperson name from the web service response.
 def salesPerson = //;
```

```
//get the percent allocation from the service response.
 def percent = //

//get the row set for the sales credit for the current line.
 def salesCredits = line.getAttribute("SalesCredits");

//create a new row for the sales credit.
 def salesCredit = salesCredits.createRow();

//set the salesperson attribute.
 sc.setAttribute("Salesperson", salesPerson);

//set the percent allocation attribute.
 sc.setAttribute("Percent", new BigDecimal(percent));

//set the sales credit type. This code assumes your implementation uses the value 1 for revenue credits.
 sc.setAttribute("SalesCreditTypeCode", 1L);

//set a unique identifier in case more than one SalesCredit exists. For example, 5768342869.
 salesperson.setAttribute('SourceTransactionSalesCreditIdentifier',5768342869);

//insert the new row in the rowset.
 salesCredits.insertRow(salesCredit);

}

/**
 *Return a boolean that indicates whether the item is of interest.
 */
boolean itemOfInterest(String itemNumber) {

//For brevity, and to keep focus on calling the web service,
//this example does not include details about how to implement this method.
//The logic you use is specific to your business process.

//Decide whether the item number is of interest. Pseudocode:
 // if (some condition)
 return true;
 // else
 // return false
}
```

Here's the same code without comments.

```
def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {
 def line = lines.next();
 def itemNumber = line.getAttribute("ProductNumber");

 if( itemOfInterest(itemNumber) ) {
 allocateSalesCredits(line);
 }
}
void allocateSalesCredits(def line, String itemNumber) {
 def serviceInvoker = context.getServiceInvoker();
 def poNumber = header.getAttribute("CustomerPONumber");

String payLoad = "<ns1:GetSalesCreditAllocation

xmlns:ns1=\"http://your.address/SalesCreditWS/\">" +

serviceInvoker.invokeSoapService("SalesCreditAllocationService",
payLoad).getSoapBody();
 def salesPerson = //;
 def percent = //
```

```
 def salesCredits = line.getAttribute("SalesCredits");
 def salesCredit = salesCredits.createRow();
 sc.setAttribute("Salesperson", salesPerson);
 sc.setAttribute("Percent", new BigDecimal(percent));
 sc.setAttribute("SalesCreditTypeCode", 1L);
 salesperson.setAttribute('SourceTransactionSalesCreditIdentifier',5768342869);
 salesCredits.insertRow(salesCredit);

}
boolean itemOfInterest(String itemNumber) {

//implement this method.

//Decide whether the item number is of interest. Pseudocode:
 // if (some condition)
 return true;
 // else
 // return false
}
```

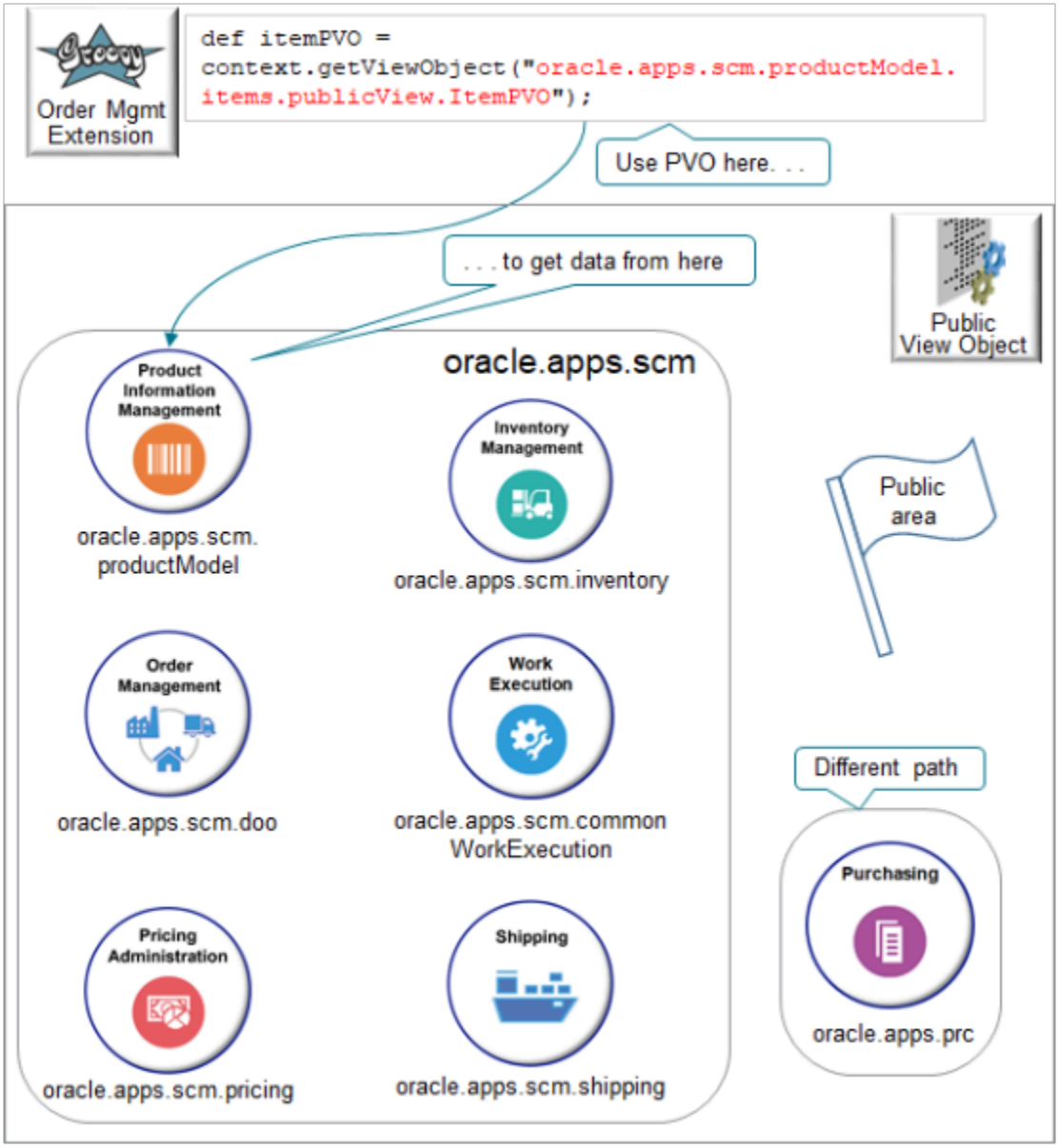
Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Guidelines for Using Extensions to Get Data from Oracle Applications

Use a public view object (PVO) in your order management extension to get data from a variety of sources, such as an Oracle Application.

Each public view object uses a path.



Note

- Use the getViewObject method.
- Most of the view objects that you need are in the oracle.apps.scm path. In some cases you might need to go outside of oracle.apps.scm. For example, use oracle.apps.prc to access purchase order data from Oracle Purchasing.
- For example, you can use a public view object to look up an item category in the Product Information Management work area or a purchase order in Oracle Purchasing. This example uses `oracle.apps.scm.productModel.items.publicView.ItemPVo` to get details about an item from Product Information Management.
- `public` means that the view object makes its data available to other Oracle Applications.

CAUTION: We recommend that you use REST API to access data through business objects. However, if you can't use REST API for some reason, then you can use a public view object instead, but you must use only the public view objects that Oracle supports. To get the full list of them and the attributes that they contain, download the XLS file from *Public View Objects in Oracle Applications Cloud (Document ID 2386411.1)*.

Filter Your PVOs

Filter your search in the XLS file that you download.

Assume you want the PVO that filters data according to the customer name on the order header.

1. Download and open the XLS file.
2. Use Excel's advanced filter to filter the columns in the XLS file.

Path	Attribute
Contains **`foundation.parties.publicView`** The Customer Name is party data, so you use this filter because Order Management gets its party data from **`foundation.parties`**. For details about party data, see *Overview of Displaying Customer Details on Sales Orders*.	Equals **`PartyName`**

3. Assume you get these results.

Path	Attribute
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName

Path	Attribute
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName
oracle.apps.cdm.foundation.parties.publ	PartyName

4. Look for the most likely path. The results contain a variety of usages, such as analytics, bicc, relationships, export, and core. You're interested in the core usage, so look for the path that includes `core`.
5. Here's what you would use for this example.

Path	Attribute
`oracle.apps.cdm.foundation.par`	`PartyName`

Other Filters

Oracle Applications might use the abbreviation FOM (Fusion Order Management) or DOO (Distributed Order Orchestration) to refer to Oracle Order Management. To find objects that are relevant to Oracle Order Management, you can also filter the Path column for rows that contain `fom` or `doo`.

Specify the Path

The path identifies the public view object. The nomenclature for each PVO can be different. Here's one example:

productFamily.functionalArea.publicView.*usage.PVOname*

Consider an example usage:

`oracle.apps.scm.maintenanceManagement.maintProgram.publicView.analytics.MaintenanceForecastFactPVO`

where

- `oracle.apps.scm` identifies the Supply Chain Management product family.
- `maintenanceManagement` identifies the Maintenance Management functional area.
- `maintProgram` identifies a component in the Maintenance Management functional area.
- `MaintenanceForecastFactPVo` identifies the name of the public view object.
- `analytics` identifies how you can use this PVO.

Here are the usages.

Usage	Description
`analytics`	Use with a report.
`bicc`	Use with Business Intelligence Cloud Connector.
`core`	Manage data from the core Oracle Applications.
`export`	Import or export data.

Usage	Description
`relationships`	Manage a relationship between objects.

Typical View Objects

Here are some of the public view objects that you typically use in an Order Management implementation.

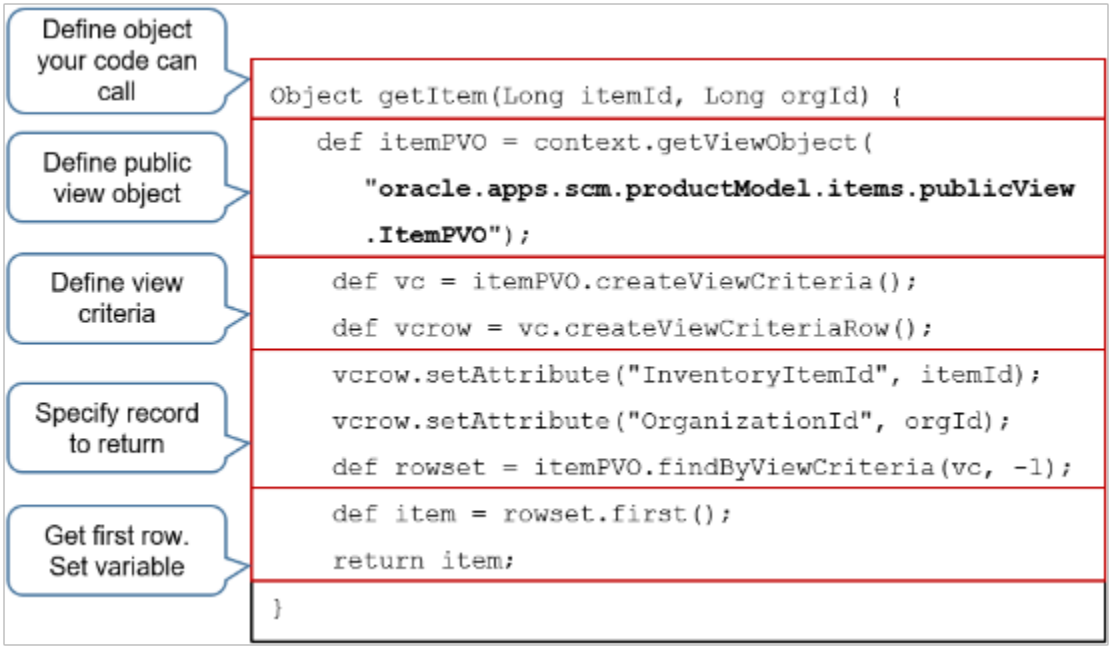
Data	View Object Name
Sales Order	HeaderPVO FulfillLinePVO
Item	ItemPVO ItemCategoryPVO
Customer	PartyPVO PartySitePVO LocationPVO
Customer Account	CustomerAccountPVO CustomerAccountSitePVO CustomerAccountSiteUsePVO
Receivable	TransactionTypePVO

Here are some of the more common paths to the various applications that you might need to access.

Application or Work Area	Path
Product Information Management	oracle.apps.scm.productModel
Order Management	oracle.apps.scm.doo
Pricing Administration	oracle.apps.scm.pricing
Inventory Management	oracle.apps.scm.inventory
Work Execution	oracle.apps.scm.commonWorkExecution
Shipping	oracle.apps.scm.shipping

Application or Work Area	Path
Purchasing	oracle.apps.prc

Code It



Note

Code	Description
`Object getItem(Long itemId, Long orgId) {`	Define a local object that your code can call. This object will contain data from the public view object. • **`Object getItem`**. Define an object named getItem. • **`Long itemId, Long orgId`**. Specify itemId and orgId as Long data type strings. These strings will contain data from the public view object.
`def itemPVO = context.getViewObject("context")`	Create an instance of the public view object for the item, then store it in the itemPVO variable. Use this format: • **`itemPVO`**. Variable that will contain the instance. • **`context.getViewObject`**. Use the getViewObject method to get the view object according to the context. • **`"context"`**. Context to use when getting the view object. For example, reference the ItemPVO public view object: `def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO")` where

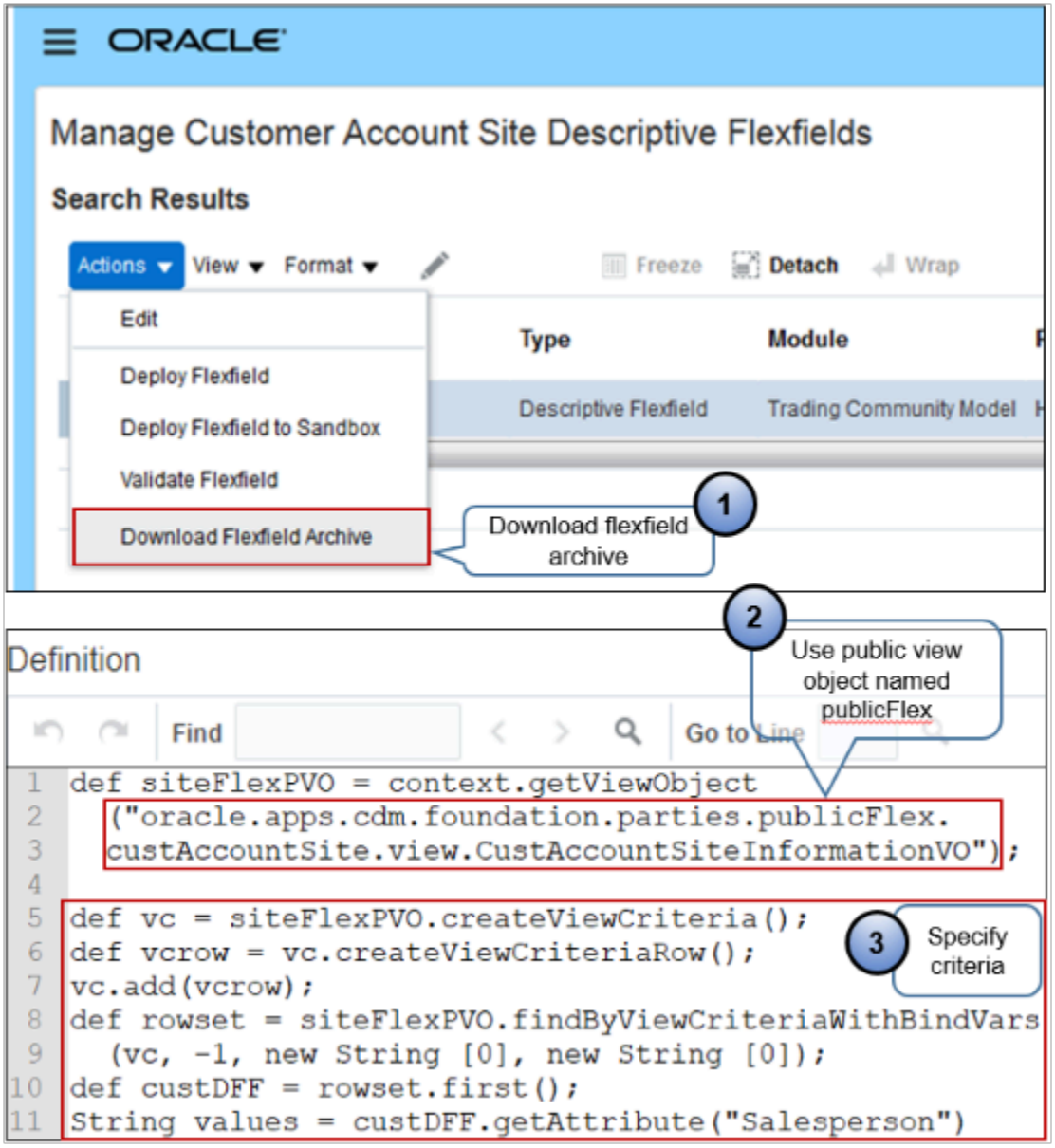
Code	Description
	• **oracle.apps.scm.productModel.items** is the context. It specifies to get a product model named **items** from supply chain management.
def vc = itemPVO.createViewCriteria();	Define the variable that will store details for the public view object. • **vc**. Variable that stores details for the view criteria object. **ViewCriteria**. A filter that specifies the data that you want the public view object to return. • **itemPVO**. Name of the public view object that contains the data you must get and filter. • **createViewCriteria**. Method that creates the view criteria object. It defines **vc** as a list that contains one or more rows.
def vcrow = vc.createViewCriteriaRow();	Define a variable for the row. • **vcrow**. Variable that stores details for rows of the view criteria object. • **vc.createViewCriteriaRow**. Use the createViewCriteriaRow method to create a row in the **vc** variable.
vcrow.setAttribute("InventoryItemI itemId);	Populate the **vcrow** variable with data from the public view object. • **setAttribute**. Method that sets the value for the attribute in the row. • **InventoryItemId**. Attribute in the public view object. This example gets the value that uniquely identifies the item in inventory. • **itemId**. Variable that the extension code defines. This code gets the value of the InventoryItemId attribute on the order line from the public view object, then stores it in itemId.
vcrow.setAttribute("OrganizationId orgId);	Populate the **vcrow** variable with data from the public view object. • **setAttribute**. Method that sets the value for the attribute in the row. • **OrganizationId**. Attribute in the public view object. This example gets the value that uniquely identifies the organization that ordered the item. • **orgId**. Variable that the extension code defines. This code gets the value of the OrganizationId attribute on the order header from the public view object, then stores it in orgId.
def rowset = itemPVO.findByViewCriteria(vc, -1);	Get the iterator that contains the number of rows that meet the criteria and store it in the **rowset** local variable. • **rowset**. Variable that stores the iterator. • **findByViewCriteria**. Method that searches the public view object according to criteria that you define. • **vc**. Contains the data that the public view object returns. • **-1**. Return all rows that match the view criteria. You can also use a positive integer to get a subset of rows. For example, use **vc, 3** to get the first three rows that match the criteria. If only two rows match the criteria, then the method returns only these two rows.
def item = rowset.first()	Define a variable named **item**.
return item	The **item** variable can now access all attributes in ItemPVO. **return** returns the **item** variable so you can reference **item** from your extension to get details from ItemPVO.

Here's the entire code without comments.

```
Object getItem(Long itemId, Long orgId) {
 def itemPVO = context.getViewObject(
 "oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 def rowset = itemPVO.findByViewCriteria(vc, -1);
 def item = rowset.first();
 return item;
}
```

Get Descriptive Flexfields from a Public View Object

The customer master in Oracle Trading Community Architecture (TCA) might include descriptive flexfields. However, public view objects that contain site details for the customer account, such as PartyPVO, might not include these descriptive flexfields. Use this technique to get them.



Note

1. Go to the Manage Customer Account Site Descriptive Flexfields page, locate the descriptive flexfield that you must access, click **Download Flexfield Archive**, then examine the structure that it uses.
2. Reference the publicFlex public view object to reference the descriptive flexfield. In this example, you reference publicFlex for custAccountSite.

   ```
   oracle.apps.cdm.foundation.parties.publicFlex.custAccountSite.view.CustAccountSiteInformationVO
   ```

3. Use the structure that you examined when you downloaded the archive to help determine how to specify the view criteria.

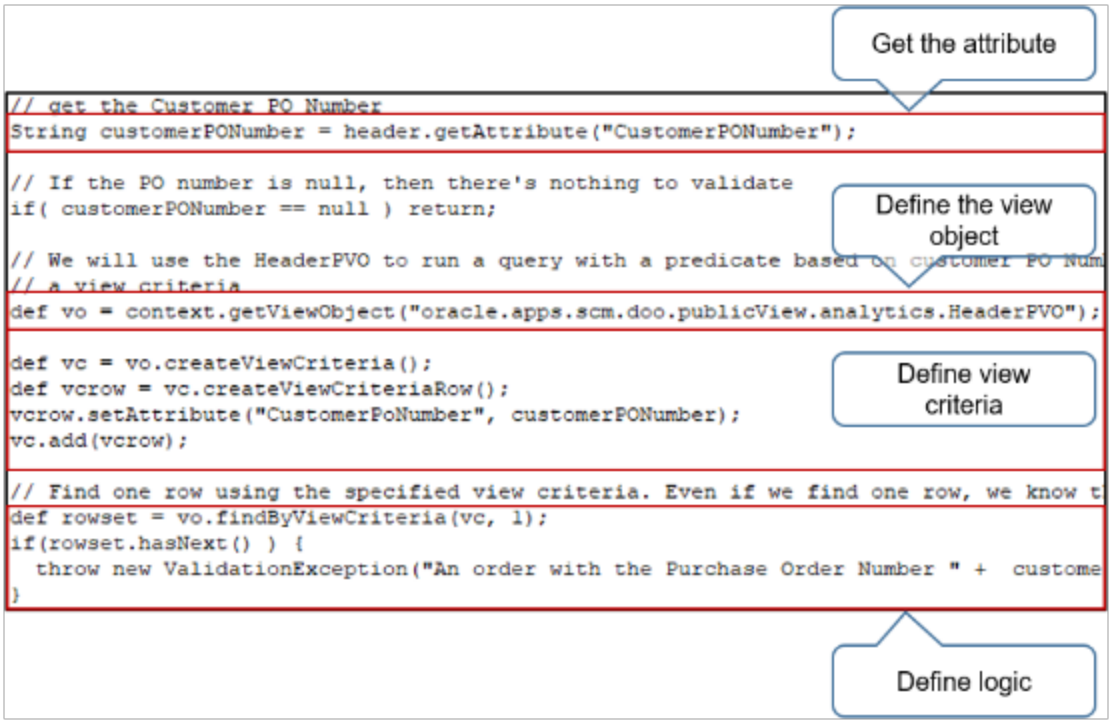
Here's the entire code.

```
def siteFlexPVO = context.getViewObject
 ("oracle.apps.cdm.foundation.parties.publicFlex.
 custAccountSite.view.CustAccountSiteInformationVO");
```

```
def vc = siteFlexPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vc.add(vcrow);
def rowset = siteFlexPVO.findByViewCriteriaWithBindVars
 (vc, -1, new String [0], new String [0]);
def custDFF = rowset.first();
String values = custDFF.getAttribute("Salesperson")
```

Get Data From Sales Orders That Aren't in the Current Sales Order

You can use HeaderPVO to access order header data and FulfillLinePVO to access order line or fulfillment line data. For example, if the Order Entry Specialist sets the purchase order number on a sales order, then make sure some other sales order doesn't already use this purchase order number. Do this validation when the Order Entry Specialist submits the sales order.



Here are some more examples.

- If a new sales order includes return lines, then get the order type from the original sales order and use it to set the order type on each return line.
- If order revision 2 increases the total price of the sales order by 10% or more over the total price on order revision 1, then don't allow the revision. For example, version 1 of sales order x is a complete and separate sales order from version 2 of sales order x. Use a public view object to get data from a version that isn't the current version.

Get Details about Tables, Views, and Public View Objects

If you need to use a public view object that contains details for sales order headers and order lines, then have a look at the documentation. Go to *Oracle Fusion Cloud SCM: Tables and Views for SCM*, expand **Order Management > Tables**, then click one of:

- DOO_HEADERS_ALL
- DOO_LINES_ALL

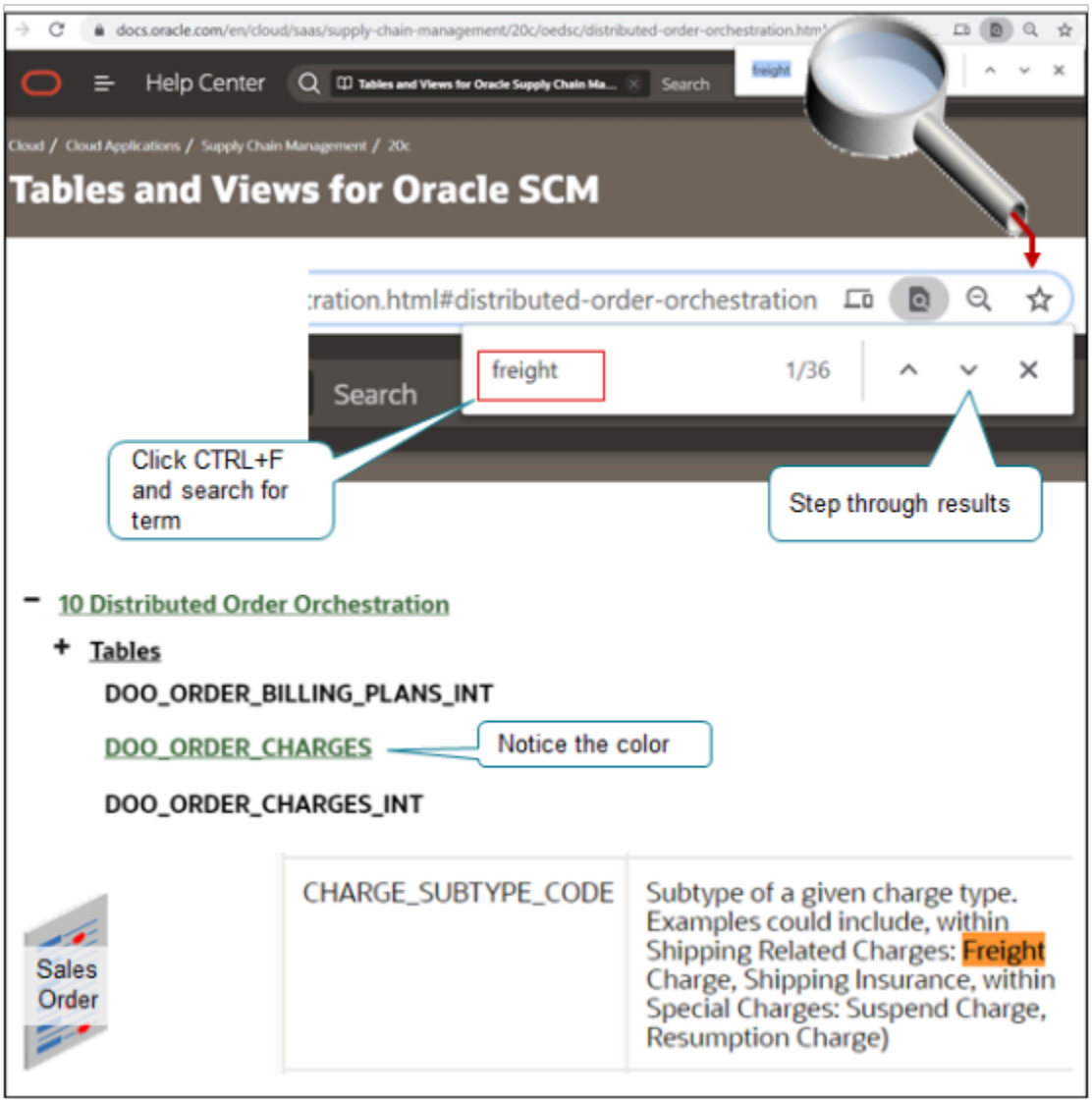
- DOO_FULFILL_LINES_ALL

Get Details for Charges

Assume you're creating a sales order report and need to include freight charges on the report. You have a sales order that's in Awaiting Shipping status or Awaiting Billing status. You view the order in a fulfillment view. The Your Price column on the fulfillment line contains a value for the Sale Price attribute but the Freight attribute is empty. It's also empty in Oracle Shipping.

You use SQL to examine the DOO_LINES_ALL table and the DOO_FULLFILL_LINES_ALL table, but no luck there either.

You can use the documentation to identify the table and column that has the freight charge.



Try it.

1. Go to *Tables and Views for Oracle SCM Cloud*.
2. In the table of contents, click **Order Management**.

3. Press **CTRL+F**, type in the word `freight`, then press **ENTER**.

   The search returns all results starting at the beginning of the Order Management chapter.
4. Click the **down arrow** until find what you're looking for.

   In this example, keep clicking until you see the DOO_ORDER_CHARGES link in the table of contents turn green, and that the CHARGE_SUBTYPE_CODE column contains a relevant description.

   ```
   Subtype of a given charge type. Examples could include, within Shipping Related Charges: Freight Charge,
    Shipping Insurance, within Special Charges: Suspend Charge, Resumption Charge)
   ```

   You might see CHARGE_SUBTYPE_CODE in other tables, such as DOO_FS_CHARGES_INT, but in most cases you use the DOO_ORDER_CHARGES table.
5. Run an SQL to get the freight charge.

Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Use Extensions to Get Data from Oracle Applications

Write an order management extension that calls an API on the context object so it queries the table and accesses the public view object that the table contains.

If you deploy into Oracle Applications, then you can't programmatically access the Oracle database. You can't use PL/I (Programming Language One), SQL (Structured Query Language), or JDBC (Java Database Connectivity) to access data in other application tables. Write an extension instead.

Here's an example extension that accesses a public view object.

Code	Description
`def itemPVO = context.getViewObject("oracle.apps`	Create an instance of the public view object for the item. You must specify the entire name of the public view object. This code uses each public view object as a row iterator, so the methods on the RowIterator interface are also available on each public view object. You can use the RowSetIterator method to access a method that gets the rows that the view object query returns, and to navigate the row set for the public view object. Have a look at the ViewObject method. For details, see *Methods That You Can Use with Order Management Extensions*.
`def item = itemPVO.first();`	Access the first item.

Here are the public view objects that you typically use.

Data	View Object Name
Sales Order	HeaderPVO FulfillLinePVO
Item	ItemPVO ItemCategoryPVO
Customer	PartyPVO PartySitePVO LocationPVO
Customer Account	CustomerAccountPVO CustomerAccountSitePVO CustomerAccountSiteUsePVO
Receivable	TransactionTypePVO

CAUTION: We recommend that you use REST API to access data through business objects. However, if you can't use REST API for some reason, then you can use a public view object instead, but you must use only the public view objects that Oracle supports. To get the full list of them and the attributes that they contain, download the XLS file from *Public View Objects in Oracle Applications Cloud (Document ID 2386411.1)*.

Use View Criteria Objects to Filter Results

The example above is only for demonstration purposes. It's too simplistic for practical use because it gets arbitrary items from the items table. It's more likely that you must query for an item or a set of items that match a criterion. To support SQL, your extension must provide a WHERE clause that only selects rows that match a criterion. Use the ViewCriteria object for this purpose.

An order management extension can create a view criteria object that adds filtering logic to the public view object before it gets data from these objects. A **view criteria object** is a filter that you create and apply programmatically to a view object. Order Management converts each of these filters into a WHERE clause when your extension runs the query that you define in the public view object.

You use the **view criteria row object** to create a view criteria object. The view criteria object contains the attribute names and attribute values that become part of the WHERE clause.

In this example, you will create an extension that:

- Creates a view criteria and uses it to query a public view object.
- Queries the item master for an item according to the item Id and inventory organization Id.
- Examines the HazardousMaterialFlag attribute on the item.
- If the HazardousMaterialFlag attribute flags the item as hazardous, then the extension sets the Shipping Instruction flexfield context segment to indicate that the item needs hazardous handling.

This topic uses example values. You might need different values, depending on your business requirements.

Use view criteria objects to filter results:

1. Get the item.

Code	Description
`def lines = header.getAttribute("Lines");`	Get the row set for the order lines.
`while( lines.hasNext() ) {`	Determine whether we must process more order lines.
`def line = lines.next();`	Get the next line and assign it to the variable line.
`def inventoryItemId = line.getAttribute("ProductIden`	Get the inventory item Id for the item from the order line that the Order Entry Specialist selected.
`def orgId = line.getAttribute("InventoryOr`	Get the organization for the item from the order line that the Order Entry Specialist selected.
`def item = getItem(inventoryItemId, orgId);`	Get the item. Use the item Id and the organization Id to call the getItem method.
`String hazardous = item.getAttribute("HazardousMa`	Get the HazardousMaterialFlag attribute from the item.
`if( "Y".equals(hazardous) ) {`	Determine whether HazardousMaterialFlag flags the item as hazardous.
`def packShipInstruction = line. getOrCreateContextRow("PackShi`	Get the row for the extensible flexfield context named PackShipInstruction.
`packShipInstruction.setAttribu ShippingInstruction", "Hazardous Handling Required.");`	Set the Shipping Instruction context segment.

2. Define the public view object.

Code	Description

`Object getItem(Long itemId, Long orgId) {`
 	
`def itemPVO = context.getViewObject("oracle.`
 	
`scm.productModel.items.publicV`
	Create an instance of the public view object for the item.

`def vc = itemPVO.createViewCriteria();`	Create the view criteria object.
`def vcrow = vc.createViewCriteriaRow();`	Create the view criteria row.
`vcrow.setAttribute("InventoryI itemId);`	Set the inventory item attribute so you can include it in the filter condition, and set the value that you will use to compare to this attribute.
`vcrow.setAttribute("Organizati orgId);`	Set the organization attribute so you can include it in the filter condition, and set the value that you will use to compare to this attribute.
`def rowset = itemPVO.findByViewCriteria(vc, -1);`	Define the view criteria that filters the rows when you query the public view object.
`def item = rowset.first();`	Get the first item row that meets the condition.

This code uses these parameters:

- `param itemId`. Inventory item Id that identifies the item.
- `param orgId`. Inventory organization Id that identifies the organization that owns of the item.

Code Without Comments

Here's the entire code for this example with no comments.

```
def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def item = getItem(inventoryItemId, orgId);
 String hazardous = item.getAttribute("HazardousMaterialFlag");
 if( "Y".equals(hazardous) ) {
```

```
def packShipInstruction = line. getOrCreateContextRow("PackShipInstruction");
packShipInstruction.setAttribute("_ShippingInstruction", "Hazardous Handling Required.");
 }
}

Object getItem(Long itemId, Long orgId) {
def itemPVO = context.getViewObject("oracle.apps.scm.productModel.items.publicView.ItemPVO");
 def vc = itemPVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("InventoryItemId", itemId);
 vcrow.setAttribute("OrganizationId", orgId);
 def rowset = itemPVO.findByViewCriteria(vc, -1);
 def item = rowset.first();
 return item;
}
```

Example

This example calls a web service that does a credit check.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

def poNumber = header.getAttribute("CustomerPONumber");

if (poNumber != "PMC TEST") return;

//get attribute to populate in the payload
String customer = header.getAttribute("BillToCustomerName");
Long accountId = header.getAttribute("BillToCustomerIdentifier");
BigDecimal amount = new BigDecimal(1000);

//prepare the payload
String payLoad = "<ns1:creditChecking xmlns:ns1=\"http://xmlns.oracle.com/apps/financials/receivables/
creditManagement/creditChecking/creditCheckingService/types/\">" +
 "<ns1:request xmlns:ns2=\"http://xmlns.oracle.com/apps/financials/receivables/creditManagement/
creditChecking/creditCheckingService/\">" +
 "<ns2:CustomerName>" + customer + "</ns2:CustomerName>" +
 "<ns2:CustomerAccountNumber>" + accountId + "</ns2:CustomerAccountNumber>" +
 "<ns2:RequestType>Authorization</ns2:RequestType>" +
 "<ns2:PriceType>ONE_TIME</ns2:PriceType>" +
 "<ns2:RecurrencePeriod></ns2:RecurrencePeriod>" +
 "<ns2:RequestAuthorizationAmount currencyCode=\"USD\">" + amount + "</ns2:RequestAuthorizationAmount>" +
 "<ns2:RequestAuthorizationCurrency>USD</ns2:RequestAuthorizationCurrency>" +
 "<ns2:ExistingAuthorizationNumber></ns2:ExistingAuthorizationNumber>" +
 "<ns2:Requestor>ar_super_user</ns2:Requestor>" +
 "</ns1:request>" +
 "</ns1:creditChecking>";

//Use the CreditCheckService web service connector to call the Check Credit service. Use the Manage External
 Interface Web Service Details task to set up the connector. This is a secured service,
//so we're using a message protection policy. We use the https URL of the service to register it.
def response = context.invokeSoapService("CreditCheckService", payLoad);

//Print a debug message. Append the entire response to the shipping instructions attribute.
//To avoid performance problems, you must comment out all debug statements in your production environment.
debug(response.getSoapBody().getTextContent());

//The response XML that the Credit Check service sends contains an element named Response. A YES value in
 the response means credit check passed. So, we extract the contents of the Response tag. The following XML
 API returns all tags
//that include the name Response in a NodeList element. We are expecting only one element in our XML
 response.
def nodeList = response.getSoapBody().getElementsByTagNameNS("*", "Response");
```

```
//Print out the lenght of the node list.
debug(nodeList.getLength());

//Get the first element that contains the name Response. We are expecting only one response. Then get its
 text content
String ccResponse = nodeList.item(0).getTextContent();

debug(ccResponse);

//Determine whether credit check passed.
if (ccResponse != 'YES') {
 //Credit check failed, so we a warning validation exception.
 throw new ValidationException(new Message(Message.MessageType.WARNING, "Credit check failed."));
} else {
 //Credit check passed.
 //Write the credit check response in an extensible flexfield.
 def psiContext = header.getOrCreateContextRow("ComplianceDetails");
 psiContext.setAttribute("_ComplianceInfo", ccResponse);
}

/**
 * Append the message that we received into the Shipping Instructions attribute. Use this method only for
 debugging purposes.
 */
void debug(def msg) {
 String si = header.getAttribute("ShippingInstructions");
 header.setAttribute("ShippingInstructions", si + ", " + msg.toString());
}
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Example of Using an Extension to Get Data from Oracle Applications

Identify the public view object you need, then reference it in an order management extension.

Assume you need to use an order management extension to set the default value of the Freight Term attribute.

- Use a descriptive flexfield to store the value for the Freight Term Code attribute.
- Use an order management extension to get the value of the flexfield, then set the Freight Term attribute on the sales order.

You can get the value from the flexfield during testing. You notice that Order Management stores the value of the Freight Term Code as a number, but you need the meaning too. The WSH_FREIGHT_CHARGE_TERMS lookup stores the meaning.

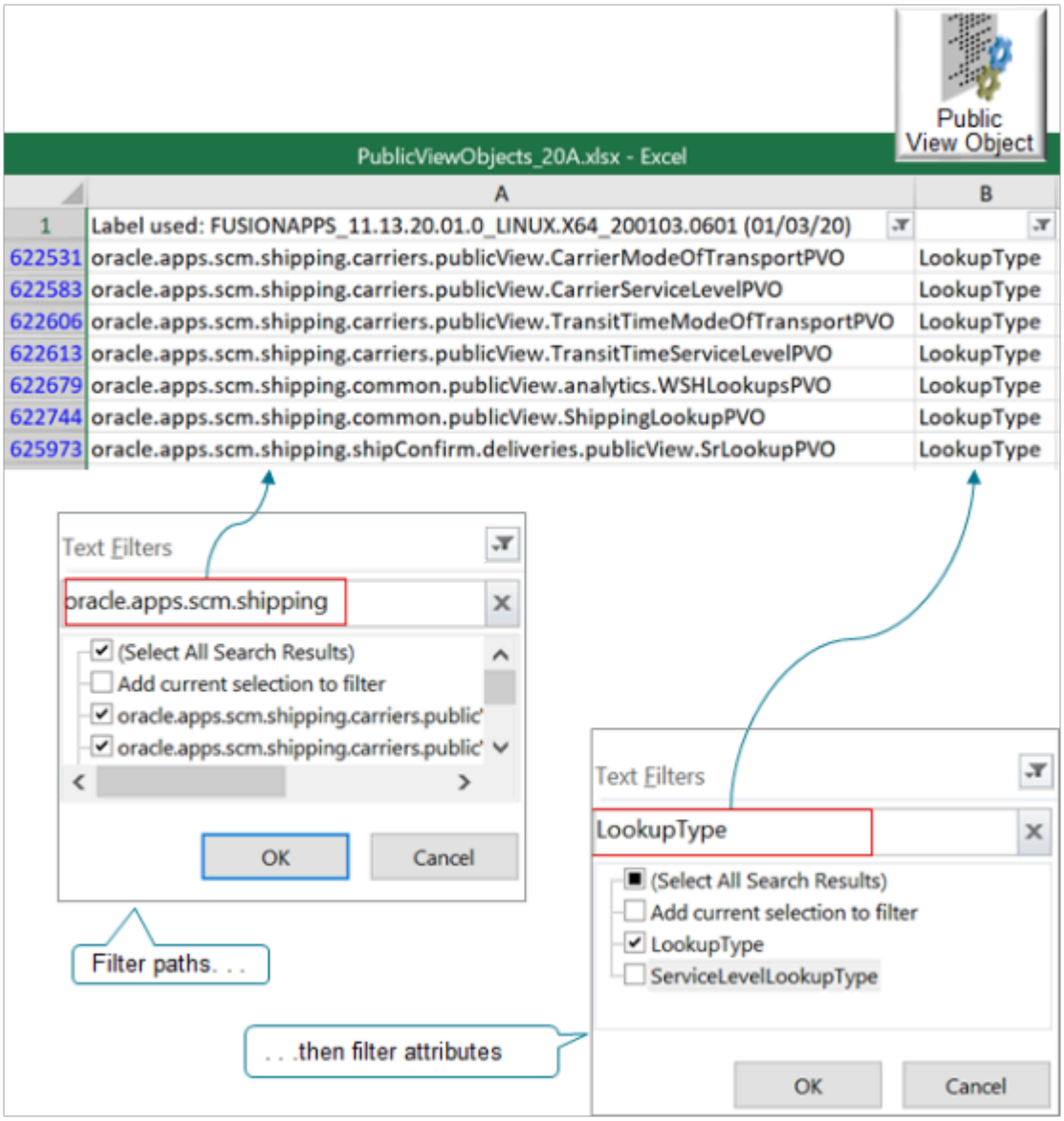
You get the meaning, then use `String vFreightTermsMeaning = lookupRow.getAttribute('Meaning');` to store it in a string, which you can then display in an attribute on the order header or the order line.

Try it.

1. Go to *Public View Objects in Oracle Applications Cloud (Doc ID 2386411.1)*.
2. Download and open the XLS file for your update, such as Update 22A.

3. In Microsoft Excel, open the POVs and Attributes worksheet.
4. Click the **Data** tab, then click **Filter**.
5. Click the `Filter icon` in row 1 of Column A.

   There are over 600,000 public view objects, so you need to filter for only the ones you need.



6. In the Filter dialog, in the search window, enter `oracle.apps.scm`.

   Most of the view objects you need are in the oracle.apps.scm path. Scroll through the search results in the Filter dialog so you can get an idea of the data you can access.

   In this example you need to get a value from shipping, so enter `oracle.apps.scm.shipping` in the search window, add a check mark to the Select All Search Results option, then click **OK**.

   For other examples, you might need to search and select values that start with a different value, depending on your requirements.

Path	Contains Data From
oracle.apps.scm.doo	Order Management
oracle.apps.scm.pricing	Oracle Pricing
oracle.apps.scm.inventory	Oracle Inventory Management
oracle.apps.scm.productModel	Product Information Management
oracle.apps.scm.commonWorkExecutior	Work Execution
oracle.apps.scm.dos	Supply Chain Orchestration

   In some cases you might need to go outside of oracle.apps.scm. For example, use oracle.apps.prc to access purchase order data from Oracle Purchasing.
7. Search for text that likely contains the data you need.

   You're looking for data in WSH_FREIGHT_CHARGE_TERMS, which is a type of lookup, so click the `Filter icon` in row 1 of Column B, then enter `LookupType`.
8. Notice the results.

Path	Attribute
oracle.apps.scm.shipping.carriers.public'	LookupType
oracle.apps.scm.shipping.carriers.public'	LookupType
oracle.apps.scm.shipping.carriers.public'	LookupType
oracle.apps.scm.shipping.carriers.public'	LookupType

Path	Attribute
oracle.apps.scm.shipping.common.publi	LookupType
oracle.apps.scm.shipping.common.publi	LookupType
oracle.apps.scm.shipping.shipConfirm.d	LookupType

Most of the paths reference specific kinds of data, such as mode of transport or transit time. You need the more generic lookup, which is in `oracle.apps.scm.shipping.common.publicView.ShippingLookupPVO`.

9. Reference the public view object in your extension.

```
def lookupPVO = context.getViewObject("oracle.apps.scm.shipping.common.publicView.ShippingLookupPVO");
def vc = lookupPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();

vcrow.setAttribute('LookupType', 'WSH_FREIGHT_CHARGE_TERMS' );
vcrow.setAttribute('LookupCode', <CODE> );
vc.add(vcrow);
def rowset = lookupPVO.findByViewCriteriaWithBindVars(vc, -1, new String [0], new String [0]);
def lookupRow = rowset.first();
String vFreightTermsMeaning = lookupRow.getAttribute('Meaning');
```

where

Code	Description
`def lookupPVO = context.getViewObject("oracle.`	Get the value from the oracle.apps.scm.shipping.common.publicView.ShippingLookupPVO
`vcrow.setAttribute('LookupType 'WSH_FREIGHT_CHARGE_ TERMS' );`	Set the value of the LookupType attribute to WSH_FREIGHT_CHARGE_TERMS.
`vcrow.setAttribute('LookupCode <CODE> );`	Set the value of the LookupCode attribute. Replace CODE with the value of the lookup code.
`def rowset = lookupPVO.findByViewCriteriaWi -1, new String [0], new String [0]);`	Create the string where you will store the value of the meaning.
`def lookupRow = rowset.first();`	Get the value of the Meaning attribute, then store it in the string.

Code	Description
`String vFreightTermsMeaning = lookupRow.getAttribute('Meanin`	

Related Topics

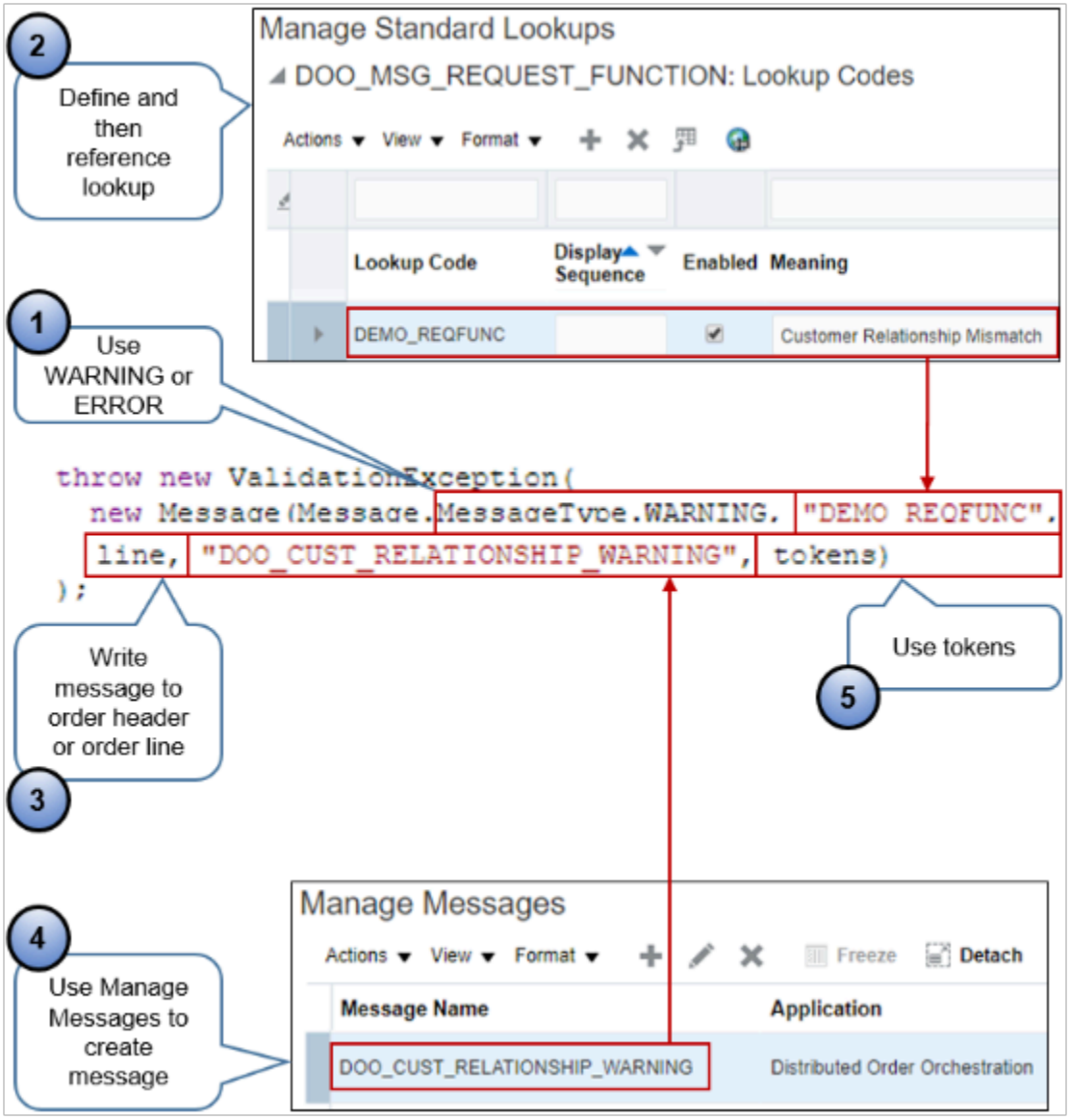
- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Messages

Guidelines for Using Extensions to Manage Messages

Use an extension to define a message.

Set Up Your Message



Note

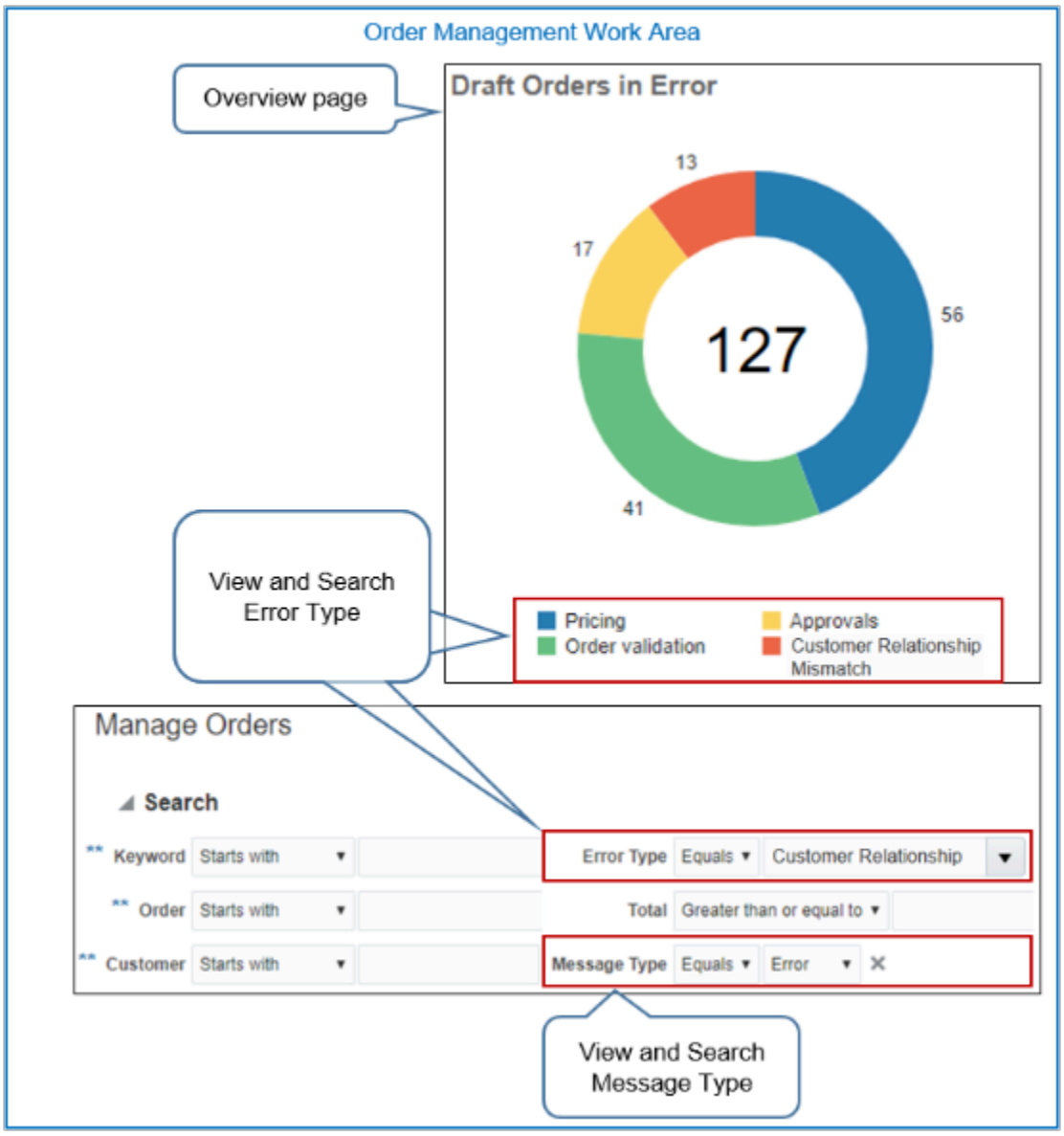
1. **messageType**. Define the messageType as WARNING or ERROR. The Order Management work area uses a different icon for each type.
2. **Lookup**. Use the Manage Order Lookups page in the Setup and Maintenance work area to categorize your messages, then reference them from your code.
   Use predefined lookups or create your own. Categorizing helps the Order Entry Specialist search for messages and view them in analytic charts.
   Use Error to improve search and display because errors display in work area Order Management in attribute Error Type.
   In this example, assume you define the Meaning as Customer Relationship Mismatch.
3. **Header or line**. Indicate where the error happens. Use one of these values.
   - **header**. Apply the message to an error that happens on the order header.

- **line**. Apply the message to an error that happens on an order line.

  The example in the screen print uses `line`. This example includes code (not visible, for brevity) above `throw new ValidationException` that defines `line` as a variable, uses it to iterate through all the lines in the sales order, then saves the order line number where the error happens, such as line 3. At run time, Order Management displays an icon on line 3 that the Order Entry Specialist can click to examine message details.

4. **Manage Messages**. Use the Manage Messages page in the Setup and Maintenance work area.
   - Referencing a message provides advantages over hard coding the message.
   - Reference the message from different extensions, manage the message independently of the extension, and translate the message to some other language besides English.
   - Using the Manage Messages page can simplify message management because other developers and administrators can then use Manage Messages instead of modifying `hard coded` messages in your extension, which requires knowledge of writing in Groovy.
   - You can also define a message to help you troubleshoot your extension during development. You can include details in the message that display the state of various objects to help you pinpoint problem areas. Remove the message when you're ready to deploy your extension to your production environment.
5. Use tokens as placeholders for variable content.

The Order Entry Specialist can view and search data in the Order Management work area according to Error Type or Message Type. In this example, notice that the Overview page displays Customer Relationship Mismatch as a label in the Draft Orders in Error diagram.



The Order Entry Specialist can take action.

- Click on the orange part of the circle to drill into sales orders that are in error because of the mismatch.
- Search for the Error Type attribute on the Manage Orders for Customer Relationship Mismatch page.

Accumulate Error Messages and Display Them

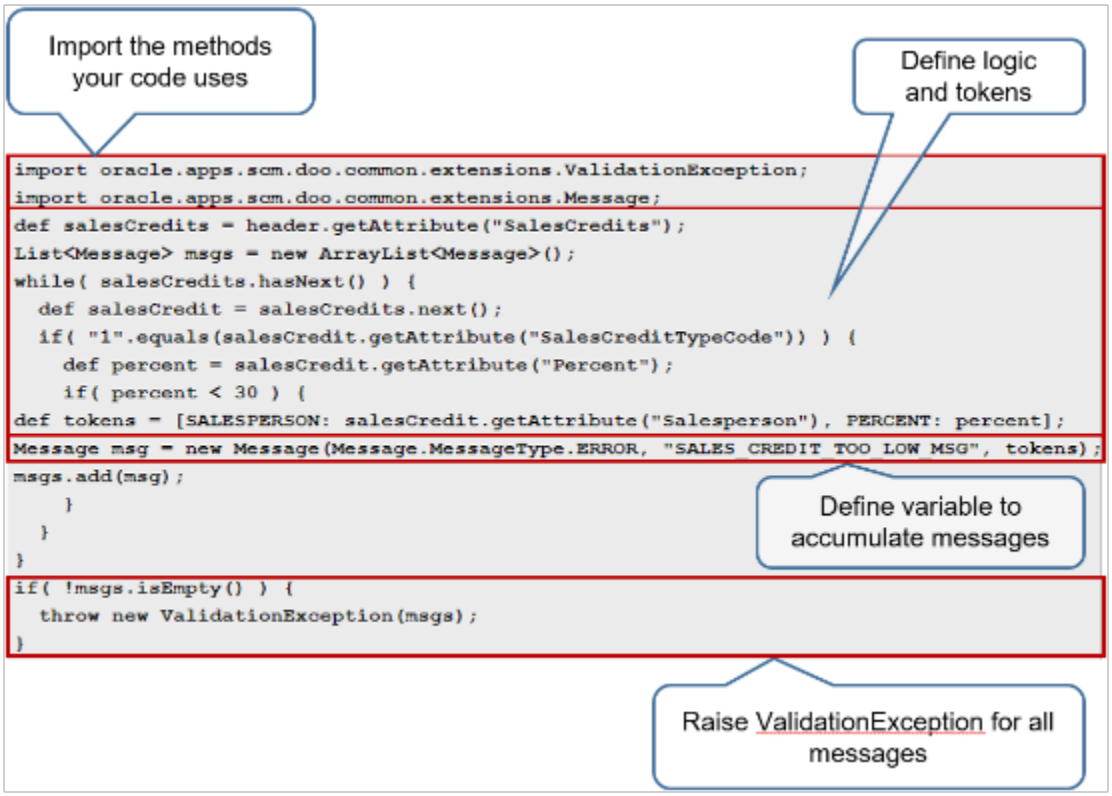
Accumulate messages, then display them in the Order Management work area all together at one time.

For example, assume you write an extension that contains three lines that check for error conditions, x, y, and z. At run time, assume x, y and z all meet their error conditions, but the extension stops immediately after it encounters x and displays the message for error x.

Instead, write your extension so it continues to run through y and z, saves each message to a temporary list, then displays messages for x, y and z together in a single dialog. This technique allows the Order Entry Specialist to examine

all errors together, correct them, then resubmit the sales order instead of correcting x, submit, correct y, submit, correct z, and submit.

For example:



If the revenue percentage is less than 30% for the sales credit, then the extension creates a message and stores it in a local list. The extension processes all sales credits. If the local list contains any messages after it finishes processing sales credits, then the extension displays them in a dialog in the Order Management work area.

Here is the code that this example uses to examine each sales credit.

Code	Description
`import oracle.apps.scm.doo.common.extens`	Import the ValidationException method so your code can use it.
`import oracle.apps.scm.doo.common.extens`	Import the Message method so your code can use it.
`def salesCredits = header.getAttribute("SalesCredits`	Define a local variable named salesCredits, use the getAttribute method to get the value of the SalesCredits attribute from the order header, then set the value of this variable to the value that getAttribute returns. getAttrbute("SalesCredits") returns an iterator you can use to access sales credits rows. For example, the code references salesCredits later in this topic to get SalesCreditTypeCode, Percent, and Salesperson.

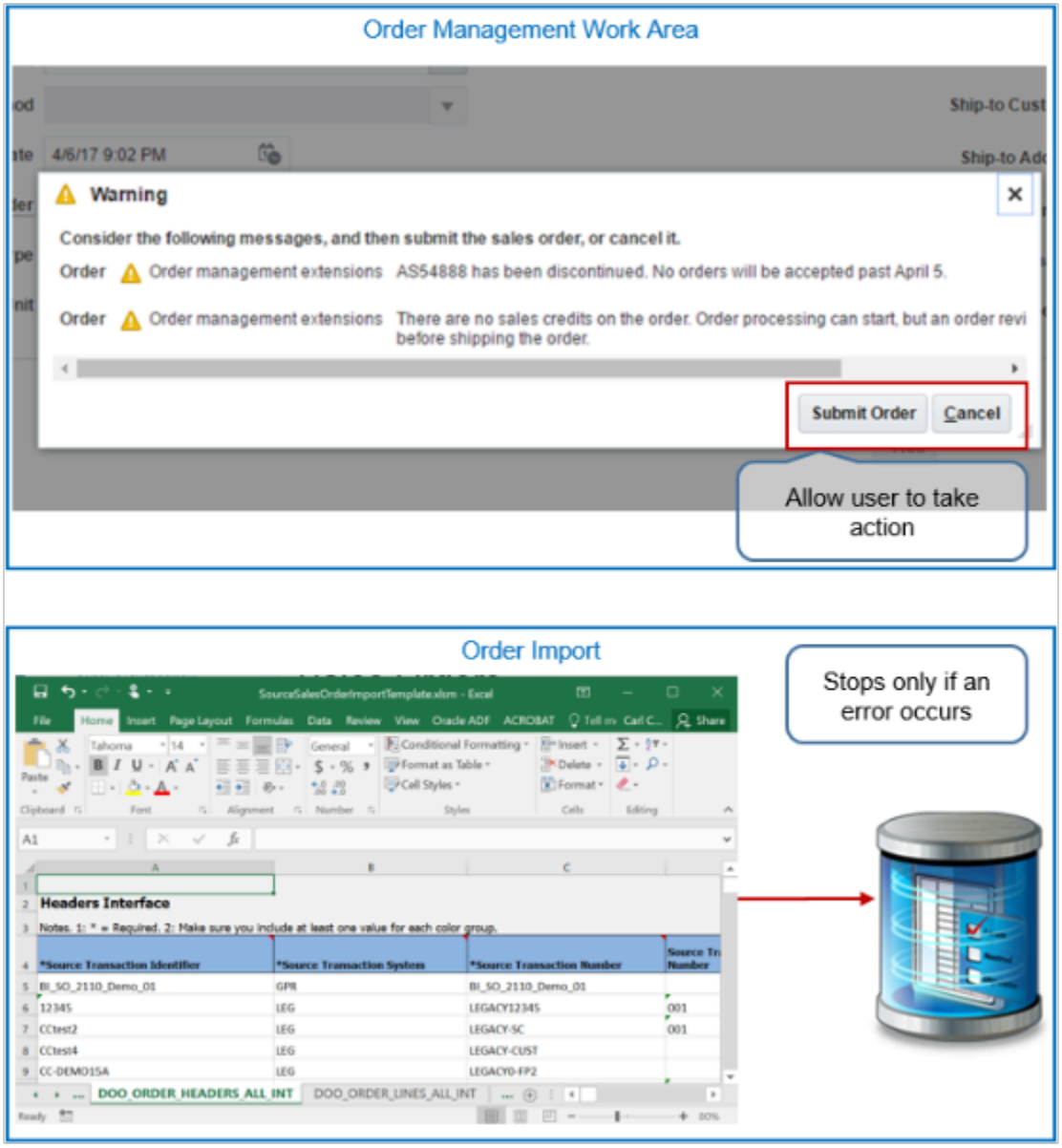
Code	Description
`List<Message> msgs = new ArrayList<Message>();`	Use this code. • **`List<Message> msgs`**. Define a local variable named **`msgs`** as a list type. Use this technique to add more than one message into the list so ValidationException can create all messages at one time. You typically use this technique in a longer extension that contains more than one condition. The extension code stops when it displays the first message, so you can use ValidationException to let all validations run, then display a dialog that includes all errors. • **`new ArrayList<Message>()`**. Define an array that uses list format. You will use this array to store messages that the **`while`** loop creates.
The **`while`** loop	Iterate through each sales credit until it finishes processing all of them. Examine attributes SalesCreditTypeCode, Percent, and Salesperson for each sales credit.
`def percent = salesCredit.getAttribute("Percent`	Define a local variable named **`percent`**, use method getAttribute to get the value of attribute Percent from local variable salesCredit, then set this variable to the value that getAttribute returns.
`if ( percent < 30 )`	If the condition is True, then create a message, and add it to the local **`msgs`** list. Messages accumulate in the **`msgs`** list until the extension finishes processing all sales credits.
`def tokens`	Define these tokens. • SALESPERSON. **`def tokens`** sets the value of this token to the value that method getAttribute gets for attribute Salesperson from local variable salesCredit. • PERCENT. **`def tokens`** sets the value of this token to the value of variable **`percent`** that the **`while`** loop defines.
`Message msg = new Message`	Define a local variable named **`msg`** and add it to local array **`Message`**. **`msg`** stores each message that the code adds to list object **`msgs`**.
`(Message.MessageType.ERROR, "SALES_CREDIT_TOO_LOW_MSG", tokens);`	Use this code. • **`Message.`** Use method Message to define a message. • **`MessageType.ERROR`**. Set the Message type to ERROR. • **`SALES_CREDIT_TOO_LOW_MSG`**. Get message text details from message SALES_CREDIT_TOO_LOW_MSG from page Manage Message, then add this text to local variable **`msg`**. • **`tokens`**. Get the value of local variable tokens, then add it to local variable **`msg`** immediately after message text SALES_CREDIT_TOO_LOW_MSG.
`msgs.add(msg);`	Add the contents of local variable **`msg`**, which contains a single message, to local list variable **`msgs`**, which is an array that includes all messages that the code creates in this scenario.
`if( !msgs.isEmpty() ) { throw new ValidationException(msgs); }`	If local list **`msgs`** isn't empty, then run method ValidationException. Display all the messages that local list **`msgs`** contains. Perform this check after the while loop finishes processing all sales credits.

Here's the complete code without comments.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
def salesCredits = header.getAttribute("SalesCredits");
List<Message> msgs = new ArrayList<Message>();
while( salesCredits.hasNext() ) {
 def salesCredit = salesCredits.next();
 if( "1".equals(salesCredit.getAttribute("SalesCreditTypeCode")) ) {
 def percent = salesCredit.getAttribute("Percent");
 if ( percent < 30 ) {
def tokens = [SALESPERSON: salesCredit.getAttribute("Salesperson"), PERCENT: percent];
Message msg = new Message(Message.MessageType.ERROR, "SALES_CREDIT_TOO_LOW_MSG", tokens);
msgs.add(msg);
 }
 }
}
if( !msgs.isEmpty() ) {
 throw new ValidationException(msgs);
}
```

Consider Order Import Behavior

Import behavior it different depending on whether your extension creates a warning message or error message.



Note

- **Sales orders entered in the Order Management work area**. The Warning dialog in work area Order Management allows the Order Entry Specialist to examine the warning, then submit the sales order or cancel.
- **Source orders imported from a source system**. Order processing stops only if an error happens. If only warnings happen, then you can view the messages but the import submits the sales order to order fulfillment. It doesn't allow user intervention.

Related Topics

- Overview of Creating Order Management Extensions
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions
- Manage Connector Details Between Order Management and Your Fulfillment System

Use Extensions to Manage Messages

Uses messages in your extension to get and display data.

Manage Validation Exceptions

To display a validation error, Order Management creates a validation exception, stops running the extension that failed, then runs a subsequent extension for the extension point that you set up.

Order Management accumulates each message that it creates from validation exceptions for all the extensions that it runs during the extension point, then displays them together to the Order Entry Specialist or, if its a source order from a source system, returns them through a web service call in the response to the caller.

If Order Management creates a validation exception, then your extension code must send a message that describes the cause of the failure. You must do one of:

- **Add the message text to your extension**. You must specify Distributed Order Orchestration as the application for the message. Order Management will display this text in the Order Management work area without modification.
- **Add the name of an Oracle message and message parameters in your extension**. Order Management will get the message from the Oracle message repository, populate any message tokens that the message references, display the message in the Order Management work area, then add it to the message log. You can use the Manage Messages page to create a message. For details, see *Set Up Messages in Order Management*.

This topic frequently mentions ValidationException. For details, see *Methods That You Can Use with Order Management Extensions*.

Use this code to create a validation exception.

Code	Description
`import oracle.apps.scm.doo.common.extens`	Import the ValidationException class.
`def salesCredits = header.getAttribute("SalesCredits`	Get the row set for the sales credits that the order header references.
`while( salesCredits.hasNext() )`	Determine whether more sales credit rows exist.
`def salesCredit = salesCredits.next();`	Get the next sales credit row.
`if( "1".equals(salesCredit.getAtt`	Determine whether the sales credit is a revenue percent.

Code	Description
`def percent = salesCredit.getAttribute("Percent`	Get the percent allocation of the sales credit.
`if( percent < 30 ) {`	If percent is less than 30, then create a validation error.
`def tokens = [SALESPERSON: salesCredit.getAttribute("Salespe PERCENT: percent];`	Specify the token values to send to ValidationException.
`throw new ValidationException("SALES_ CREDIT_TOO_LOW_MSG", tokens);`	Create the exception, and then stop the execution.

Here's the same code without comments.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def salesCredits = header.getAttribute("SalesCredits");
while( salesCredits.hasNext() ) {
def salesCredit = salesCredits.next();
if( "1".equals(salesCredit.getAttribute("SalesCreditTypeCode")) ) {
 def percent = salesCredit.getAttribute("Percent");
 if( percent < 30 ) {
def tokens = [SALESPERSON: salesCredit.getAttribute("Salesperson"), PERCENT: percent];
 throw new ValidationException("SALES_CREDIT_TOO_LOW_MSG", tokens);
 }
 }
}
```

Hard Coding the Message

Here's a variation that creates a validation exception that hard codes the message text. You replace `def tokens` and `throw new` with this code.

```
String messageText = "The " + percent + "% sales credit for salesperson " + salesperson + " is less than the
 required minimum, which is 30%.";
throw new ValidationException(messageText);
```

For example, if `percent` equals 20, and if `salesperson` equals Diane Cho, then here's the text that displays at run time.

```
The 20% sales credit for salesperson Diane Cho is less than the required minimum, which is 30%.
```

Reference Request Functions

Here's a variation that creates a validation exception that references a request function that you define instead of the default function that comes predefined with Order Management. You replace `def tokens` and `throw new` with this code.

```
def tokens = [SALESPERSON: salesCredit.getAttribute("Salesperson"), PERCENT: percent];
throw new ValidationException("ORA_CUSTOM_REQ_FUNCTION", "SALES_CREDIT_TOO_LOW_MSG", tokens);
```

Handle an Unexpected Exception

Handle an unexpected exception during the On Save extension point.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Person;
```

```
String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("UnexpectedExceptionOnSave_run_extension")) return;

List < Message > messages = new ArrayList < Message > ();

//Test DOO:::DOO_CX_EXECUTION_ERROR for NoDataFoundException when setAttribute on header
//An error occurred when running extension **, during event **: JBO-25002: Definition ** of type Attribute
 is not found..
//header.setAttribute("NonExistAttribute", "anyvalue"); //An error occurred when running extension
 UnexpectedExceptionOnSave, during event On Save: JBO-25058: Definition NonExistAttribute of type Attribute
 is not found in Header..
//header.getAttribute("HeaderAddresses"); //HeaderEffCategories

//Test DOO:::DOO_CX_ATTRIB_NO_DATA
//Expected message: A value could not be assigned to attribute {ATTRIBUTE}, while running extension
 {EXTENSION}, during event {EVENT}, because no data was found.
//Acutally got: An error occurred when running extension UnexpectedExceptionOnSave,
 during event On Save: oracle.jbo.JboException: JBO-29000: Unexpected exception caught:
 oracle.apps.scm.doo.common.extensions.NoDataFoundException, msg=No data was found using the provided
 parameters.
def person = new Person("First Name", "Last Name");
header.setBillToAccount(person);

//header.setAttribute("FreezePriceFlag", "1231"); //Error message: Value 100 for field FreezePriceFlag
 exceeds the maximum length allowed.

//def lines = header.getAttribute("Lines");
//if (lines.hasNext()) {
//def line = lines.next();
//Test DOO:::DOO_CX_ATTRIB_NO_DATA for NoDataFoundException when setAttribute on line
//line.setAttribute("NonExistAttribute", "anyvalue");

//Test DOO:::DOO_CX_CONN_NOT_FOUND for ServiceDetailNotFound when invokeSoapService
//def itemNumber = line.getAttribute("ProductNumber");
//String payLoad = "<ns1:GetSalesCreditAllocation xmlns:ns1=\"http://www.yourCompany.com/SalesCreditWS/\">"
 +
//"<ns1:poNumber>" + poNumber + "</ns1:poNumber>" +
//"<ns1:itemNumber>" + itemNumber + "</ns1:itemNumber>" +
//"</ns1:GetSalesCreditAllocation>";
//Correct integration name is SalesCreditAllocationService
//def responseBody = (context.invokeSoapService("SalesCreditAllocationService", payLoad)).getSoapBody();
////def serviceInvoker = context.getServiceInvoker();
////def responseBody = (serviceInvoker.invokeSoapService("SalesCreditAllocationService",
 payLoad)).getSoapBody();
//messages.add(responseBody);


//}

//ValidationException ex = new ValidationException(messages);
//throw ex;
```

If your billing application already authorized payment, then run this extension to prevent the user from editing payment attributes when revising a sales order. Run it on the On End of Submission Request extension point.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

import oracle.apps.scm.doo.common.extensions.Message;

if (!"PreAuthExtension".equals(header.getAttribute("CustomerPONumber"))) return;

String messageText = "The request to revise the sales order failed because the billing application already
 authorizedpayment for the order. ";
```

```
List < Message > messages = new ArrayList < Message > ();

if (header.getAttribute("ChangeVersionNumber") == null || header.getAttribute("ChangeVersionNumber") <= 1)
 return;

def paymentRows = header.getAttribute("Payments");
def isPreAuth = false;

if (paymentRows.hasNext()) {
 def paymentRow = paymentRows.next();
 isPreAuth = ("Y".equals(paymentRow.getAttribute("AuthorizedInSourceFlag")));
}
if (isPreAuth) {
 //Check OrderTotal
 if(null != header.getAttribute("ReferenceHeaderId")) {
 def origTotal = getPayNowTotalByHeaderId(header.getAttribute("ReferenceHeaderId"));
 def newTotal = getPayNowTotalByHeaderRow(header);
 if(newTotal > origTotal ) {
 messages.add(new Message(Message.MessageType.ERROR, messageText + "The total for the sales total is greater
 than the original order. The original total is :" + origTotal));
 ValidationException ex = new ValidationException(messages);
 throw ex;
 }
 }
def lines = header.getAttribute("Lines");
 while (lines.hasNext()) {
 def line = lines.next();
 def refFulfillLineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");
 if (line.getAttribute("ModifiedFlag").equals("Y")) {
 //Line is modified - compare the bill to with that of the base row.
 def refLine = getRefLine(refFulfillLineId);
 if (null != refLine && !
refLine.getAttribute("FulfillLineBillToCustomerId").equals(line.getAttribute("BillToCustomerIdentifier"))
 ||
 !
refLine.getAttribute("FulfillLineBillToSiteUseId").equals(line.getAttribute("BillToAccountSiteUseIdentifier")))
 {
 messages.add(new Message(Message.MessageType.ERROR, "You can't modify billing details for source
 transaction SourceTransactionLineNumber " + line.getAttribute("SourceTransactionLineNumber")". Discard
 your revision, create a new one, don't modify any attributes on the Billing and Payment Details tab, don't
 modify bill-to details on the order line, then submit your sales order."));
 ValidationException ex = new ValidationException(messages);
 throw ex;
 }
 }
 }
}


Object getRefLine(Long refFulfillLineId) {
 def fLinePVO = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");
 def vc = fLinePVO.createViewCriteria();
 def vcrow = vc.createViewCriteriaRow();
 vcrow.setAttribute("FulfillLineId", refFulfillLineId);

 def rowset = fLinePVO.findByViewCriteria(vc, -1);
 def line = rowset.next();
 return line;
}

Object getPayNowTotalByHeaderId(Long headerId) {
def totalPVO = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.OrderTotalPVO");
def vc = totalPVO.createViewCriteria();
def vcrow = vc.createViewCriteriaRow();
vcrow.setAttribute("OrderTotalHeaderId", headerId);
vcrow.setAttribute("OrderTotalTotalCode", "QP_TOTAL_PAY_NOW");
```

```
 def rowset = totalPVO.findByViewCriteria(vc, -1);
 def total = rowset.next();
 if(null != total)
 return total.getAttribute("OrderTotalTotalAmount");
 else
 return 0;
}

Object getPayNowTotalByHeaderRow(def headerRow) {
def totals = header.getAttribute("OrderTotals");
 while (totals.hasNext()) {
 def total = totals.next();
 if(total.getAttribute("TotalCode").equals("QP_TOTAL_PAY_NOW"))
 return total.getAttribute("TotalAmount");
 }
 return 0;
}
```

Display More Than One Validation Message

In the example above, the extension stops running as soon as it finds one sales credit record that doesn't meet the requirement, then reports the error. This extension code doesn't create or use a Message object but instead works directly against the ValidationException class.

An extension can create one or more instances of a Message object before calling ValidationException. Use this capability to report all sales credit rows that fail the requirement, and to control how you code your extension.

Here's an example that uses the Message object and provides detailed error reporting. It reports all sales credit rows that fail validation.

Code	Description
`import oracle.apps.scm.doo.common.extens`	Import the ValidationException class so you can construct a ValidationMessage object.
`import oracle.apps.scm.doo.common.extens`	Import the Message class so you can construct a Message object.
`def salesCredits = header.getAttribute("SalesCredits`	Get the row set for the sales credits that the order header references.

`List<Message> msgs = new ArrayList<Message>();`
 	
`while( salesCredits.hasNext() )`
	Determine whether more sales credit rows exist that we must process.

`def salesCredit = salesCredits.next();`	Get the next sales credit row.
`if( "1".equals(salesCredit.getAtt` `{`	Determine whether the sales credit is a revenue percent.
`def percent = salesCredit.getAttribute("Percent`	Get the percent allocation of the sales credit.

Code	Description
`if( percent < 30 ) {`	If percent is less than 30, then create a validation error.
`def tokens = [SALESPERSON: salesCredit.getAttribute("Salespe PERCENT: percent];`	Specify the token values to send to ValidationException.
`Message msg = new Message(Message.MessageType.ERROR "SALES_CREDIT_TOO_LOW_MSG", tokens);`	Create a new message.
`msgs.add(msg);`	Accumulate each message in a list.
`if( !msgs.isEmpty() ) {`	If the **msgs** list isn't empty, then at least one sales credit row is in error.
`throw new ValidationException(msgs);`	Create and throw ValidationException.

Here's the same code without comments.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
def salesCredits = header.getAttribute("SalesCredits");
List<Message> msgs = new ArrayList<Message>();
while( salesCredits.hasNext() ) {
 def salesCredit = salesCredits.next();
 if( "1".equals(salesCredit.getAttribute("SalesCreditTypeCode")) ) {
 def percent = salesCredit.getAttribute("Percent");
 if( percent < 30 ) {
def tokens = [SALESPERSON: salesCredit.getAttribute("Salesperson"), PERCENT: percent];
Message msg = new Message(Message.MessageType.ERROR, "SALES_CREDIT_TOO_LOW_MSG", tokens);
msgs.add(msg);
 }
 }
}
if( !msgs.isEmpty() ) {
 throw new ValidationException(msgs);
}
```

Display a Warning

This example displays a warning when fulfillment can't meet the order date. It uses the server time to determine whether fulfillment can ship the item according to the order entry date. If it can't, then display a warning.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

//Get the sold-to customer.
String customer = header.getAttribute("BuyingPartyName");

//If cusomer is not PMC - Snow Enterprise then we do not want to check for the order date
if( customer != "PMC - Snow Enterprise" ) return;

//Initialize the variable that indicates whether the sales order is past the cut off time to false.
boolean orderAfterCutOffTime = false;
```

```
//Initialize calendar object. The calendar has the current system time, by default. In this example, we want
 to set up the calendar so it's in Eastern Stardard Time.
Calendar now = new GregorianCalendar(TimeZone.getTimeZone("EST"));

//The following commented out code line is just for debugging purposes. It prints out the customer, and hour
 of day values in the shipping instructions attribute which is visible in the UI.
//This is an easy way to inspect variable values and debug code.
 header.setAttribute("ShippingInstructions", customer + ", " + now.get(Calendar.HOUR_OF_DAY) + ", " +
 now.get(Calendar.MINUTE));

CutOffHour = 10
CurrentHour = now.get(Calendar.HOUR_OF_DAY)

if( CurrentHour > CutOffHour ) {
 orderAfterCutOffTime = true;
}

//We will iterate through the lines in the order to see whether the ship from organization contains Vision
 Operations on any of the order lines. This will save some CPU cycles.
//We'll iterate through lines only if the order is past the cut-off time.
if( orderAfterCutOffTime ) {
 //get the lines iterator
 count = 0

 def lines = header.getAttribute("Lines");//get the lines row set
 while( lines.hasNext() ) {//if there are more order lines
 def line = lines.next();
 count = count +1;
 String shipFromOrgName = line.getAttribute("FulfillmentOrganizationName");

 if( shipFromOrgName != null) {
 if( shipFromOrgName != "Vision Operations" ){
 msg = new Message(Message.MessageType.WARNING, "This order has been entered after the cut off time " +
 CutOffHour + " and will not be shipped tomorrow, current hour is : " + CurrentHour + " !!!!!");
 throw new ValidationException(msg);
 }
 }
 }
}
```

Get and Display Transactional Data

This example uses messages to get and display tax data for a sales order.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

if (!"OrderTaxDetails_run_extension".equals(header.getAttribute("CustomerPONumber"))) return;

List < Message > messages = new ArrayList < Message > ();

messages.add(new Message(Message.MessageType.ERROR, "Status Code is " + header.getAttribute("StatusCode")));

def lines = header.getAttribute("Lines");
while (lines.hasNext()) {
 def line = lines.next();
 def charges = line.getAttribute("OrderCharges");
 while (charges.hasNext()) {
 def charge = charges.next();
 def chargeComponents = charge.getAttribute("OrderChargeComponents");
 while (chargeComponents.hasNext()) {
 def chargeComponent = chargeComponents.next();
 def taxDetails = chargeComponent.getAttribute("OrderTaxDetails");
 while (taxDetails.hasNext()) {
```

```
 def taxDetail = taxDetails.next();
 messages.add(new Message(Message.MessageType.ERROR, "OrderChargeComponentId is " +
 taxDetail.getAttribute("OrderChargeComponentId")));
 messages.add(new Message(Message.MessageType.ERROR, "TaxRate is " + taxDetail.getAttribute("TaxRate")));
 messages.add(new Message(Message.MessageType.ERROR, "TaxIncludedFlag is " +
 taxDetail.getAttribute("TaxIncludedFlag")));
 messages.add(new Message(Message.MessageType.ERROR, "HeaderCurrencyTaxUnitAmount is " +
 taxDetail.getAttribute("HeaderCurrencyTaxUnitAmount"))); //Not HdrCurrTaxUnitAmt
 messages.add(new Message(Message.MessageType.ERROR, "HeaderCurrencyTaxableUnitAmount is " +
 taxDetail.getAttribute("HeaderCurrencyTaxableUnitAmount"))); //Not HdrCurrTaxableUnitAmt
 messages.add(new Message(Message.MessageType.ERROR, "TaxRateIdentifier is " +
 taxDetail.getAttribute("TaxRateIdentifier"))); //Not TaxRateId
 messages.add(new Message(Message.MessageType.ERROR, "OrderTaxDetailId is " +
 taxDetail.getAttribute("OrderTaxDetailId")));
 messages.add(new Message(Message.MessageType.ERROR, "TaxExemptReasonCode is " +
 taxDetail.getAttribute("TaxExemptReasonCode")));
 messages.add(new Message(Message.MessageType.ERROR, "TaxExemptionCertificateNumber is " +
 taxDetail.getAttribute("TaxExemptionCertificateNumber"))); //Not TaxExemptCertificateNumber
 }
 }
 }

}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Get and Display Document References

This example uses messages to get and display document references for a sales order.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

String poNumber = header.getAttribute("CustomerPONumber");
if (poNumber == null) return;
if (!poNumber.startsWith("DocReferences")) return;

List < Message > messages = new ArrayList < Message > ();
messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + header.getAttribute("HeaderId")));
messages.add(new Message(Message.MessageType.ERROR, "Pre-submit"));

def docReferences = header.getAttribute("DocumentReferences");
while (docReferences.hasNext()) {
 def docRef = docReferences.next();
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalNumber: " +
 docRef.getAttribute("DocumentAdditionalNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionaldentifier: " +
 docRef.getAttribute("DocumentAdditionaldentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentIdentifier: " +
 docRef.getAttribute("DocumentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineNumber: " +
 docRef.getAttribute("DocumentAdditionalLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineIdentifier" +
 docRef.getAttribute("DocumentAdditionalLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineIdentifier: " +
 docRef.getAttribute("DocumentLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineNumber: " +
 docRef.getAttribute("DocumentLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentReferenceType: " +
 docRef.getAttribute("DocumentReferenceType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineNumber: " +
 docRef.getAttribute("DocumentAdditionalSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineIdentifier: " +
 docRef.getAttribute("DocumentAdditionalSubLineIdentifier")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineIdentifier: " +
 docRef.getAttribute("DocumentSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineNumbe: " +
 docRef.getAttribute("DocumentSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentNumber: " +
 docRef.getAttribute("DocumentNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineIdentifier: " +
 docRef.getAttribute("FulfillLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "LineIdentifier: " +
 docRef.getAttribute("LineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableId: " +
 docRef.getAttribute("OwnerTableId")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableName: " +
 docRef.getAttribute("OwnerTableName")));
 messages.add(new Message(Message.MessageType.ERROR, "TaskType: " + docRef.getAttribute("TaskType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSystemReferenceIdentifier: " +
 docRef.getAttribute("DocumentSystemReferenceIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + docRef.getAttribute("HeaderId")));
}

def lines = header.getAttribute("Lines");
messages.add(new Message(Message.MessageType.ERROR, "get lines"));
while (lines.hasNext()) {
 messages.add(new Message(Message.MessageType.ERROR, "A line"));
 def line = lines.next();
 def lineDocReferences = line.getAttribute("DocumentReferences");
 while (lineDocReferences.hasNext()) {
 messages.add(new Message(Message.MessageType.ERROR, "has doc references"));
 def lineDocRef = lineDocReferences.next();
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionaldentifier: " +
 lineDocRef.getAttribute("DocumentAdditionaldentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentIdentifier: " +
 lineDocRef.getAttribute("DocumentIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalLineIdentifier" +
 lineDocRef.getAttribute("DocumentAdditionalLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineIdentifier: " +
 lineDocRef.getAttribute("DocumentLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentLineNumber: " +
 lineDocRef.getAttribute("DocumentLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentReferenceType: " +
 lineDocRef.getAttribute("DocumentReferenceType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineNumber: " +
 lineDocRef.getAttribute("DocumentAdditionalSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentAdditionalSubLineIdentifier: " +
 lineDocRef.getAttribute("DocumentAdditionalSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineIdentifier: " +
 lineDocRef.getAttribute("DocumentSubLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSubLineNumbe: " +
 lineDocRef.getAttribute("DocumentSubLineNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentNumber: " +
 lineDocRef.getAttribute("DocumentNumber")));
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineIdentifier: " +
 lineDocRef.getAttribute("FulfillLineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "LineIdentifier: " +
 lineDocRef.getAttribute("LineIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableId: " +
 lineDocRef.getAttribute("OwnerTableId")));
 messages.add(new Message(Message.MessageType.ERROR, "OwnerTableName: " +
 lineDocRef.getAttribute("OwnerTableName")));
 messages.add(new Message(Message.MessageType.ERROR, "TaskType: " + lineDocRef.getAttribute("TaskType")));
 messages.add(new Message(Message.MessageType.ERROR, "DocumentSystemReferenceIdentifier: " +
 lineDocRef.getAttribute("DocumentSystemReferenceIdentifier")));
```

```
 messages.add(new Message(Message.MessageType.ERROR, "HeaderId: " + lineDocRef.getAttribute("HeaderId")));
 }
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Test a Transaction Attribute

This example uses messages to test a transaction attribute during the On Save extension point.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;

if (!"TestTIAOnSave_run_extension".equals(header.getAttribute("CustomerPONumber"))) return;

String poNumber = header.getAttribute("CustomerPONumber");
//Long headerId = header.getAttribute("HeaderId");

List < Message > messages = new ArrayList < Message > ();
/*messages.add(new Message( Message.MessageType.ERROR, "Enter TIA On Save"));
messages.add(new Message( Message.MessageType.ERROR, "CustomerPONumber: " + poNumber));
messages.add(new Message( Message.MessageType.ERROR, "HeaderId: " + headerId));*/

def lines = header.getAttribute("Lines");
def i = 0;
while (lines.hasNext()) {
 def line = lines.next();
 def tias = line.getAttribute("TransactionAttributes");
 while (tias.hasNext()) {
 messages.add(new Message(Message.MessageType.ERROR, "FulfillLineId: " +
 line.getAttribute("FulfillmentLineIdentifier")));
 def tia = tias.next();
 messages.add(new Message(Message.MessageType.ERROR, "TransactionAttributeIdentifier: " +
 tia.getAttribute("TransactionAttributeIdentifier")));
 messages.add(new Message(Message.MessageType.ERROR, "TransactionAttributeCode: " +
 tia.getAttribute("TransactionAttributeCode")));
 messages.add(new Message(Message.MessageType.ERROR, "TransactionAttributeName: " +
 tia.getAttribute("TransactionAttributeName")));
 messages.add(new Message(Message.MessageType.ERROR, "CharacterValue: " +
 tia.getAttribute("CharacterValue")));
 messages.add(new Message(Message.MessageType.ERROR, "NumberValue: " + tia.getAttribute("NumberValue")));
 messages.add(new Message(Message.MessageType.ERROR, "DateValue: " + tia.getAttribute("DateValue")));
 messages.add(new Message(Message.MessageType.ERROR, "TimestampValue: " +
 tia.getAttribute("TimestampValue")));
 /*
 //tia.setAttribute("TransactionAttributeCode", "Color"); //TransactionAttributeIdentifier will be
 300100061374755
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 TransactionAttributeCode: " + tia.getAttribute("TransactionAttributeCode")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 TransactionAttributeName: " + tia.getAttribute("TransactionAttributeName")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 TransactionAttributeIdentifier: " + tia.getAttribute("TransactionAttributeIdentifier")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 CharacterValue: " + tia.getAttribute("CharacterValue")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 NumberValue: " + tia.getAttribute("NumberValue")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 DateValue: " + tia.getAttribute("DateValue")));
 //messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeCode to Color,
 TimestampValue: " + tia.getAttribute("TimestampValue")));

 //tia.setAttribute("CharacterValue", "Purple");
```

```
//messages.add(new Message( Message.MessageType.ERROR, "After setting CharacterValue to Purple,
CharacterValue: " + tia.getAttribute("CharacterValue")));

//tia.setAttribute("TransactionAttributeName", "zcz colors 2"); //TransactionAttributeIdentifier will be
300100033383860, code: zcz_colors_2
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, TransactionAttributeCode: " + tia.getAttribute("TransactionAttributeCode")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, TransactionAttributeName: " + tia.getAttribute("TransactionAttributeName")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, TransactionAttributeIdentifier: " + tia.getAttribute("TransactionAttributeIdentifier")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, CharacterValue: " + tia.getAttribute("CharacterValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, NumberValue: " + tia.getAttribute("NumberValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, DateValue: " + tia.getAttribute("DateValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeName to zcz
colors 2, TimestampValue: " + tia.getAttribute("TimestampValue")));

//tia.setAttribute("TransactionAttributeIdentifier", "300100039021944"); //mapping to name:
zCZ_FRAME_COLOR, display name: Frame Color
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, TransactionAttributeCode: " + tia.getAttribute("TransactionAttributeCode")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, TransactionAttributeName: " + tia.getAttribute("TransactionAttributeName")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, TransactionAttributeIdentifier: " + tia.getAttribute("TransactionAttributeIdentifier")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, CharacterValue: " + tia.getAttribute("CharacterValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, NumberValue: " + tia.getAttribute("NumberValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, DateValue: " + tia.getAttribute("DateValue")));
//messages.add(new Message( Message.MessageType.ERROR, "After setting TransactionAttributeIdentifier to
300100039021944, TimestampValue: " + tia.getAttribute("TimestampValue")));

tia.setAttribute("TransactionAttributeIdentifier", "300100005319663"); //set back to the original
TransactionAttributeIdentifier
tia.setAttribute("CharacterValue", "BLUE");
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, TransactionAttributeCode: " + tia.getAttribute("TransactionAttributeCode")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, TransactionAttributeName: " + tia.getAttribute("TransactionAttributeName")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, TransactionAttributeIdentifier: " + tia.getAttribute("TransactionAttributeIdentifier")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, CharacterValue: " + tia.getAttribute("CharacterValue")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, NumberValue: " + tia.getAttribute("NumberValue")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, DateValue: " + tia.getAttribute("DateValue")));
messages.add(new Message( Message.MessageType.ERROR, "After resetting TransactionAttributeIdentifier to
300100005319663, TimestampValue: " + tia.getAttribute("TimestampValue")));*/
 }
}

ValidationException ex = new ValidationException(messages);
throw ex;
```

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions
- Entities That You Can Use With Order Management Extensions

See How Long it Takes for Your Extension to Finish

Your order management extension might result in taking a long time to save or submit a sales order.

For example, a complex extension might integrate with a service that has a slow response. To troubleshoot this problem, you can use this code in your extension and measure how much time it takes to finish running:

```
logger = context.getLogger();
def startTime = context.getCurrentTime();
try {
 /*your custom code*/
}finally {
 logDuration(startTime);
}
/*your custom functions*/
void logDuration(java.sql.Timestamp startTime) {
 def endTime = context.getCurrentTime();
 long diff = endTime.getTime() - startTime.getTime();
 logger.logSevere( "ExtensionDurationCheck;" +
context.getExtensionName() + " " + header.getAttribute("OrderNumber") + ";" + diff.toString());
}
```

If it exceeds a time threshold, then you can end the extension, or least examine your logs to identify which extension is taking a long time to finish, for which sales order, then take corrective action.

Consider this code.

```
OALOMExtensionDurationCheck;extension_name;extension_runtime_duration.
```

where

- Order Management replaces `extension_name` with the name of your extension.
- Order Management replaces `extension_runtime_duration` with the time in hour, minutes and seconds.

This code makes sure that Order Management creates a log for your extension even when you set it at a SEVERE level.

Assume the code adds these details to your log.

```
'OALOMExtensionDurationCheck;CopyOrderFields for Return Orders;122'
```

It means an extension named `CopyOrderFields for Return Orders` took 122 milliseconds to finish.

Examine Your Log

As an alternative, you can examine your log. Consider an example extension that runs on the On End of Submission Request extension point.

1. Set the logging profile to FINER.
2. Create and submit a sales order.

3. Open the log for your extension and examine the results.

```
Finished invoking extensions for event ON_END_SUBMIT in 5ms
```

Here's an example log. Search it for the text `in 5ms`.

```
scm.doo.common.extensions.model.applicationModule.DooCommonExtensionsAMImpl] [SRC_CLASS:
 oracle.apps.fnd.applcore.log.AppsODLHandler] [SRC_METHOD: publish] Invoking ON_END_SUBMIT Extensions for
 headerId: 300100185658909
[2020-03-27T20:24:14.877+00:00] [UIServer_2] [TRACE:16] [] [oracle.apps.appslogger] [tid:
 130] [userId: ORDER_ENTRY_SPEC] [ecid: 005cTjY2YVbDg^O5Ijo2yf00055I0001Ye,0:4] [APP:
 ORA_FSCM_UIAPP#V2.0.n-TV_j4ycNhG-GhU36AdAA] [APPS_USER_NAME: ORDER_ENTRY_SPEC] [APPS_SESSION_ID:
 A1D8F6426C73FA3CE0530503F40A6116] [APPS_THREAD_NAME: [ACTIVE].ExecuteThread: '20' for queue:
 'weblogic.kernel.Default (self-tuning)'] [APPS_TERRITORY: US] [APPS_AUTO_LOG: false] [APPS_USER_ID:
 27E079FDC4D0AE55B8F4F424AD9940EA] [APPS_DB_CONNECTION_URL: jdbc:oracle:thin:@FA_DEFAULT] [APPS_SOURCE:
 scm.doo.orchInfra.protectedModel.extensionsFwk.applicationModule.ExtensionsRuntimeAMImpl] [SRC_CLASS:
 oracle.apps.fnd.applcore.log.AppsODLHandler] [SRC_METHOD: publish] Entering invokeExtensions - eventCode:
 ON_END_SUBMIT
[2020-03-27T20:24:14.881+00:00] [UIServer_2] [TRACE] [] [oracle.apps.appslogger] [tid:
 130] [userId: ORDER_ENTRY_SPEC] [ecid: 005cTjY2YVbDg^O5Ijo2yf00055I0001Ye,0:4] [APP:
 ORA_FSCM_UIAPP#V2.0.n-TV_j4ycNhG-GhU36AdAA] [APPS_USER_NAME: ORDER_ENTRY_SPEC] [APPS_SESSION_ID:
 A1D8F6426C73FA3CE0530503F40A6116] [APPS_THREAD_NAME: [ACTIVE].ExecuteThread: '20' for queue:
 'weblogic.kernel.Default (self-tuning)'] [APPS_TERRITORY: US] [APPS_AUTO_LOG: false] [APPS_USER_ID:
 27E079FDC4D0AE55B8F4F424AD9940EA] [APPS_DB_CONNECTION_URL: jdbc:oracle:thin:@FA_DEFAULT] [APPS_SOURCE:
 scm.doo.orchInfra.protectedModel.extensionsFwk.applicationModule.ExtensionsRuntimeAMImpl] [SRC_CLASS:
 oracle.apps.fnd.applcore.log.AppsODLHandler] [SRC_METHOD: publish] Finished invoking extension Default EFF
 in 0ms
[2020-03-27T20:24:14.882+00:00] [UIServer_2] [TRACE] [] [oracle.apps.appslogger] [tid:
 130] [userId: ORDER_ENTRY_SPEC] [ecid: 005cTjY2YVbDg^O5Ijo2yf00055I0001Ye,0:4] [APP:
 ORA_FSCM_UIAPP#V2.0.n-TV_j4ycNhG-GhU36AdAA] [APPS_USER_NAME: ORDER_ENTRY_SPEC] [APPS_SESSION_ID:
 A1D8F6426C73FA3CE0530503F40A6116] [APPS_THREAD_NAME: [ACTIVE].ExecuteThread: '20' for queue:
 'weblogic.kernel.Default (self-tuning)'] [APPS_TERRITORY: US] [APPS_AUTO_LOG: false] [APPS_USER_ID:
 27E079FDC4D0AE55B8F4F424AD9940EA] [APPS_DB_CONNECTION_URL: jdbc:oracle:thin:@FA_DEFAULT] [APPS_SOURCE:
 scm.doo.orchInfra.protectedModel.extensionsFwk.applicationModule.ExtensionsRuntimeAMImpl] [SRC_CLASS:
 oracle.apps.fnd.applcore.log.AppsODLHandler] [SRC_METHOD: publish] Finished invoking extensions for event
 ON_END_SUBMIT in 5ms
[2020-03-27T20:24:14.882+00:00] [UIServer_2] [TRACE:16] [] [oracle.apps.appslogger] [tid:
 130] [userId: ORDER_ENTRY_SPEC] [ecid: 005cTjY2YVbDg^O5Ijo2yf00055I0001Ye,0:4] [APP:
 ORA_FSCM_UIAPP#V2.0.n-TV_j4ycNhG-GhU36AdAA] [APPS_USER_NAME: ORDER_ENTRY_SPEC] [APPS_SESSION_ID:
 A1D8F6426C73FA3CE0530503F40A6116] [APPS_THREAD_NAME: [ACTIVE].ExecuteThread: '20' for queue:
 'weblogic.kernel.Default (self-tuning)'] [APPS_TERRITORY: US] [APPS_AUTO_LOG: false] [APPS_USER_ID:
 27E079FDC4D0AE55B8F4F424AD9940EA] [APPS_DB_CONNECTION_URL: jdbc:oracle:thin:@FA_DEFAULT]
 [APPS_SOURCE: scm.doo.common.extensions.model.applicationModule.DooCommonExtensionsAMImpl] [SRC_CLASS:
 oracle.apps.fnd.applcore.log.AppsODLHandler] [SRC_METHOD: publish] Exiting invokeExtensions
```

Use a Warning Message

As another alternative you can use a warning message.

Here's an example extension that measures the time that the extension runs, then uses the messaging framework to display it in an Oracle warning message.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
import oracle.apps.scm.doo.common.extensions.Message;
import oracle.apps.scm.doo.common.extensions.Message;
List<Message> messages = new ArrayList<Message>();
 def starttime = context.getCurrentTime(); //get current time

//assume you want to get the order lines that are in the sales order
def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();

 // do whatever processing you need
```

```
}

def endtime = context.getCurrentTime(); // assuming that this is the end of your macro, register the time

 // this example code creates a warning message named TEST_LOGGING_MESSAGE. It uses 3 tokens: start time,
 end time, and extension name.

def tokens = [START_TIME: starttime.toString(), END_TIME: endtime.toString(), EXTENSION_NAME:
 context.getExtensionName()]; //create

Message logMessage = new Message(Message.MessageType.WARNING, "ORA_MANAGE_EXTENSIONS",
 "TEST_LOGGING_MESSAGE",tokens);
messages.add(logMessage);
ValidationException ex = new ValidationException(messages);

 // if you want to print only the time, do this:

 //def printstring = "Start time : " + starttime.toString() + "\n" + "End time : " + endtime.toString();
//ValidationException ex = new ValidationException(printstring);
//ValidationException ex = new ValidationException("ORA_MANAGE_EXTENSIONS", "TEST_LOGGING_MESSAGE", tokens);

throw ex;
```

Reference

Methods That You Can Use with Order Management Extensions

Get details about the methods that you can use with your order management extension.

canApplyHold Method

canApplyHold makes sure that the line isn't a child line of a model or kit, and that you haven't already applied a hold on the line.

```
boolean canApplyHold(def line, def holdCode) {

 if (isChildLine(line)) {
 debug("is child line");
 return false;
 }

 if (holdExists(line, holdCode)) {
 debug("hold already exists");
 return false;
 }

 return true;
}
```

canApplyHold returns a value.

- true. The line isn't a child line of a model or kit, and you haven't already applied a hold on the line.
- false. The line is a child line of a model or kit, or you already applied a hold on the line.

For details, see *Use Order Management Extensions to Apply Holds*.

Debug Method

Use the `debug` method to get details that can help you troubleshoot any problems that might come up when you use an extension to apply a hold. For details, see the Use the Debug Method subtopic in *Test Your Order Management Extension*.

Execution Context Method

The ExecutionContext method calls a SOAP web service.

Use this format.

```
SoapServiceResponse invokeSoapService(String integrationName, String xmlStr)
```

where

- **integrationName**. Identifies the name of the integration.
- **xmlStr**. XML representation of the SOAP body in String format.

ExecutionContext returns details.

Parameter	Description
`public String getExtensionName()`	String. Name of the order management extension that's currently running.
`public String getEventCode()`	String. Abbreviation that identifies the event that triggered the extension.
`public Logger getLogger()`	Logger. Logger where the extension can write logs.
`public ServiceInvoker getServiceInvoker()`	Returns a service that you can use to call a web service from the code in your order management extension.
`public String getUserName()`	Returns the name of the user currently signed in.
`public boolean isUserInRole(String roleName)`	Returns one of. • **True**. The user currently signed in is using the role that roleName specifies. • **False**. The user isn't using this role. where • **roleName**. A string that identifies the role name.
`public String getLanguage()`	Returns an abbreviation that identifies the language that the user is using in the current session.
`public Date getCurrentDate()`	Returns the current system date in an instance of java.sql.Date.
`public Timestamp getCurrentTime()`	Returns the current system time in a java.sql.Timestamp object.

Get Attachments Method

Use the getAttachments() method to get these attributes:

Entity	Value to Use in Code	Parent	Read During All Events	Write During Save or Start of Submission Request	Write During End of Submission Request
Header attachment	Not applicable	Header	Yes	Yes	Yes
Fulfillment line attachment	Not applicable	Lines	Yes	Yes	No

holdExists Method

See whether you already applied a hold on the line.

```
boolean holdExists(def line, def holdCode) {
 ArrayList holds = line.getHolds();
 for (hold in holds) {
 if (holdCode.equals(hold.getAttribute("HoldCode"))) {
 debug("Found hold code on line");
 return true;
 }
 }
 return false;
}
```

holdExists returns a value.

- true. You already applied a hold on the line.
- false. You haven't applied a hold on the line.

isChildLine Method

See whether the line is a child of a model or kit.

```
boolean isChildLine(def line) {
 def rootParentLineId = line.getAttribute("RootParentLineReference");
 def fulfillLineId = line.getAttribute("FulfillmentLineIdentifier");
 return (rootParentLineId != null && rootParentLineId != fulfillLineId);
}
```

isChildLine returns a value.

- true. The line is a child line of a model or kit.
- false. The line isn't a child line of a model or kit.

isFirstDraftOrder Method

Make sure the sales order is in draft status.

```
boolean isFirstDraftOrder() {
 String statusCode = header.getAttribute("StatusCode");
 if ("DOO_DRAFT".equals(statusCode)) {
 def cvn = header.getAttribute("ChangeVersionNumber");
```

```
    if (cvn == 1) {
    return true;
    }
    }

    return false;
   }
```

isFirstDraftOrder returns a value.

- true. The sales order is in draft status.
- false. The sales order isn't in draft status.

Logger Method

The Logger method returns the message log.

Use this format.

Format	Description
`public boolean isEnabled(Level level)`	Returns one of. • **True**. Logging is enabled at the level that `Level` specifies. • **False**. Logging isn't enabled at this level. where • **level**. Sets the logging level according to a constant from java.util.logging.Level.
`public void logFine(String message)`	Saves `message` to the log at the fine level. where • `message`. A text string that the command saves to the log.
`public void logFine(String messageFormat, Object... arguments)`	Saves the message at the fine level. where • **messageFormat**. Format of the message. The command uses this format to create the message string. It passes the format and `arguments`to String.format(). • **arguments**. Arguments that replace the tokens in the format string.
`public void logFiner(String message)`	Saves `message` to the log at the finer level. where • `message`. A text string that the command saves to the log.
`public void logFiner(String messageFormat, Object... arguments)`	Saves the message at the finer level. where • **messageFormat**. Format of the message. The command uses this format to create the message string. It passes the format and `arguments`to String.format(). • **arguments**. Arguments that replace the tokens in the format string.

Format	Description
`public void logFinest(String message)`	Saves **message** to the log at the finest level. where • **message**. A text string that the command saves to the log.
`public void logFinest(String message)`	Saves **message** to the log at the finest level. where • **message**. A text string that the command saves to the log.
`public void logSevere(String messageFormat, Object... arguments)`	Saves the message at the severe level. where • **messageFormat**. Format of the message. The command uses this format to create the message string. It passes the format and **arguments** to String.format(). • **arguments**. Arguments that replace the tokens in the format string.
`public void logSevere(String messageFormat, Object... arguments)`	Saves the message at the severe level. where • **messageFormat**. Format of the message. The command uses this format to create the message string. It passes the format and **arguments** to String.format(). • **arguments**. Arguments that replace the tokens in the format string.

Message Method

The Message method uses a message type and message text to create a message for the sales order. It uses the ORA_MANAGE_EXTENSIONS request function to log messages, by default.

Here's the format to use to display a literal message that uses the default message category and no tokens.

```
Message(MessageType type, String text)
```

where

- **type**. Type of message. You can use ERROR or WARNING.
- **text**. Text of the message to display, enclosed with one set of double quotation marks (" ").

Here's an example warning message that includes a literal string.

```
Message(MessageType Message.MessageType.WARNING, String "This is the warning message.")
```

Here's an error message that includes a literal string.

```
Message(MessageType Message.MessageType.ERROR, String "This is the error message.")
```

Specify Message Name and Tokens

Here's the format to use to specify the message name and message token parameters.

```
Message(MessageType type, String name, Map<String, Object> parameters)
```

where

- **type**. Type of message. You can use ERROR or WARNING.
- **name**. Name of a message from the Manage Messages page.

  Here are the values that attributes for this message must use on the Manage Messages page.

Attribute	Value
Application	Distributed Order Orchestration
Module	Distributed Order Orchestration

  For details, see *Set Up Messages in Order Management*.
- **parameters**. A map that contains message tokens and token values. The Message method uses this map to populate tokens with values. These tokens are part of the message.

Here's some example code that displays the contents of a message named DOO_CUST_RELATIONSHIP_WARNING.

```
Message(MessageType Message.MessageType.WARNING, String "DOO_CUST_RELATIONSHIP_WARNING", Map<String, Object>
 messageParams)
```

Create Messages for Sales Orders and Sales Order Lines

You can use the Request function, message name, and message token parameters to create a message for the sales order or the order line. The Message method will log the message with the request function that you specify.

Here's the format to use for the sales order.

```
Message(MessageType type, String requestFunction, String name, Map string, object msgParams)
```

Here's the format to use for the order line.

```
Message(MessageType type, String requestFunction, Object line, String name, Map string, object msgParams)
```

where

- **Object line**. Identifies the order line to reference when logging the message.

For details about the other parameters, see the Map Tokens for the Message Repository section in this topic.

Here's some example code that uses a request function.

```
Message(MessageType Message.MessageType.WARNING, String "DEMO_REQFUNC", String
 "DOO_CUST_RELATIONSHIP_WARNING", Map string, object msgParams)
```

where

- DEMO_REQFUNC is a value in lookup DOO_MSG_REQUEST_FUNCTION.

Row Set Iterator Method

The RowSetIterator method manages rows in the iterator.

Use this format.

```
Row first()
```

RowSetIterator returns details.

Parameter	Description
`Row first()`	Returns the first row in the iterator.
`hasNext()`	Returns a Boolean value that indicates whether another row exists after the current row.
`Row next()`	Returns the next row in the iterator. If no more rows exist after the current row, then this parameter returns a null value.
`Row last()`	Returns the last row in the iterator.
`hasPrevious()`	Returns a Boolean value that indicates whether another row exists before the current row.
`Row previous()`	Returns the previous row in the iterator. If no more rows exist before the current row, then this parameter returns a null value.
`createRow()`	Creates a new row. If the entity doesn't support creating a new row, then this parameter returns an error message when your extension calls this method.
`insertRow(Row row)`	Inserts the row into the current iterator, where row specifies the row number to insert. Use this method to add a new row to the iterator after you create the iterator.

Service Invoker Method

The ServiceInvoker method calls a service.

Validation Exception Method

The ValidationException method uses the error text that you specify to create a validation exception. It displays the message text that you specify.

Use this format.

```
ValidationException(String messageText)
```

Display Messages from the Message Repository

ValidationException can get the error message text from the Oracle Applications message repository, populate token values, and then display the message in the Order Management work area.

Use this format.

```
ValidationException(String name, Map string, object parameters)
```

where

- **name**. Name of the message that you specify on the Manage Messages page.
- **parameters**. A map that contains message tokens and token values. ValidationException uses this map to populate tokens with values in the message.

Map Tokens for the Message Repository

Here's the format to use with ValidationException to create the map that specifies.

```
ValidationException(java.lang.String requestFunction, java.lang.String name,
 java.util.Map<java.lang.String,java.lang.Object> parameters)
```

where

- **requestFunction**. Lookup code of the request function to log the message against. You use the Order Management lookup named Request Function to define a request function. If this parameter passes an empty value, then ValidationException defaults the request function to ORA_MANAGE_EXTENSIONS.
- **name**. Name of the message that you specify on the Manage Messages page. You must specify Distributed Order Orchestration as the application for this message when you use the Manage Messages page.
- **parameters**. A map that contains message tokens and token values. ValidationException uses this map to populate tokens with values in the message.

Display Message Objects from the Message Repository

ValidationException can use a list of message objects to create a validation exception. It can accommodate a validation error that includes more than one message.

Use this format.

```
ValidationException(List message messages)
```

where

- **messages**. A list of message objects. For details about how to create a message object, see the Message Method section in this topic.

Use Reference Lookup Codes

Here's the format you use to reference a lookup code.

```
throw new ValidationException("lookup_code")
```

The lookup type named DOO_MSG_REQUEST_FUNCTION contains lookup codes.

Lookup Code	Description
ASSIGN_PROC	Process assignment.
MANAGE_TASK	Fulfillment task.
MANAGE_HOLD	Hold.
MANAGE_PROC	Fulfillment process.

Lookup Code	Description
PLAN_PROC	Fulfillment planning.
PROC_CHANGE	Change order.
XFORM_ORDER	Order transformation. Location that Order Management uses to store an incoming source order before it transforms it to a sales order.
VALD_CONSTRAINT	Processing constraint.
VALD_PROC_DEFN	Process definition.
VALD_ACTION_ELGB	Action eligibility.
ORA_VALD_ORDER_DATA	Order validation.
ORA_VALD_CONFIG	Configuration.
ORA_VALD_PRICING	Pricing.
ORA_NOTIFY_EVENT	Event notification.
ORA_VALD_PAYMENT	Payment.
ORA_VERIFY_TRADE_COMPLIANCE	Verify that fulfillment lines meet trade compliance policies.
ORA_CREDIT_CHECK	Verify credit check failure.
ORA_ORDER_APPROVALS	Message function classification for Order Approval messages.
ORA_MANAGE_EXTENSIONS	Default type for messages that the extensions framework logs.

You can specify one of these lookup codes or add new ones to the lookup.

View Object Method

The ViewObject method finds and returns a view criterion.

Use this format.

```
ViewCriteria getViewCriteria(name)
```

where

- **name**. Name of the view criterion.

A view criterion is a list of row criterion for the WHERE clause in a view object. Each row criterion is an array that contains criterion for each attribute.

Create View Criterion

Use this format.

```
ViewCriteria createViewCriteria()
```

ViewObject returns the new view criterion object.

Finding Rows According to View Criterion

Here's the format to use to find rows according to the view criterion that you provide and return them in an iterator.

```
RowIterator findByViewCriteria(ViewCriteria viewCriteria, int numberOfRows)
```

where

- **viewCriteria**. Name of the view criteria to use to find rows.
- **numberOfRows**. Number of rows to return. Use one of.
  - **-1**. Get all rows that match the view criteria.
  - **Positive integer**. Get a subset of rows. For example, use `3` to get the first three rows that match the criteria. If only two rows match the criteria, then the method returns only these two rows.

For example, here's some code that gets all rows and stores them in a local variable named `vc`.

```
def rowset = itemPVO.findByViewCriteria(vc, -1);
```

Here's the format to include an array of variable names and values.

```
RowIterator findByViewCriteriaWithBindVars(ViewCriteria viewCriteria, int maxNumOfRows, String[]
 variableNames, Object[] variableValues)
```

where

- **variableNames**. Array of variable names to use with the view criteria.
- **variableValues**. Array of variable values to use with the view criteria.

Here's the format to use to find rows according to the name of the view criteria.

```
RowIterator findByViewCriteriaWithBindVars(String viewCriteriaName, int maxNumOfRows, String[]
 variableNames, Object[] variableValues)
```

where

- **viewCriteriaName**. Name of the view criteria to use to find the rows.

For more, see *Public View Objects in Oracle Applications Cloud (Doc ID 2386411.1)*.

Address Methods

You can use these methods with address data.

1. public Address(String addressLine1, String addressLine2, String addressLine3, String addressLine4, String city, String state, String postalCode, String county, String province, String country)
2. public void setAddress1(String address1)
3. public String getAddress1() { return mAddress1; }

4. public void setAddress2(String address2)
5. public String getAddress2() { return mAddress2; }
6. public void setAddress3(String address3)
7. public String getAddress3() { return mAddress3; }
8. public void setAddress4(() {String address4; }
9. public String getAddress4() { return mAddress4; }
10. public void setCity(String city)
11. public String getCity() { return mCity; }
12. public void setPostalCode(String postalCode)
13. public String getPostalCode() {return mPostalCode; }
14. public void setState(String state)
15. public String getState() { return mState; }
16. public void setProvince(String province)
17. public String getProvince() { return mProvince; }
18. public void setCounty(String county)
19. public String getCounty() { return mCounty; }
20. public void setCountry(String country)
21. public String getCountry() { return mCountry; }

Attachment Methods

You can use these methods to manage attachments.

1. public String getDatatypeCode() { return mDatatypeCode; }
2. public void setDatatypeCode(String datatypeCode)
3. public void setAttachedDocumentId(Long attachedDocumentId)
4. public Long getAttachedDocumentId() { return mAttachedDocumentId; }
5. public void setCategoryName(String mCategoryName)
6. public String getCategoryName() { return mCategoryName; }
7. public void setDocumentAttributes(String documentAttributes)
8. public String getDocumentAttributes() { return mDocumentAttributes; }
9. public void setDescription(String description)
10. public String getDescription() { return mDescription; }
11. public void setEntityAttributes(String entityAttributes)
12. public String getEntityAttributes() { return mEntityAttributes; }
13. public void setEntityName(String entityName)
14. public String getEntityName() { return mEntityName; }
15. public void setFileContentType(String fileContentType)
16. public String getFileContentType() { return mFileContentType; }
17. public void setFileName(String fileName)
18. public String getFileName() { return mFileName; }
19. public void setPk1Value(String pk1Value)
20. public String getPk1Value() { return mPk1Value; }
21. public void setTitle(String title)
22. public String getTitle() { return mTitle; }
23. public void setUrl(String url)
24. public String getUrl() { return mUrl; }
25. public BlobDomain getFileContent() { return mFileContent; }
26. public void setFileContent(byte[] fileContent)
27. public void setFileContent(BlobDomain fileContent) { }
28. public String getText() { return mText; }
29. public void setText(String text)
30. public void setDocumentName(String documentId)

31. public String getDocumentName() { return mDocumentId; }
32. public void setDocumentIdentifier(String documentVersion)
33. public String getDocumentIdentifier() { return mDocumentVersion; }
34. public void setContentRepositoryFileShared(boolean contentRepositoryFileShared)
35.
```
public boolean isContentRepositoryFileShared() {
 return mContentRepositoryFileShared;
 }
```

CreateLineParams Methods

You can use these methods with the CreateLineParams operation.

1. public void setProductIdentifier(Long productIdentifier)
2. public void setProductNumber(String productNumber)
3. public String getProductNumber()
4. public void setOrderedUOMCode(String orderedUOMCode)
5. public String getOrderedUOMCode()
6. public void setOrderedUOM(String orderedUOM)
7. public String getOrderedUOM()
8. public void setOrderedQuantity(BigDecimal orderedQuantity)
9. public BigDecimal getOrderedQuantity()

Extensible Flexfield Methods

You can use these methods when your extension references an extensible flexfield on an order header or order line.

Method	Description
public getOrCreateContextRow(String contextCode)	Use this method to write a value to or to get a value from an extensible flexfield. It returns the value that's in the flexfield context. If the context doesn't already exist, then the method creates and returns a new context. Use the contextCode parameter on this method to get the flexfield context's code. If the context exists, then the method returns the context, else it creates a new one and returns it.
public Row getContextRow(String contextCode)	Does the same thing as getOrCreateContextRow but only gets the value that the extensible flexfield currently contains.

Order Header and Order Line Methods

You can use these methods on the order header or the order line.

Method	Description
public String getAttribute(String attributeName)	Get the value of any attribute that you specify in the method's attributeName parameter. Specify the attribute's name or the entity name in the attributeName parameter. You can also use getAttribute to get the value of these header entities: • DocumentReferences • HeaderAddresses

Method	Description
	• Lines • OrderTotals • SalesCredits It can return a collection of header entities.
public List getAttachments()	Get a list of attachments that are on the order header or order line.
public createAttachment(Attachment attachment)	Add an attachment to the order header or order line.
public deleteAttachment(Attachment attachment)	Delete an attachment from the order header or order line.
public duplicateAttachments(List attachmentList)	Duplicate the attachments that are on the order header or order line.
public Line createNewLine(CreateLineParams createLineParams)	Add a free item. It returns the line object of the line that it just added. If you use createNewLine on the: • Order header. It creates an order line that doesn't reference an item on another line. • Order line. It creates order line x that references an item on order line y. The item on line x is free.
public boolean isSellingProfitCenterUserUpdated()	Contains true or false. True: the user updated the SellingProfitCenter attribute. False: the user didn't update the SellingProfitCenter attribute.

Address Methods

The extension uses the address data that you specify in the method's parameter to find the attribute's value.

Method	Description
public void setShipToAddress(Address address)	Set the ship to address attribute on the order line.
public void setBillToAddress(Address address)	Set the bill to address attribute on the order line.
public void setSupplierAddress(Address address)	Set the supplier's address attribute on the order line.
public void setDestinationShippingAddress(Address address)	Set the destination shipping address attribute on the order line.
public void setFinalDischargeLocationAddress(Address address)	Set the final discharge address attribute on the order line.

Party Methods

The extension uses the person data that you specify in the method's parameter to find the attribute's value.

Method	Description
public void setShipToParty(Person person)	Set the ship to party attribute on the order line.
public void setShipToContact(Person person)	Set the ship to contact attribute on the order line.
public void setBillToAccount(Person person)	Set the bill to customer attribute on the order line.
public void setBillToContact(Person person)	Set the bill to contact attribute on the order line.

Order Line Methods

Method	Description
public String getAttribute(String attributeName)	Get the value of any attribute from the order line. Specify the name of the attribute in the attributeName parameter. You can also use the attributeName parameter to specify any of these entities that are in the line entity: • AssetTransactionDetails • BillingPlans • ChannelPrograms • DocumentReferences • FulfillLineDetails • Holds • LotSerial • ManualPriceAdjustments • OrderCharges • OrderTerms • Payments • Projects • SalesCredits • TransactionAttributes You can't use this method to get an extensible flexfield's value.
public setAttribute(String attributeName, Object value)	Set the value of an attribute on the order line. • Use the attributeName parameter to specify the attribute that you want to set. • Use the value parameter to specify the attribute's value.
public List getTransformedLines()	Get a list of the lines that the extension transformed for the current line. A transformed line is a line that contains a free item.

Method	Description
public boolean isTransformed()	Contains true or false. True: the order line is a transformed line. False: the order line isn't a transformed line.
public boolean isOpen()	Contains true or false. True: the order line is open. False: the order line isn't open.
public boolean isClosed()	Contains true or false. True: the order line is closed. False: the order line isn't closed.
public boolean isCanceled()	Contains true or false. True: the order line is cancelled. False: the order line isn't cancelled.
public List getHolds()	Get a list of the holds that are on the order line.
public Hold applyHold(String holdCode)	Apply a hold on the order line. This method also returns the hold object that it just added.

Person Methods

1. public Person(String firstName, String lastName)
2. public Person(String firstName, String middleName, String lastName)
3. public Person(String firstName, String lastName, String middleName, String suffix, String title)
4. public void setFirstName(String firstName)
5. public String getFirstName() {return mFirstName;}
6. public void setLastName(String lastName)
7. public String getLastName() {return mLastName;}
8. public void setMiddleName(String middleName
9. public String getMiddleName() {return mMiddleName; }
10. public void setSuffix(String suffix)
11. public String getSuffix() {return mSuffix;}
12. public void setTitle(String title)
13. public String getTitle() {return mTitle; }

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Entities That You Can Use With Order Management Extensions

Entities That You Can Use With Order Management Extensions

Get details about the entities that you can use with your order management extensions.

Each extension can read data from or write data to a sales order entity. Use the getAttribute() method to access them.

Entity	Value to Use in Code	Parent	Read During All Events	Write During Save or Start of Submission Request	Write During End of Submission Request
Order header	Header	Not applicable	Yes	Yes	No
Order line	Lines	Header	Yes	Yes	No

Entity	Value to Use in Code	Parent	Read During All Events	Write During Save or Start of Submission Request	Write During End of Submission Request
Fulfillment line detail	FulfillLineDetails	Lines	Yes	No	No
Sales credit	SalesCredits	Header, Lines	Yes	Yes	No
Order total	OrderTotals	Header	Yes	No	No
Manual price adjustment	ManualPriceAdjustmen	Lines	Yes	No	No
Order charge	OrderCharges	Lines	Yes	No	No
Order charge component	OrderChargeComponer	OrderCharges	Yes	No	No
Lot serial	LotSerial	Lines	Yes	No	No
Transactional attribute	TransactionAttributes	Lines	Yes	Yes	No
Payment	Payments	Headers, Lines	Yes	No	No
Billing plan	BillingPlans	Lines	Yes	No	No
Order tax detail	OrderTaxDetails	OrderChargeComponer	Yes	No	No
Document reference	DocumentReferences	Header, Lines	No	No	No
Project	Projects	Lines	Yes	Yes	No

Related Topics

- Overview of Creating Order Management Extensions
- Use Proven Coding Techniques
- Points Where You Can Run Order Management Extensions
- Methods That You Can Use with Order Management Extensions

Attributes That You Can Use With Order Management Extensions

Get details about the attributes that you can use with your order management extensions.

Overview

- Use the Line entity to get the value of the ItemSubTypeCode attribute.
- If you set a default value for the ShipSetName attribute, then you must also set the PartialShipAllowedFlag attribute to N.
- If you set the PartialShipAllowedFlag to Y, then you must make sure the line isn't part of a shipment set or kit.

The text `Identifier`, `Id`, or `ID` typically identifies an identifier attribute.

- An extension can read from or write to the attributes that identify data in a sales order, such as BillToCustomerIdentifier.
- Order Management converts each value to an identifier when it saves your code.
- Some identifier values use the Oracle Mobile Supply Chain identifier.

View Objects

- Your extension code can use the attribute name of the public view object.
- You can use a public view object, but it might be necessary for you to also use the identifier attributes.
- You can't use the public view object for an order line that the user adds in the Order Management work area before the user clicks Submit because Order Management hasn't yet saved it to the database.

The rest of this topic lists the attributes that you can use with your extension. To get them in an Excel file, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, then download the Attributes You Can Use With Extensions attachment.

Order Header Attributes

Attribute	Can Update
AgreementHeaderId	Yes
AgreementNumber	Yes
AllowCurrencyOverrideFlag	No
AppliedCurrencyCode	No
AppliedCurrencyName	No
BillToAccountContactIdentifier	Yes
BillToAccountContactName	Yes
BillToAccountContactNumber	Yes
BillToAccountDescription	No
BillToAccountPersonFirstName	No
BillToAccountPersonLastName	No
BillToAccountPersonMiddleName	No
BillToAccountPersonNameSuffix	No
BillToAccountPersonTitle	No

Attribute	Can Update
BillToAddress1	No
BillToAddress2	No
BillToAddress3	No
BillToAddress4	No
BillToCity	No
BillToContactFirstName	No
BillToContactLastName	No
BillToContactMiddleName	No
BillToContactNameSuffix	No
BillToContactTitle	No
BillToCountry	No
BillToCounty	No
BillToCustomerIdentifier	Yes
BillToCustomerName	Yes
BillToCustomerNumber	Yes
BillToCustomerSiteIdentifier	Yes
BillToPartyType	Yes
BillToPostalCode	No
BillToProvince	No
BillToState	No
BusinessUnitIdentifier	Yes
BusinessUnitName	Yes
BuyingPartyContactFirstName	No
BuyingPartyContactId	Yes
BuyingPartyContactLastName	No
BuyingPartyContactMiddleName	No
BuyingPartyContactName	Yes
BuyingPartyContactNameSuffix	No
BuyingPartyContactNumber	Yes
BuyingPartyContactTitle	No
BuyingPartyIdentifier	Yes

Attribute	Can Update
BuyingPartyName	Yes
BuyingPartyNumber	Yes
BuyingPartyPersonFirstName	No
BuyingPartyPersonLastName	No
BuyingPartyPersonMiddleName	No
BuyingPartyPersonNameSuffix	No
BuyingPartyPersonTitle	No
BuyingPartyType	Yes
CancelReason	Yes
CancelReasonCode	Yes
CarrierId	Yes
CarrierName	Yes
ChangeVersionNumber	No
Comments	Yes
CurrencyConversionDate	Yes
CurrencyConversionRate	Yes
CurrencyConversionTypeCode	Yes
CustomerPONumber	Yes
DemandClass	Yes
DemandClassCode	Yes
EarliestAcceptableShipDate	Yes
FOBPoint	Yes
FOBPointCode	Yes
FreezePriceFlag	Yes
FreezeShippingChargeFlag	Yes
FreezeTaxFlag	Yes
FreightTerms	Yes
FreightTermsCode	Yes
FulfillmentOrganizationCode	Yes
FulfillmentOrganizationIdentifier	Yes
FulfillmentOrganizationName	Yes

Attribute	Can Update
HeaderId	No
LatestAcceptArrivalDate	Yes
LatestAcceptableShipDate	Yes
ModifiedFlag	No
OpenFlag	Yes
OrderNumber	No
OrigSysDocumentRef	Yes
PackingInstructions	Yes
PartialShipAllowedFlag	Yes
PaymentTerm	Yes
PaymentTermCode	Yes
PreCreditCheckedFlag	Yes
PreferredBillToContactPointIdentifier	Yes
PreferredShipToContactPointIdentifier	Yes
PreferredSoldToContactPointIdentifier	Yes
PricedOn	No
PricingSegmentCode	No
PricingSegmentExplanation	No
PricingStrategy	No
PricingStrategyExplanation	No
PricingStrategyIdentifier	No
ReferenceHeaderId	No
RequestArrivalDate	Yes
RequestCancelDate	Yes
RequestShipDate	Yes
RequestingBusinessUnit	No
RequestingBusinessUnitIdentifier	Yes
RequestingLegalUnit	Yes
RequestingLegalUnitIdentifier	Yes
RevisionSourceOrderSystem	Yes
SalesChannel	Yes

Attribute	Can Update
SalesChannelCode	Yes
Salesperson	Yes
SalespersonIdentifier	Yes
SegmentExplanationMessage	No
SegmentExplanationMsgName	No
ShipClassOfService	Yes
ShipClassOfServiceCode	Yes
ShipModeOfTransport	Yes
ShipModeOfTransportCode	Yes
ShipToAddress1	No
ShipToAddress2	No
ShipToAddress3	No
ShipToAddress4	No
ShipToCity	No
ShipToContactFirstName	No
ShipToContactLastName	No
ShipToContactMiddleName	No
ShipToContactNameSuffix	No
ShipToContactTitle	No
ShipToCountry	No
ShipToCounty	No
ShipToPartyContactIdentifier	Yes
ShipToPartyContactName	Yes
ShipToPartyContactNumber	No
ShipToPartyIdentifier	Yes
ShipToPartyName	Yes
ShipToPartyNumber	Yes
ShipToPartyPersonFirstName	No
ShipToPartyPersonLastName	No
ShipToPartyPersonMiddleName	No
ShipToPartyPersonNameSuffix	No

Attribute	Can Update
ShipToPartyPersonTitle	No
ShipToPartySiteIdentifier	Yes
ShipToPartyType	Yes
ShipToPostalCode	No
ShipToProvince	No
ShipToState	No
ShipmentPriority	Yes
ShipmentPriorityCode	Yes
ShippingInstructions	Yes
ShipsetFlag	Yes
SoldToPartyContactPointId	Yes
SourceTransactionIdentifier	No
SourceTransactionNumber	No
SourceTransactionRevisionNumber	No
SourceTransactionSystem	No
StatusCode	No
StrategyExplanationMessage	No
StrategyExplanationMsgName	No
SubmittedBy	Yes
SubmittedDate	Yes
SubmittedFlag	No
SubstituteAllowedFlag	Yes
SupplierAddressCity	No
SupplierAddressCountry	No
SupplierAddressCounty	No
SupplierAddressLine1	No
SupplierAddressLine2	No
SupplierAddressLine3	No
SupplierAddressLine4	No
SupplierAddressPostalCode	No
SupplierAddressProvince	No

Attribute	Can Update
SupplierAddressState	No
SupplierCode	Yes
SupplierName	Yes
SupplierNumber	Yes
SupplierSiteIdentifier	Yes
TradeComplianceResult	Yes
TradeComplianceResultCode	No
TransactionDocumentTypeCode	Yes
TransactionOn	Yes
TransactionType	No
TransactionTypeCode	Yes
TransactionalCurrencyCode	Yes
TransactionalCurrencyName	Yes
SubEntity	-
DocumentReferences	No
Lines	No
OrderTotals	No
Payments	No
SalesCredits	No

Note

- CustomerPONumber displays as the Purchase Order Number attribute in the Order Management workarea.
- SoldToPartyContactPointId is the SOLD_TO_CONTACT_ID in the base table.
- SourceTransactionNumber displays as the Source Order attribute in the Order Management workarea.
- SupplierCode is the SUPPLIER_ID in the base table.
- TransactionOn displays as the Ordered Date attribute in the Order Management workarea and is ORDERED_DATE in the base table.
- TransactionTypeCode displays as the Order Type attribute in the Order Management workarea.

Order Line Attributes

Don't use the EarliestAcceptArrivalDate attribute or the EarliestAcceptArrivalDateTransient attribute in any of your extensions.

Attribute	Can Update
AccountingRule	Yes
AccountingRuleCode	Yes
ActionType	Yes
ActionTypeCode	Yes
AgreementHeaderId	Yes
AssessableValue	Yes
AssetGroupNumber	Yes
AssetTrackedFlag	Yes
AssetTransactionDetails	No
BillingPlans	No
BillingTransactionTypeIdentifier	Yes
BillToAccountContactIdentifier	Yes
BillToAccountContactName	Yes
BillToAccountContactNumber	Yes
BillToAccountPersonFirstName	No
BillToAccountPersonLastName	No
BillToAccountPersonMiddleName	No
BillToAccountPersonNameSuffix	No
BillToAccountPersonTitle	No
BillToAccountSiteUseIdentifier	Yes
BillToAddress1	No
BillToAddress2	No
BillToAddress3	No
BillToAddress4	No
BillToCity	No
BillToContactFirstName	No
BillToContactLastName	No
BillToContactMiddleName	No
BillToContactNameSuffix	No
BillToContactTitle	No
BillToCounty	No

Attribute	Can Update
BillToCustomerIdentifier	Yes
BillToCustomerName	Yes
BillToCustomerNumber	Yes
BillToPartyType	Yes
BillToPostalCode	No
BillToProvince	No
BillToState	No
BusinessUnitIdentifier	Yes
BusinessUnitName	Yes
CanceledFlag	No
CanceledQuantity	No
CancelReason	Yes
CancelReasonCode	Yes
ChannelPrograms	No
Comments	Yes
ComponentIdPath	Yes
ConfigHeaderId	Yes
ConfigRevisionNumber	Yes
ConfiguratorPath	Yes
ContractStartDate	Yes
ContractEndDate	Yes
ContractStartDateTime	Yes
ContractEndDateTime	Yes
CreditCheckAuthorizationCode	Yes
CreditCheckAuthorizationExpiryDate	Yes
CustomerPOLineNumber	Yes
CustomerPONumber	Yes
CustomerPOScheduleNumber	Yes
CustomerProductDescription	Yes
CustomerProductIdentifier	Yes
CustomerProductNumber	Yes

Attribute	Can Update
DefaultTaxationCountry	Yes
DefaultTaxationCountryShortName	Yes
DemandClass	Yes
DemandClassCode	Yes
DemandSourceLineReference	Yes
DemandSourceScheduledFlag	No
DestinationShippingAddressCity	No
DestinationShippingAddressCountry	No
DestinationShippingAddressCounty	No
DestinationShippingAddressLine1	No
DestinationShippingAddressLine2	No
DestinationShippingAddressLine3	No
DestinationShippingAddressLine4	No
DestinationShippingAddressPostalCode	No
DestinationShippingAddressProvince	No
DestinationShippingAddressState	No
DestinationShippingLocationIdentifier	Yes
DestinationShippingOrganizationCode	Yes
DestinationShippingOrganizationIdentifier	Yes
DestinationShippingOrganizationName	Yes
DisplayLineNumber	No
DocumentReferences	No
DocumentSubType	Yes
DocumentSubTypeName	Yes
EarliestAcceptableShipDate	Yes
EndCreditMethod	Yes
EndCreditMethodCode	Yes
EndDate	Yes
EndReason	Yes
EndReasonCode	Yes
ExtendedAmount	Yes

Attribute	Can Update
ExternalPriceBookName	Yes
FinalDischargeLocationAddressCity	No
FinalDischargeLocationAddressCountry	No
FinalDischargeLocationAddressCounty	No
FinalDischargeLocationAddressLine1	No
FinalDischargeLocationAddressLine2	No
FinalDischargeLocationAddressLine3	No
FinalDischargeLocationAddressLine4	No
FinalDischargeLocationAddressPostalCode	No
FinalDischargeLocationAddressProvince	No
FinalDischargeLocationAddressState	No
FinalDischargeLocationIdentifier	Yes
FirstPartyTaxRegistration	Yes
FirstPartyTaxRegistrationNumber	Yes
FOBPoint	Yes
FOBPointCode	Yes
FreightTerms	Yes
FreightTermsCode	Yes
FulfilledQuantity	No
FulfillInstanceId	Yes
FulfillLineDetails	No
FulfillmentLineIdentifier	No
FulfillmentOrganizationCode	Yes
FulfillmentOrganizationIdentifier	Yes
FulfillmentOrganizationName	Yes
FulfillmentSplitReferenceId	No
FulfillToleranceAbove	Yes
FulfillToleranceBelow	Yes
FulfillUpdateParentRefId	No
HeaderId	No
Holds	No

Attribute	Can Update
IntegrateSubscriptionFlag	No
IntendedUseClassificationIdentifier	Yes
IntendedUseClassificationName	Yes
InventoryInterfacedFlag	No
InventoryOrganization	No
InventoryOrganizationIdentifier	No
InventoryTransactionFlag	Yes
InvoiceInterfacedFlag	No
InvoicingRule	Yes
InvoicingRuleCode	Yes
IsValidConfiguration	Yes
ItemSubTypeCode	No
LatestAcceptableArrivalDate	Yes
LatestAcceptableShipDate	Yes
LineAgreementNumber	Yes
LineId	No
LineNumber	No
LineType	No
LotSerial	Yes
ManualPriceAdjustments	No
ModifiedFlag	Yes
OnHoldFlag	No
OpenFlag	No
OrchestrationProcessId	Yes
OrchestrationProcessName	Yes
OrderCharges	No
OrderedQuantity	Yes
OrderedUOM	Yes
OrderedUOMCode	Yes
OrderTerms	No
OriginalProductDescription	Yes

Attribute	Can Update
OriginalProductIdentifier	Yes
OriginalProductNumber	Yes
OverrideScheduleDateFlag	Yes
PackingInstructions	Yes
ParentLineReference	Yes
PartialShipAllowedFlag	Yes
Payments	No
PaymentTerm	Yes
PaymentTermCode	Yes
PreferredBillToContactPointIdentifier	Yes
PreferredShipToContactPointIdentifier	Yes
PriceUsingSecondaryUOMFlag	No
ProductCategory	Yes
ProductCategoryName	Yes
ProductDescription	Yes
ProductFiscalCategoryIdentifier	Yes
ProductFiscalCategoryName	Yes
ProductIdentifier	Yes
ProductNumber	Yes
ProductType	Yes
ProductTypeName	Yes
ProjectRecordIndicator	Yes
Projects	No
PromiseArrivalDate	Yes
PromiseShipDate	Yes
QuantityPerModel	Yes
ReferenceFulfillmentLineIdentifier	No
RemnantFlag	No
RequestCancelDate	Yes
RequestedArrivalDate	Yes
RequestedRatePlanId	Yes

Attribute	Can Update
RequestedRatePlanNumber	Yes
RequestedShipDate	Yes
RequestingBusinessUnitIdentifier	Yes
RequestingBusinessUnitName	Yes
RequiredFulfillmentDate	Yes
ReturnReason	Yes
ReturnReasonCode	Yes
RMADeliveredQuantity	No
RootParentLineReference	Yes
SalesCredits	No
Salesperson	Yes
SalespersonIdentifier	Yes
SalesProductType	Yes
SalesProductTypeCode	Yes
ScheduleArrivalDate	Yes
ScheduleShipDate	Yes
SecondaryOrderedQuantity	No
SecondaryRmaDeliveredQuantity	No
SecondaryShippedQuantity	No
SecondaryUOM	No
SecondaryUOMCode	No
SellingProfitCenter	Yes
SellingProfitCenterBuId	Yes
ServiceCancelDate	Yes
ServiceDuration	Yes
ServiceDurationPeriod	Yes
ServiceDurationPeriodCode	Yes
ShipmentPriority	Yes
ShipmentPriorityCode	Yes
ShippedQuantity	No
ShippingCarrier	Yes

Attribute	Can Update
ShippingCarrierCode	Yes
ShippingInstructions	Yes
ShippingMode	Yes
ShippingModeCode	Yes
ShippingServiceLevel	Yes
ShippingServiceLevelCode	Yes
ShipSetName	Yes
ShipToAddress1	No
ShipToAddress2	No
ShipToAddress3	No
ShipToAddress4	No
ShipToCity	No
ShipToContactFirstName	No
ShipToContactId	Yes
ShipToContactLastName	No
ShipToContactMiddleName	No
ShipToContactNameSuffix	No
ShipToContactTitle	No
ShipToCounty	No
ShipToPartyContactIdentifier	Yes
ShipToPartyContactName	Yes
ShipToPartyContactNumber	Yes
ShipToPartyIdentifier	Yes
ShipToPartyName	Yes
ShipToPartyNumber	Yes
ShipToPartyPersonFirstName	No
ShipToPartyPersonLastName	No
ShipToPartyPersonMiddleName	No
ShipToPartyPersonNameSuffix	No
ShipToPartyPersonTitle	No
ShipToPartySiteIdentifier	Yes

Attribute	Can Update
ShipToPartyType	Yes
ShipToPostalCode	No
ShipToProvince	No
ShipToRequestRegion	Yes
ShipToState	Yes
SourceTransactionIdentifier	No
SourceTransactionLineIdentifier	No
SourceTransactionLineNumber	No
SourceTransactionNumber	No
SourceTransactionRevisionNumber	No
SourceTransactionScheduleIdentifier	No
SourceTransactionScheduleNumber	No
SourceTransactionSystem	No
StatusCode	No
SubEntity	
SubInventoryCode	Yes
Subinventoryidentifier	Yes
SubscriptionInterfacedFlag	No
SubscriptionProfileIdentifier	Yes
SubscriptionProfileName	Yes
SubstitutionAllowedFlag	Yes
SubstitutionReason	Yes
SubstitutionReasonCode	Yes
SupplierAddressCity	No
SupplierAddressCountry	No
SupplierAddressCounty	No
SupplierAddressLine1	No
SupplierAddressLine2	No
SupplierAddressLine3	No
SupplierAddressLine4	No
SupplierAddressPostalCode	No

Attribute	Can Update
SupplierAddressProvince	No
SupplierAddressState	No
SupplierCode	Yes
SupplierName	Yes
SupplierNumber	Yes
SupplierSiteIdentifier	Yes
SupplyStatusCode	Yes
TaxClassification	Yes
TaxClassificationCode	Yes
TaxExemptFlag	Yes
TaxExemptionCertificateNumber	Yes
TaxExemptReason	Yes
TaxExemptReasonCode	Yes
TaxInvoiceDate	Yes
TaxInvoiceNumber	Yes
ThirdPartyTaxRegistration	Yes
ThirdPartyTaxRegistrationNumber	Yes
TotalContractAmount	Yes
TotalContractQuantity	Yes
TransactionAttributes	Yes
TransactionBusinessCategory	Yes
TransactionBusinessCategoryName	Yes
TransactionCategoryCode	Yes
TransactionLineType	Yes
TransactionLineTypeCode	Yes
UnitListPrice	Yes
UnitQuantity	Yes
UnitSellingPrice	Yes
UnreferencedReturnFlag	No
UserDefinedFiscClass	Yes
UserDefinedFiscClassName	Yes

Note

- FulfillmentLineIdentifier is FULFILL_LINE_ID in the base table.
- LotSerial returns an iterator that contains the lot serials for the order line.
- You can use OrchestrationProcessName only with REST API.
- OrderCharges returns an iterator that contains the pricing charges for the line.
- Payments returns an iterator that contains the payments for the line.
- SourceTransactionNumber displays as the Source Order Number attribute in the Order Management workarea.

Fulfillment Line Attributes

Attribute	Updatable
ActualDeliveryDate	No
AvailabilityShipDate	No
BillingTransactionAmount	No
BillingTransactionDate	No
BillingTransactionNumber	No
BillOfLadingNumber	No
CustomerTrxLineIdentifier	No
DeliveryName	No
ExceptionFlag	No
FulfillLineDetailId	No
FulfillLineId	No
Quantity	No
RMAReceiptDate	No
RMAReceiptLineNumber	No
RMAReceiptNumber	No
RMAReceiptTransactionIdentifier	No

Attribute	Updatable
SecondaryQuantity	No
Status	No
StatusAsOfDate	No
TaskType	No
TrackingNumber	No
TradeComplianceCode	No
TradeComplianceExplanation	No
TradeComplianceName	No
TradeComplianceResultCode	No
TradeComplianceResultName	No
TradeComplianceTypeCode	No
TradeComplianceTypeName	No
WaybillNumber	No

Order Total Attributes

Attribute	Can Update
CurrencyCode	No
DisplayName	No
EstimatedFlag	No
HeaderId	No
OrderTotalId	No
PrimaryFlag	No
TotalAmount	No
TotalCode	No

Attribute	Can Update
TotalGroup	No

Order Charge Attributes

Attribute	Can Update
ApplyTo	No
AverageUnitSellingPrice	No
BlockAllowance	No
BlockSize	No
CanAdjustFlag	No
ChargeCurrencyCode	No
ChargeCurrencyName	No
ChargeDefinition	No
ChargeDefinitionCode	No
ChargeSubtype	No
ChargeSubtypeCode	No
ChargeType	No
ChargeTypeCode	No
GSAUnitPrice	No
OrderChargeId	No
ParentEntityId	No
PricePeriodicityCode	No
PriceType	No
PriceTypeCode	No
PricedQuantity	No
PricedQuantityUOM	No
PricedQuantityUomCode	No
PrimaryFlag	No
RollupFlag	No
SequenceNumber	No
SourceChargeIdentifier	No

Attribute	Can Update
TierAggregationMethodCode	No
TierAppliesTo	No
TierAppliesToCode	No
TierBasisTypeCode	No
TieredFlag	No
UsagePriceLockFlag	No
SubEntity	
OrderChargeComponents	No
OrderChargeTiers	No

Charge Component Attributes

Attribute	Can Update
HeaderCurrencyCode	No
HeaderCurrencyDurationExtendedAmount	No
HeaderCurrencyExtendedAmount	No
HeaderCurrencyName	No
HeaderCurrencyUnitPrice	No
OrderChargeComponentId	No
OrderChargeId	No
PercentOfComparisonElement	No
PriceElement	No
PriceElementCode	No
PriceElementUsage	No
PriceElementUsageCode	No
RollupFlag	No
SequenceNumber	No
SourceChargeComponentId	No
SourceChargeId	No
SourceMpaId	No
SourceParentChargeComponentId	No

Attribute	Can Update
TaxIncludedFlag	No
SubEntity	-
OrderTaxDetails	No

Note

- OrderTaxDetails returns an iterator that contains tax details for the charge component.

Tax Attributes

Attribute	Can Update
HeaderCurrencyTaxUnitAmount	No
HeaderCurrencyTaxableUnitAmount	No
OrderChargeComponentId	No
OrderTaxDetailId	No
TaxExemptReason	No
TaxExemptReasonCode	No
TaxExemptionCertificateNumber	No
TaxIncludedFlag	No
TaxRate	No
TaxRateIdentifier	No

Billing Plan Attributes

Attribute	Can Update
BillingNumberOfPeriods	No
BillingPeriodEndDate	No
BillingPeriodNumber	No
BillingPeriodStartDate	No
BillingPlanId	No
BillingPlanTypeCode	No
BillingTrxDate	No
CancellationEffectiveDate	No
FulfillLineId	No

Attribute	Can Update
NetBillingAmountPERPeriod	No
OverridePeriod	No
OverridePeriodAmount	No
OverridePeriodQuantity	No
PeriodicityCode	No
PeriodicityName	No
SourceBillingPlanId	No

Sales Credit Attributes

Attribute	Can Update
FulfillLineId	No
HeaderId	No
Percent	Yes
SalesCreditIdentifier	No
SalesCreditType	Yes
SalesCreditTypeCode	Yes
Salesperson	Yes
SalespersonIdentifier	Yes
SourceTransactionSalesCreditIdentifier	Yes

Payment Attributes

Attribute	Updatable
AuthorizationStatus	No
InstrumentAssignmentIdentifier	No
PaymentMethod	No
PaymentMethodCode	No
PaymentSetIdentifier	No
PaymentTransactionIdentifier	No

Attribute	Updatable
PaymentType	No
PaymentTypeCode	No
SourceTransactionPaymentIdentifier	No

Manual Price Adjustment Attributes

Attribute	Updatable
AdjustmentAmount	No
AdjustmentElementBasis	No
AdjustmentElementBasisCode	No
AdjustmentType	No
AdjustmentTypeCode	No
ChargeDefinition	No
ChargeDefinitionCode	No
ChargeRollupFlag	No
Comments	No
ManualPriceAdjustmentId	No
ParentEntityCode	No
ParentEntityId	No
PricePeriodicityCode	No
Reason	No
ReasonCode	No

Attribute	Updatable
SequenceNumber	No
SourceManualPriceAdjustmentIdentifier	No
ValidationStatusCode	No

Project Attributes

Attribute	Updatable
ContractId	Yes
ContractNumber	Yes
DooOrderPrjId	No
ExpenditureItemDate	Yes
ExpenditureOrganization	Yes
ExpenditureType	Yes
ExpenditureTypeId	Yes
FundingSource	Yes
OrganizationId	Yes
ParentEntityCode	No
ParentEntityId	No
ProjectId	Yes
ProjectNumber	Yes
ReferenceProjectId	No
ReservedAttribute1	Yes
TaskId	Yes

Attribute	Updatable
TaskNumber	Yes

Lot Serial Attributes

Attribute	Can Update
FulfillLineId	No
ItemRevisionNumber	Yes
Locator	Yes
LocatorIdentifier	Yes
LotNumber	Yes
LotSerialId	No
Quantity	No
SecondaryQuantity	No
SerialNumberFrom	Yes
SerialNumberTo	Yes
SourceTransactionLotIdentifier	Yes

Transactional Attributes

Attribute	Can Update
CharacterValue	Yes
DateValue	Yes
FulfillLineId	No
HeaderId	No
LineId	No
NumberValue	Yes
SourceTransactionLineAttributeIdentifier	Yes
TimestampValue	Yes
TransactionAttrId	No
TransactionAttributeCode	Yes
TransactionAttributeIdentifier	Yes

Attribute	Can Update
TransactionAttributeName	Yes

Note

- TransactionAttrId is a surrogate key for the transaction item.

Payment Attributes

Attribute	Can Update
Amount	No
AuthorizationStatus	No
AuthorizedInSourceFlag	No
FulfillLineId	No
HeaderId	No
InstrumentAssignmentIdentifier	No
PaymentIdentifier	No
PaymentMethod	No
PaymentMethodCode	No
PaymentSetIdentifier	No
PaymentTransactionIdentifier	No
PaymentType	No
PaymentTypeCode	No
SourceTransactionPaymentIdentifier	No

Document Reference Attributes

Attribute	Can Update
DocumentAdditionalLineIdentifier	Yes
DocumentAdditionalLineNumber	Yes
DocumentAdditionalNumber	Yes
DocumentAdditionalSubLineIdentifier	Yes
DocumentAdditionalSubLineNumber	Yes
DocumentAdditionalIdentifier	Yes
DocumentIdentifier	Yes

Attribute	Can Update
DocumentLineIdentifier	Yes
DocumentLineNumber	Yes
DocumentNumber	Yes
DocumentReferenceType	Yes
DocumentSubLineIdentifier	Yes
DocumentSubLineNumber	Yes
DocumentSystemReferenceIdentifier	No
FulfillLineIdentifier	No
HeaderId	No
LineIdentifier	No
OwnerTableId	Yes
OwnerTableName	Yes
TaskType	Yes

Manual Price Adjustment Attributes

Attribute	Can Update
AdjustmentAmount	No
AdjustmentElementBasis	No
AdjustmentElementBasisCode	No
AdjustmentType	No
AdjustmentTypeCode	No
ChargeDefinition	No
ChargeDefinitionCode	No
ChargeRollupFlag	No
Comments	No
EffectivityType	No
EffectivityTypeCode	No
ManualPriceAdjustmentId	No
NumberOfPeriods	No
ParentEntityCode	No

Attribute	Can Update
ParentEntityId	No
PeriodFrom	No
PeriodUntil	No
PricePeriodicityCode	No
Reason	No
ReasonCode	No
SequenceNumber	No
SourceManualPriceAdjustmentIdentifier	No
ValidationStatusCode	No

Fulfillment Line Detail Attributes

Attribute	Can Update
CustomerTrxLineIdentifier	No
DeliveryName	No
ExceptionFlag	No
FulfillLineDetailId	No
FulfillLineId	No
Quantity	No
RMAReceiptDate	No
RMAReceiptLineNumber	No
RMAReceiptNumber	No
RMAReceiptTransactionIdentifier	No
SecondaryQuantity	No
Status	No
StatusAsOfDate	No
TaskType	No
TrackingNumber	No
TradeComplianceCode	No
TradeComplianceExplanation	No
TradeComplianceName	No

Attribute	Can Update
TradeComplianceResultCode	No
TradeComplianceResultName	No
TradeComplianceTypeCode	No
TradeComplianceTypeName	No
WaybillNumber	No

Project Attributes

Attribute	Can Update
ContractId	Yes
ContractNumber	Yes
DooOrderPrjId	No
ExpenditureItemDate	Yes
ExpenditureOrganization	Yes
ExpenditureType	Yes
ExpenditureTypeId	Yes
FundingSource	Yes
OrganizationId	Yes
ParentEntityCode	No
ParentEntityId	No
ProjectId	Yes
ProjectNumber	Yes
ReferenceProjectId	No
ReservedAttribute1	Yes
TaskId	Yes
TaskNumber	Yes

Term Attributes

Attribute	Can Update
ModifiedFlag	No
OrderTermIdentifier	No

Attribute	Can Update
ParentEntityCode	No
ParentEntityIdentifier	No
ReferenceOrderTermIdentifier	No
SourceOrderTermIdentifier	No
TermAdjustmentPercent	No
TermApplicationMethod	No
TermApplicationMethodCode	No
TermDuration	No
TermEndDate	No
TermPeriod	No
TermPeriodCode	No
TermStartDate	No
TermSubtypeCode	No
TermTypeCode	No

Charge Tier Attributes

Attribute	Can Update
AdjustmentAmount	No
AdjustmentBasisIdentifier	No
AdjustmentType	No
AdjustmentTypeCode	No
ApplicationMethod	No
ApplicationMethodCode	No
BlockSize	No
DeltaType	No
ModifiedFlag	No
OrderChargeIdentifier	No
OrderChargeTierIdentifier	No
PartialBlockAction	No
ReferenceOrderChargeTierIdentifier	No

Attribute	Can Update
SourceOrderChargeTierIdentifier	No
TierFrom	No
TierSequenceNumber	No
TierTo	No

Asset Attributes

Attribute	Can Update
AssetIdentifier	Yes
AssetTransactionDetailId	No
AssetUomCode	Yes
CustomerAssetEndDate	Yes
FulfillLineIdentifier	No
ModifiedFlag	No
Quantity	No
ReferenceAssetTransactionDtlId	No
SourceAssetTransactionDetailId	No
StatusIdentifier	Yes
TransactionActionCode	Yes

Channel Program Attributes

Attribute	Can Update
ProgramDetails	No
ProgramId	Yes
ProgramLine	No
ProgramLineId	Yes
ProgramName	No
ProgramTypeName	No
ReferenceIncentiveIdentifier	No
SourceTransactionIncentiveIdentifier	Yes
StartDate	No

Attribute	Can Update
Status	Yes
StatusCode	Yes

Hold Attributes

Attribute	Can Update
ActiveFlag	No
ApplyDate	No
ApplySystem	No
ApplyUserId	No
CreatedBy	No
CreationDate	No
DeletedFlag	No
DooHeaderId	No
DooLineId	No
FulfillLineId	No
HoldCode	No
HoldCodeId	No
HoldComments	Yes
HoldInstanceId	No
HoldReleaseComments	Yes
HoldReleaseReasonCode	Yes
HoldRunningTaskFlag	No
LastUpdateDate	No
LastUpdatedBy	No
ObjectVersionNumber	Yes
PendingFlag	No
ReleaseDate	No
ReleaseUserId	No
SourceRequestId	No
TransactionEntityId1	No

Attribute	Can Update
TransactionEntityId2	No
TransactionEntityName1	No
TransactionEntityName2	No

Flexfields

Overview

Overview of Using Extensible Flexfields in Order Management

Set up an extensible flexfield so you can add your own attribute to Order Management.

An extensible flexfield is a field you can use to capture details in a sales order that are unique to your business requirements. Each sales order in Order Management comes predefined with a lot of attributes, but you might need one that's specific to your needs.

You can use the details in an extensible flexfield to determine process automation, send and receive details when communicating with systems outside of Order Management, or provide the criteria to use in a complex business rule. For example:

- Capture consumer details in an extensible flexfield, then use them to add free samples to a shipment.
- Capture customer loyalty details in an extensible flexfield, then determine whether to call the customer or upgrade shipping priority.
- Receive a sales order request that includes unique build specifications from your customer, store them in an extensible flexfield, and then route the sales order to different manufacturing facilities according to the specifications.
- Select and ship a bottle of wine according to your customer dining preferences. Store these preferences in an extensible flexfield.
- Store the user ID of the order entry clerk who submitted a source order from a source system.
- Track source orders that include warning messages from a source system.
- Store the original schedule date and the new schedule date on the order line so you can track and report scheduling throughout the fulfillment lifecycle.

Note

- An extensible flexfield supports a one-to-many relationship between one entity and one or more attributes. You can use it to add more than one context sensitive segment.
- You can set up an extensible flexfield for a fulfillment line, or on some other object that supports an extensible flexfield.
- If you add an extensible flexfield to a fulfillment line, and if you use the *Update or Close Sales Orders* scheduled process to close the order line that the fulfillment line references at runtime, then the Order Entry Specialist.
    - Can't update the value in the extensible flexfield on the fulfillment line.

- Can revise the sales order and update the value in the extensible flexfield on the fulfillment line, but Order Management won't use the revised value when it processes the revision.
- Can update an extensible flexfield on the order header regardless of the status of the sales order or order line.

For details, see *Overview of Setting Up Extensible Flexfields in Order Management*.

Examples of Using Extensible Flexfields

Example	Description
Get details from a source system.	Get source order details from one or more source systems. A source order is an order that you import into Order Management from a source system, such as from an upstream channel. A source order contains a set of attributes. If you need details or attributes that the source order doesn't contain, then you can use an extensible flexfield to get them, then use these details during order fulfillment. You can use the same extensible flexfield attributes to receive details from each source system, or use different extensible flexfield attributes according to the unique requirements of each source system.
Send details to a fulfillment system.	Order Management sends a fulfillment request that includes a predefined set of attributes to a fulfillment system. You can use an extensible flexfield to send details that these attributes don't include, but that the fulfillment system needs to finish the fulfillment request.
Integrate with systems outside of Order Management.	For details, see *Overview of Using Flexfields to Integrate Order Management with Other Oracle Applications*.
Receive details from a fulfillment system.	A fulfillment system might send attributes that provide a business value, and that the Order Entry Specialist must view in Order Management or in the source system. In some situations, Order Management might also use these details in the next set of tasks that it runs in an orchestration process. You can use an extensible flexfield to receive these attributes.
Write a business rule.	Use an extensible flexfield attribute with a business rule. • Transformation rule. Write a rule that references an extensible flexfield to add order details, delete unnecessary details, or modify details. Use an extensible flexfield to determine the transformations to do. • Routing rule. • Assignment rule. For details, see *Overview of Using Business Rules With Order Management*.
Manage change.	Use an extensible flexfield as an order attribute to store details about a change that occurred. For example, use an extensible flexfield as input to calculate the cost of change and to select a compensation pattern when Order management receives a change order.
Display attributes in the Order Management work area.	Allow the Order Entry Specialist to search for sales orders in the Order Management work area according to the value of an extensible flexfield. Note that the Order Entry Specialist can read but not edit an extensible flexfield attribute.

Examples of Using Extensible Flexfields with Orchestration Processes

Example	Description
Assign an orchestration process.	Use the value of an extensible flexfield in an assignment rule to automatically assign an orchestration process during order fulfillment to fulfill each sales order and each order line. Use an extensible flexfield as part of the selection criteria during assignment.
Set up an orchestration process.	Create a business rule that uses an extensible flexfield to determine branching and the tasks to run in an orchestration process.
Select the fulfillment lines to process.	Use an extensible flexfield to determine whether a task in an orchestration process will process the fulfillment line or ignore it during order fulfillment.
Calculate the lead time that an orchestration process step requires.	Use an extensible flexfield to calculate lead time according to your data. For example, create an extensible flexfield named Engraving that includes a Yes or No value, then set up conditions. • If the item is a silver cup, and if engraving is yes, then set lead time to 3 days. • If the item is a silver cup, and if engraving is no, then set lead time to 1 day. For details, see *Set Up Jeopardy and Lead Time to Manage Delay*.

Entities That You Can Use With Extensible Flexfields

You can use with an extensible flexfield only with these entities:

- Header
- Fulfillment line
- Fulfillment line detail

Web Services You Can Use to Map Extensible Flexfields

You can use these web services with all task types.

Web Service	Description
Receive Fulfill Order Orchestration Task Response Service	For data that flows into Order Management. It uses this composite. • DooTaskFulfillOrderResponseInterfaceComposite
Request Fulfill Order Orchestration Task Service	For data that flows out of Order Management. It uses this composite. • DooTaskFulfillOrderRequestInterfaceComposite

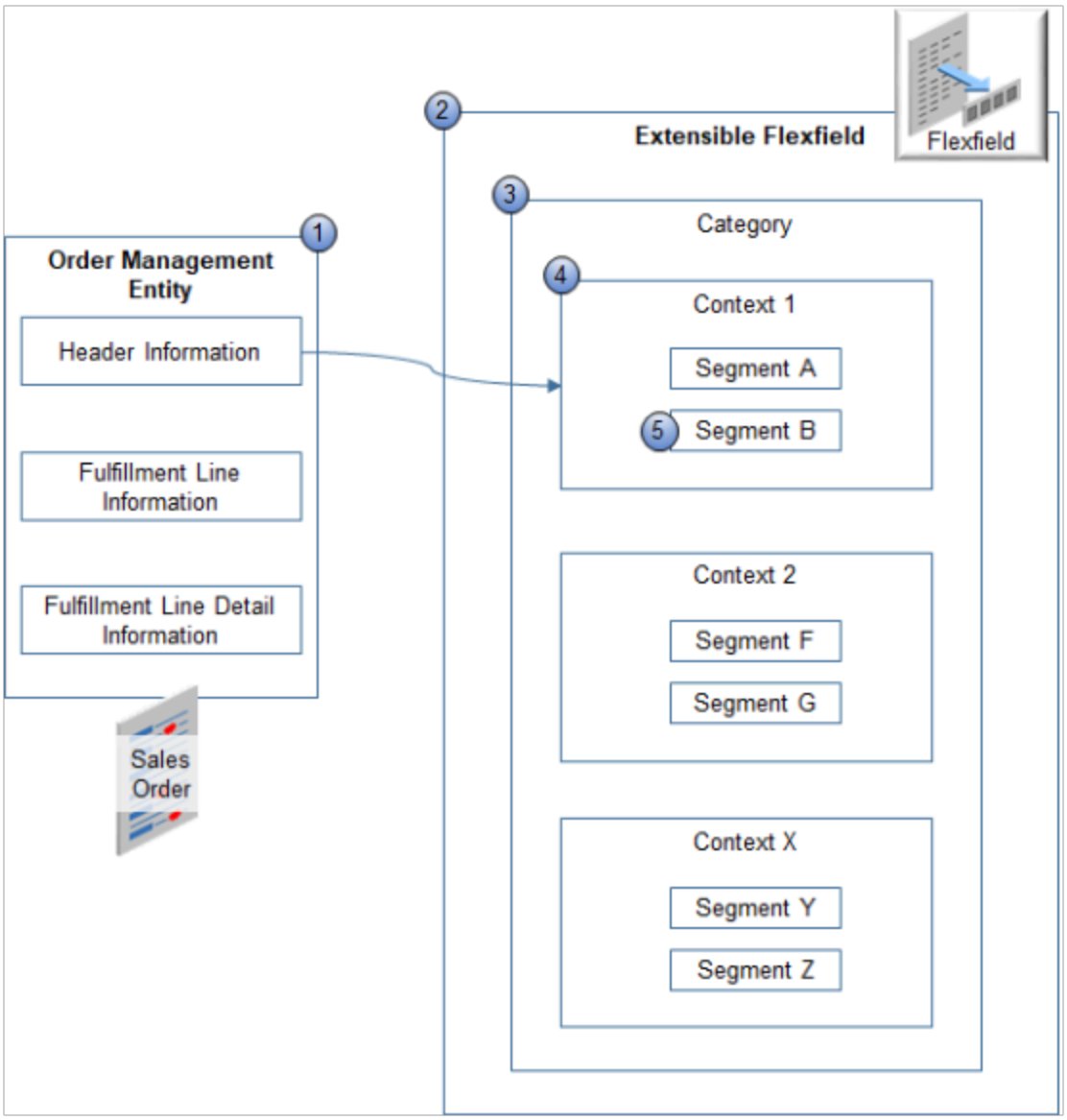
Related Topics

- Set Up Extensible Flexfields in Order Management
- Overview of Setting Up Extensible Flexfields in Order Management
- More Set Up Details for Extensible Flexfields
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Flexfields

Overview of Setting Up Extensible Flexfields in Order Management

Use a hierarchy to determine how to display an extensible flexfield in the Order Management work area.

Here's the hierarchy.

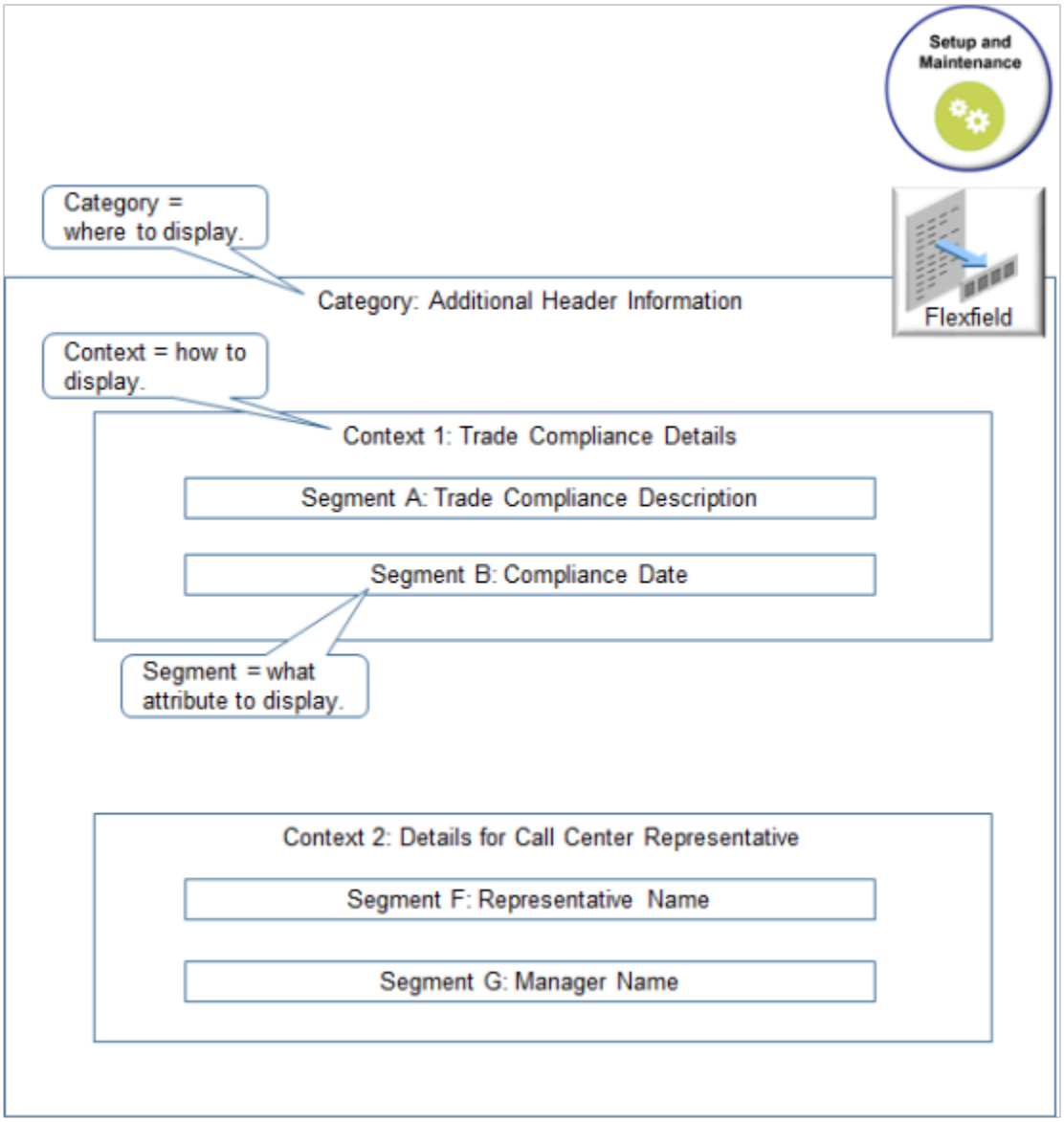


Note

1. **Order Management Entity**. The top-level object in the hierarchy. You can specify an entity.
    - **Header Information**. Store details about the order header.
    - **Fulfillment Line Information**. Store details about the order line while order fulfillment processes the order line.
    - **Fulfillment Line Detail Information**. Store details that the fulfillment system provides when you don't typically store these details on the fulfillment line.
2. **Extensible Flexfield**. Each extensible flexfield includes categories, contexts, and segments.
3. **Extensible Flexfield Category**. Organize your data into a category according to attributes that contain similar business data. Specify where to display the extensible flexfield in the Order Management work area, such as on the order header. You can use only one category for each entity.
4. **Extensible Flexfield Context**. Specify how to display the attribute. For example:
    - Specify criteria that determines the attribute to display.
    - Type of interface element that will display the attribute, such as text box or list of values.
    - Whether the Order Entry Specialist can view or edit the attribute.
    - Validation to do on the attribute.
    - Help text to display for the attribute.
    - Data type of the attribute.
    - Table column that stores the attribute.

    You can specify one or more contexts in each category.
5. **Extensible Flexfield Segment**. Specify the attributes. An extensible flexfield segment is equivalent to a sales order attribute. Each context includes one or more segments that you can use to specify sales order attributes that store your data. You can specify one or more segments in each context.

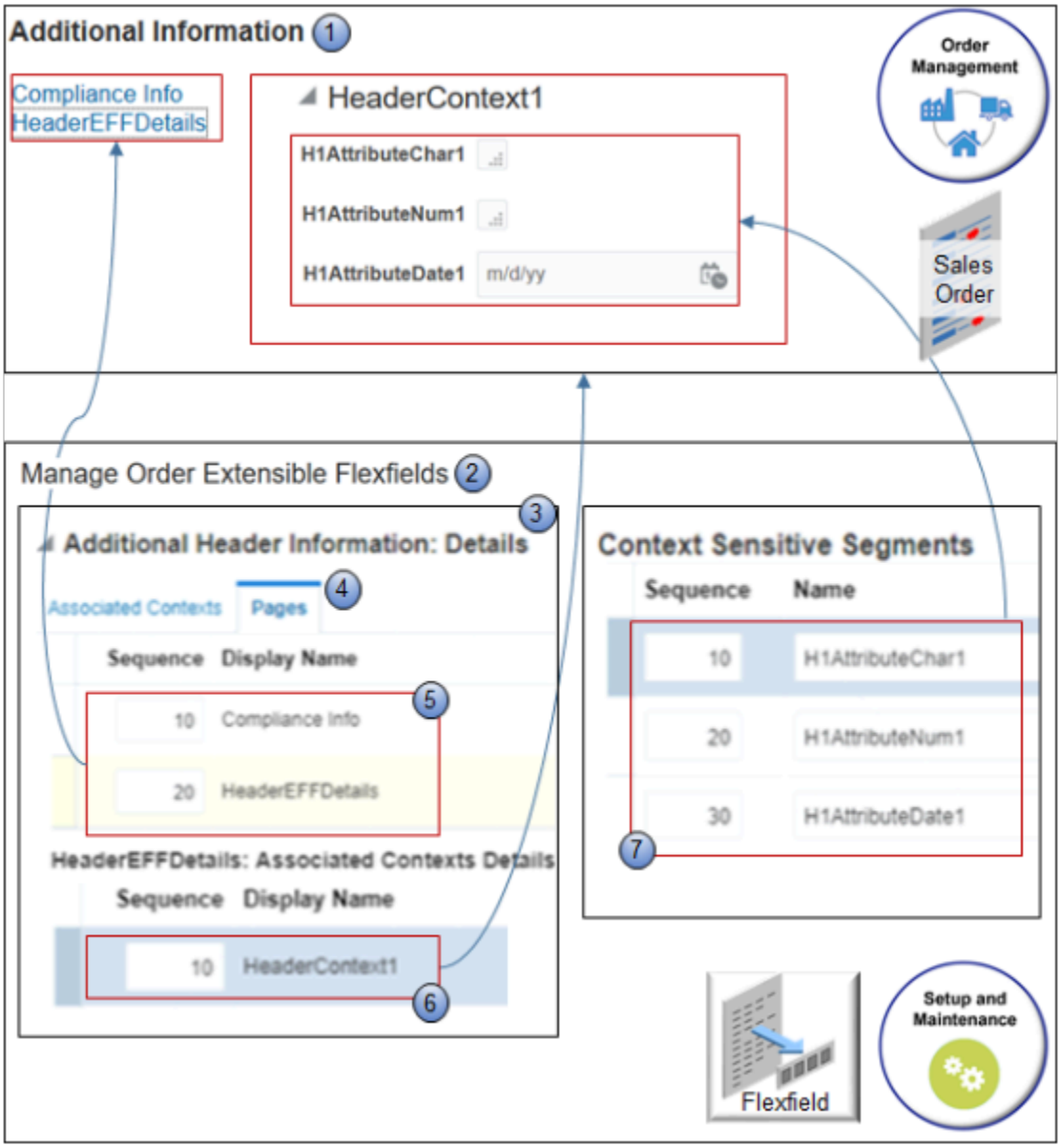
Consider an example.



Note

- Context1 stores details about trade compliance.
- SegmentA references an attribute that stores the compliance description.
- SegmentB references an attribute that stores the compliance date.
- Context2 stores details about the call center representative who manages details about trade compliance.
- SegmentF references an attribute that stores the representative name.
- SegmentG references an attribute that stores the name of the person who manages the representative.

Here's an example that displays an extensible flexfield in the Order Management work area.



Note

1. **Additional Information**. A dialog that displays when you click **Actions > Edit Additional Information** on the sales order header in the Order Management work area.
2. **Manage Order Extensible Flexfields**. The setup page that you use in the Setup and Maintenance work area to set up an extensible flexfield.
3. **Additional Header Information**. Specifies the data to display and how to display it in Additional Information.
4. **Pages tab**. You use the Pages tab to specify each area where you want to display your extensible flexfield. In this example, the Pages tab specifies one area named `Compliance Info` and another area named HeaderEFFDetails. The example displays HeaderEFFDetails in the dialog. You can specify one or more areas, and you can specify one more contexts in each area.

   In this example, the Pages tab specifies `HeaderContext1` and `HeaderContext2`. However, for brevity, the screen capture displays only `HeaderContext1`. The Sequence specifies the sequence to use when displaying these contexts in the area, with 1 displaying at the top, 2 below 1, 3 below 2, and so on.

5. **Sequence and Display Name for page**. Specifies the name for each category that displays in the dialog. Sequence specifies the sequence to display the pages in the dialog. In the example, the Sequence is 10 for Compliance Info and 20 for HeaderEFFDetails, so the dialog displays Compliance Info first, then HeaderEFFDetails.
6. **Sequence and Display Name for context**. Specifies the name for each context that displays in the dialog. For example, `HeaderContext1` organizes similar business data into a concise area of the dialog.
7. **Context Sensitive Segment**. Specifies one ore more segments that display details for each attribute. The Sequence specifies the sequence to use when displaying segments in the area. You can specify these segments in each context:
    - 20 character segments. You can use up to 150 characters for each character segment.
    - 10 number segments.
    - Five date segments.
    - Five date and time segments.

Flexfield Categories Determine Where to Display Extensible Flexfields

You can display an extensible flexfield in different sections of the Order Management work area according to the flexfield category.

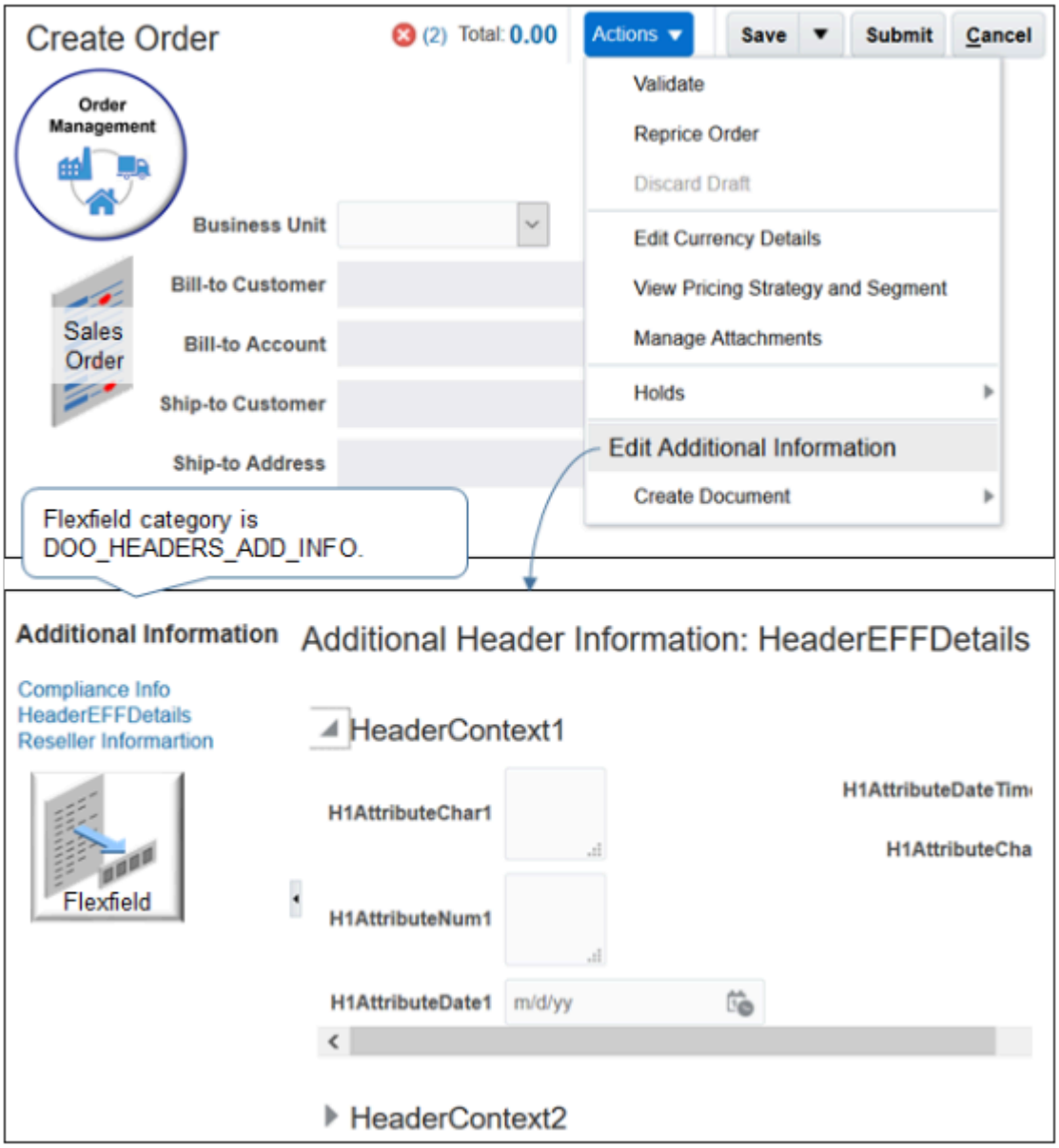
Flexfield Category	Section in the Order Management Work Area
DOO_HEADERS_ADD_INFO	Order header
DOO_FULFILL_LINES_ADD_INFO	Fulfillment line
DOO_FULFILL_LINE_DTLS_ADD_INFO	General tab on the fulfillment line

View Extensible Flexfields in a Sales Order According to Flexfield Category

1. Go to the Order Management work area, then click **Tasks > Create Order**.

2. On the Create Order page, click **Actions > Edit Additional Information**.

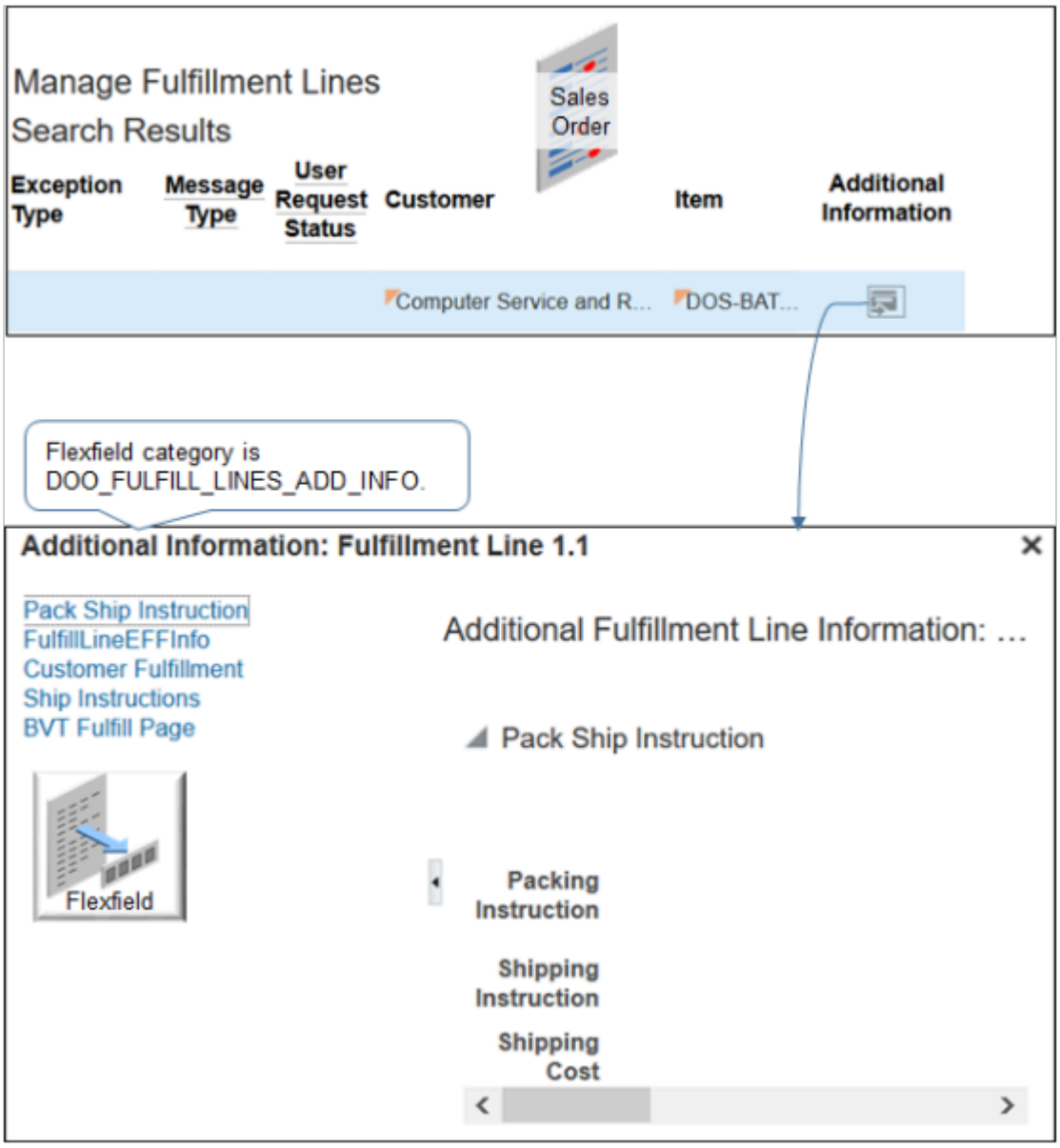
   The Additional Information dialog displays each extensible flexfield you create that references the DOO_HEADERS_ADD_INFO flexfield category. For example, HeaderEFFDetails is an extensible flexfield.



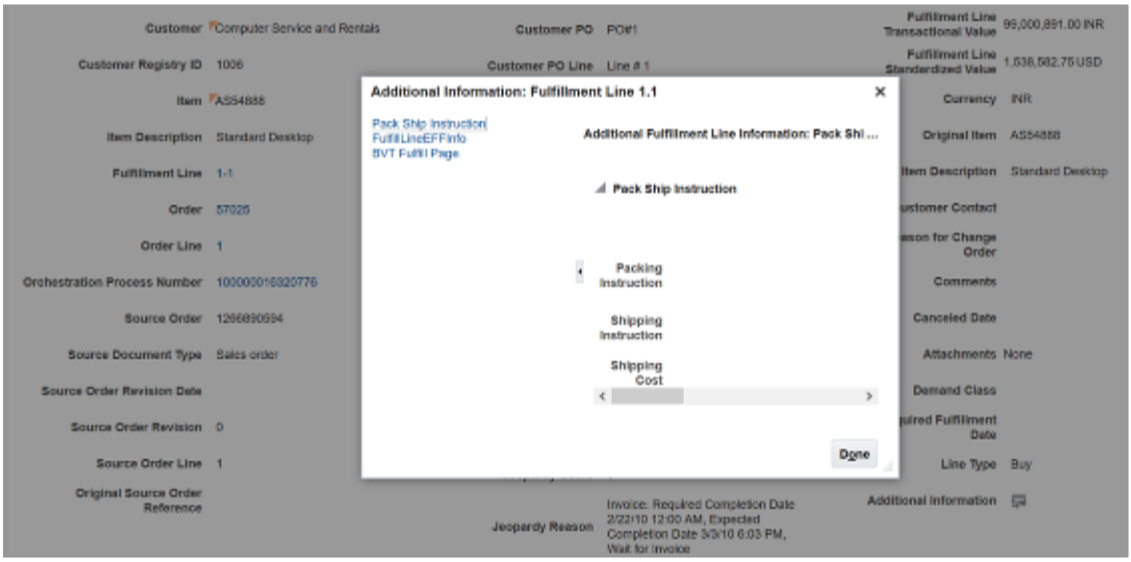
3. Click **Cancel**.
4. In the Order Lines area, search for an item, then click **Add**.
5. On the order line, click the **down arrow**, then click **Edit Additional Information**.
6. Click **Submit**, then copy the sales order number, such as 57025.
7. On the Overview page, click **Tasks > Manage Fulfillment Lines**.
8. On the Manage Fulfillment Lines page, search for your sales order.

9. In the Search Results, scroll to the far right, then click **Additional Information**.

   The Additional Information dialog displays the extensible flexfield you create that references the DOO_FULFILL_LINE_DTLS_ADD_INFO flexfield category. For example, Pack Ship Instruction is an extensible flexfield.

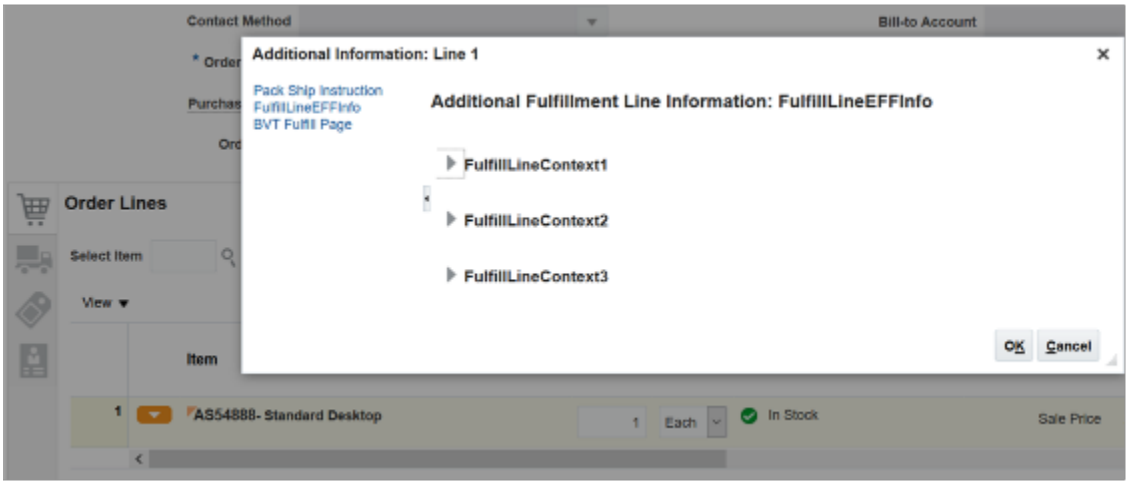


10. In the Fulfillment Line Details area, in the Attributes area, click **General**, scroll down, click **Additional Information**, then notice the dialog that displays.

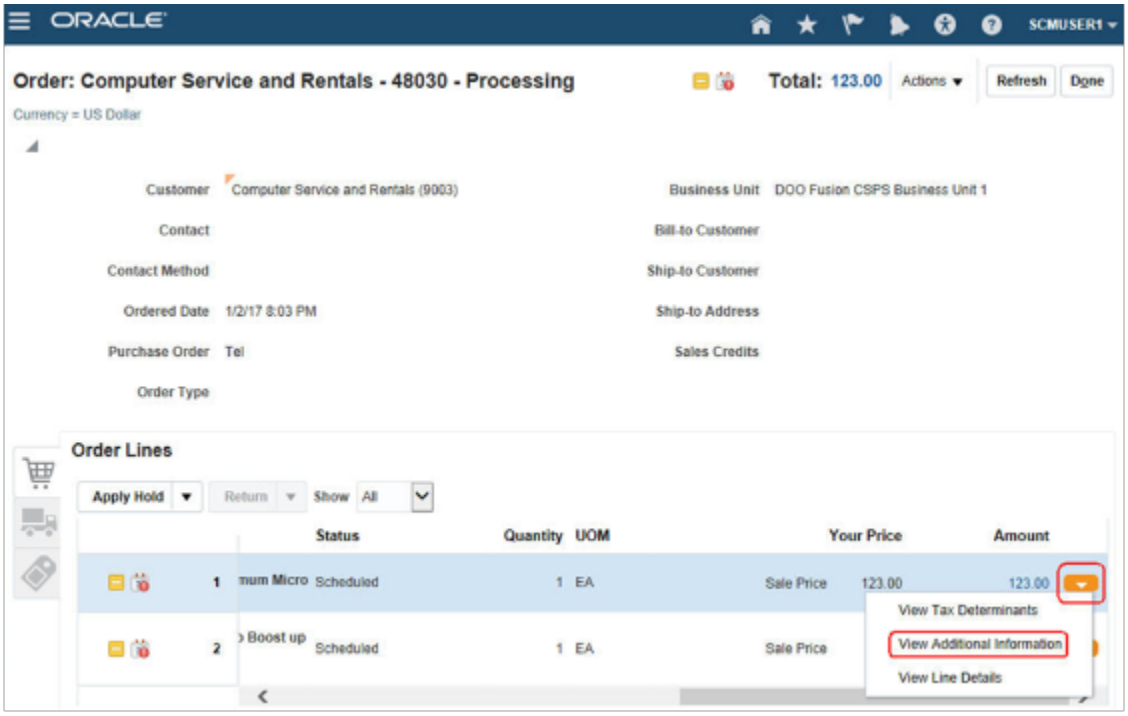


Another Example

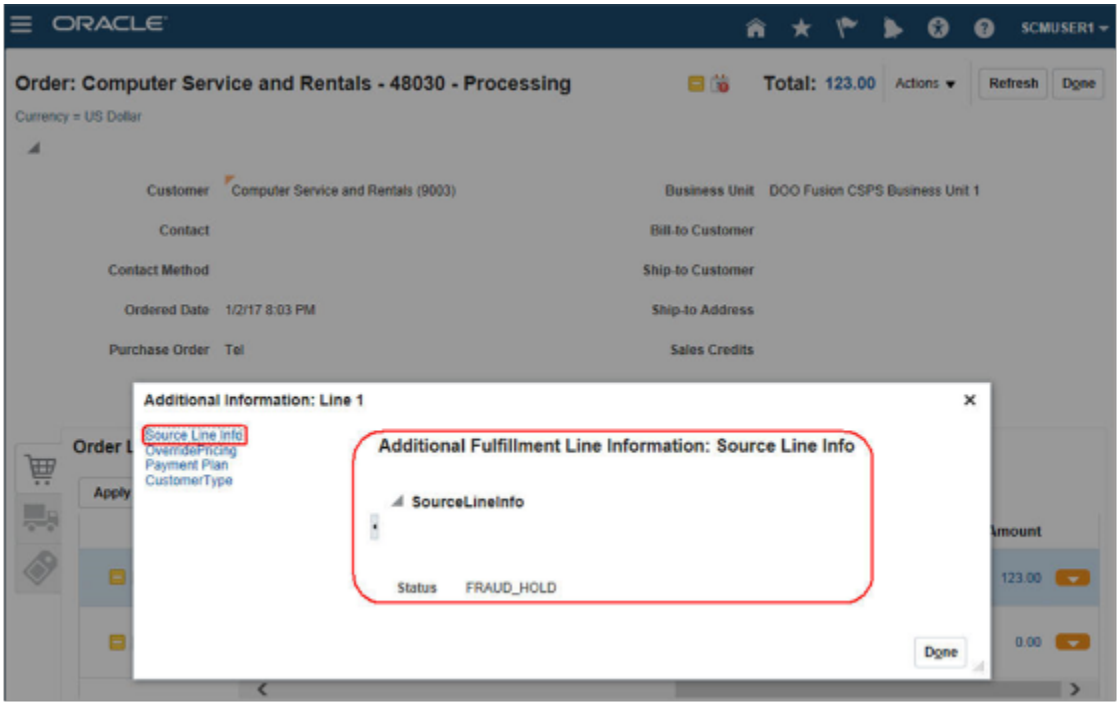
Here's an Additional Information dialog that displays each extensible flexfield that references the DOO_FULFILL_LINES_ADD_INFO flexfield category. For example, FulfillLineEFFInfo is an extensible flexfield.



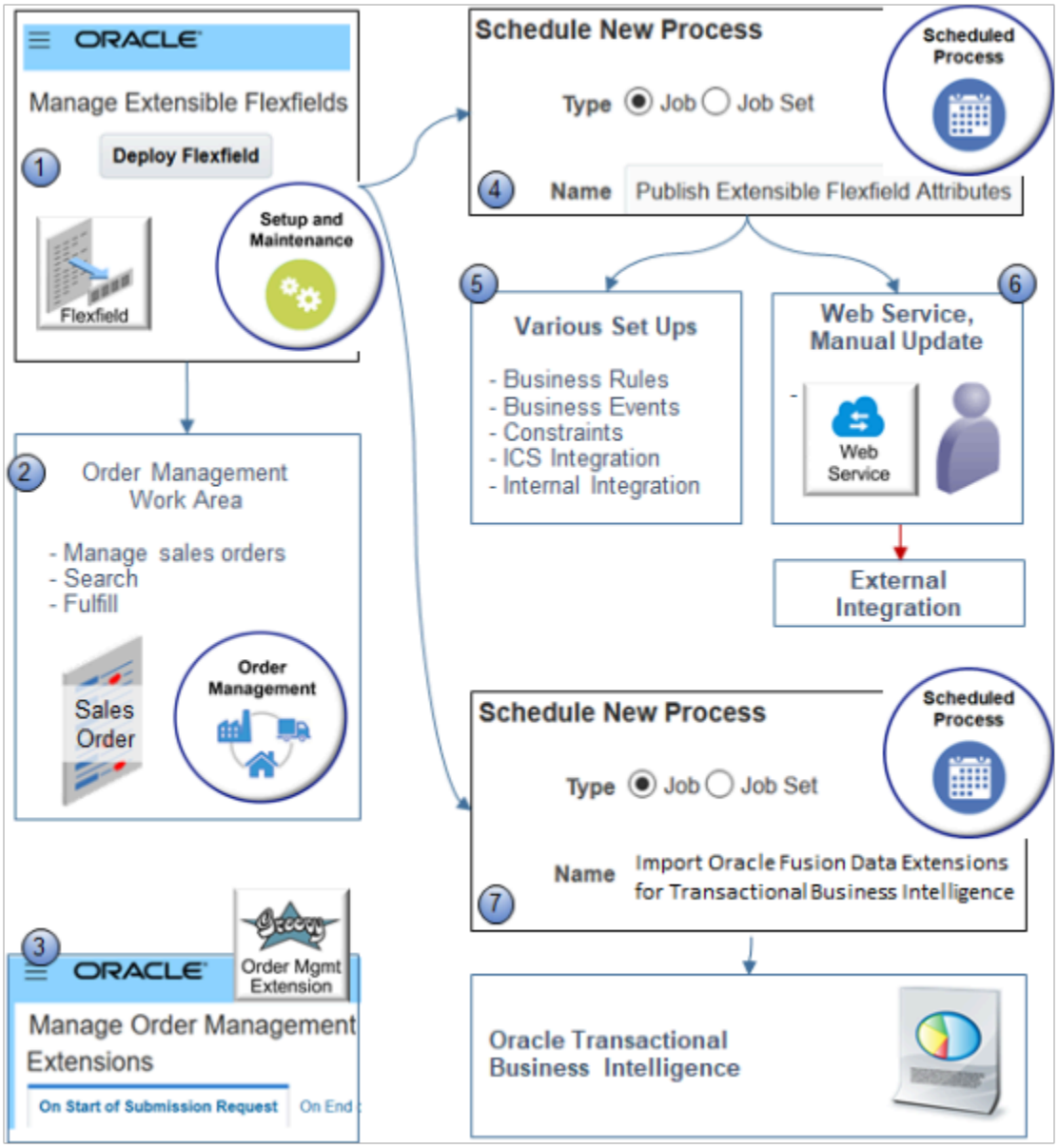
Order Management displays each extensible flexfield in an Additional Information dialog. Here's an example that displays the View Additional Information menu item on each order line on the Order Lines tab.



In this example, View Additional Information allows the Order Entry Specialist to access the `Source Line Info` extensible flexfield.



Overview of Your Setup



You do some or all of this set up, depending on your business requirements. It might be necessary to also do some of this flow when you update your extensible flexfield definition.

1. **Manage Extensible Flexfields**. Use the Manage Extensible Flexfields page in the Setup and Maintenance work area to define your extensible flexfield. Click Deploy Flexfield to make it available throughout the application and to other set up pages.
2. **Order Management Work Area**. Test your extensible flexfield. Depending on your setup, you create a new sales order, search for a sales order, use a fulfillment view, and so on. For example, use the Additional Information dialog when you create a new sales order to make sure your extensible flexfield displays and works as expected.
3. **Manage Order Management Extensions**. Use the Manage Order Management Extensions page in the Setup and Maintenance work area to add your extensible flexfield to an order management extension.
4. **Schedule New Process**. Use the *Publish Extensible Flexfield Attributes* scheduled process to make your extensible flexfield available to various parts of Order Management, other set ups in Oracle Applications, and

integrations. This scheduled process updates the rule dictionaries in business rules, constraints, business events, Integration Cloud Service, and the service mappings that you use for an integration.

5. **Various Set Ups**. Here's where you can use your extensible flexfield.

Set Up	Description
Business Rule	Use your extensible flexfield in a business rule. o Pretransformation o Transformation o Posttransformation o Routing o Assignment o Line Selection Criteria o Branching Expression o Lead Time Expression o Compensation Pattern o Cost of Change You can't use an extensible flexfield in an approval rule.
Business Event	Use an extensible flexfield to help determine when to use a business event, and the actions to take when you use it.
Constraint	Use an extensible flexfield to constraint a sales order attribute. Use a constraint to constrain an extensible flexfield.
Integration Cloud Service	Use an extensible flexfield with Integration Cloud Service to help achieve and manage an integration between Order Management and a technology or application. o Order Import o Order Information Service o Transportation Management o Global Trade Management
Integration with an Oracle Application	Use a service mapping to integrate with an Oracle application. o Shipping o Accounts Receivable o Purchasing o Receiving

6. **Web Service**. If you use one of these web services, then manually update your extensible flexfield definition in the web service payload.
    - CreateOrder
    - FulfillOrderService
    - FulfillmentResponseService
7. **Schedule New Process**. Use the Import Oracle Fusion Data Extensions for Transactional Business Intelligence scheduled process to make sure your extensible flexfield is available in the Subject Areas of Oracle Transactional Business Intelligence area. For details, see *Overview of Flexfield Change Import*.

Related Topics

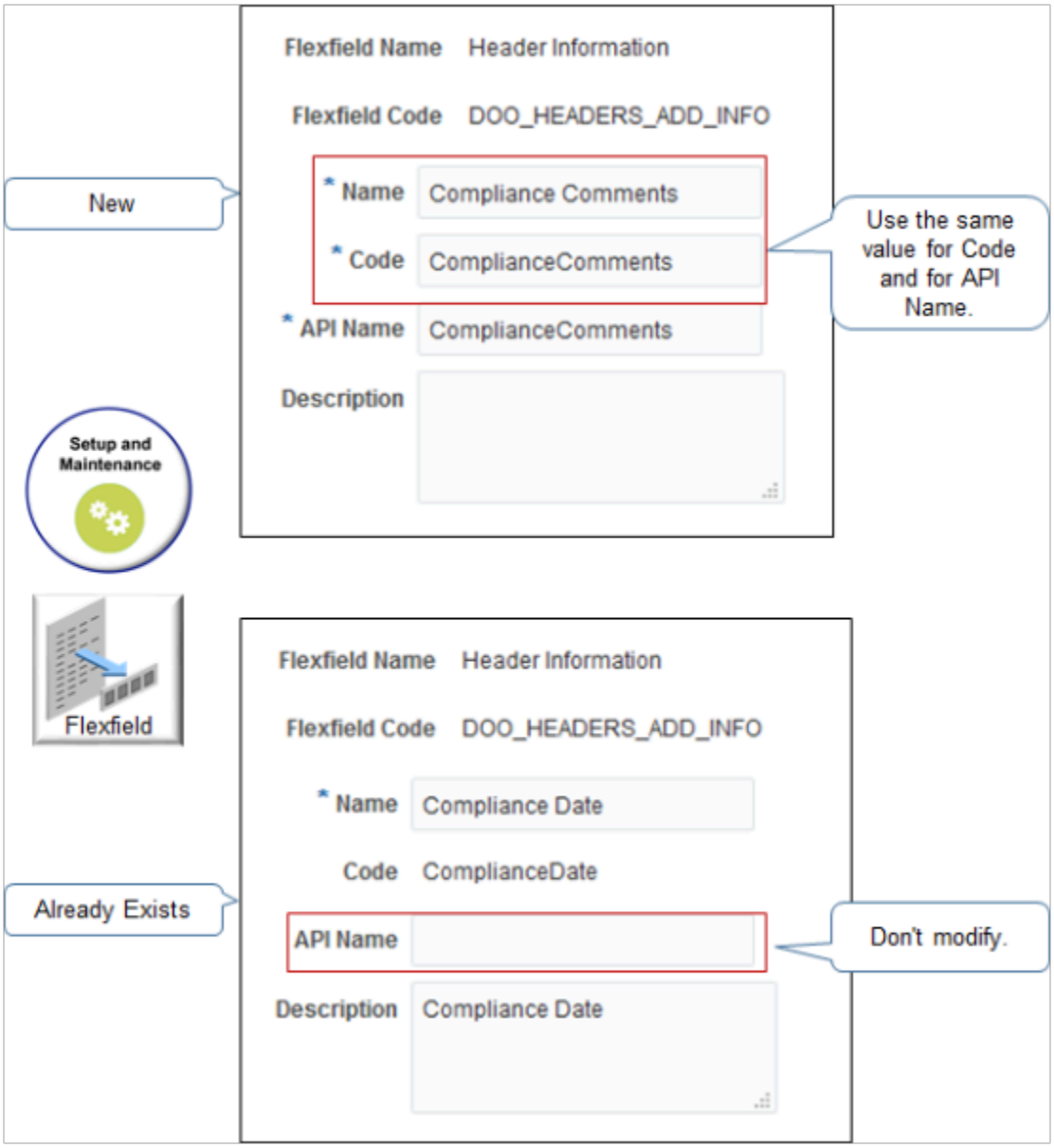
- Set Up Extensible Flexfields in Order Management
- More Set Up Details for Extensible Flexfields
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Importing Orders Into Order Management

Guidelines for Setting Up Extensible Flexfields in Order Management

Apply guidelines when you set up an extensible flexfield in Order Management.

- Unlike a descriptive flexfield, you can add as many segments as you need.
- Use the context to organize your business data into a concise area of the user interface element, such as a dialog.
- Organize your attributes into logical groups according to your display and usage requirements.
- Consider where an extensible flexfield might be useful in the order capture and order fulfillment lifecycle.
- An extensible flexfield that you add on the sales order header is independent of an extensible flexfield that you add on a fulfillment line. Order Management doesn't communicate or transfer values between them, even if you use the same name for each extensible flexfield. If you require dependence between them, then create an order management extension, then define dependence logic in the extension.

Create Segments and Contexts



If you are:

- **Creating a new segment.** Use the same value for the Code attribute and the API Name attribute. This will simplify setup and maintenance. You use API Name throughout the setup, such as during integration setup, with web services, business rules, business intelligence, and so on. You can't modify Code or API Name after you create the segment.
- **Modifying a segments that already exists.** Don't modify API Name. Extensible flexfield logic examines API Name. If its empty, then the logic uses the value in Code, so it isn't necessary to specify a value in API Name. If you do modify API Name, then your flexfield might fail with unexpected results at run time.

Note

- Enter a unique code, name, and description for each segment. The Order Management work area doesn't display these values, so use whatever nomenclature is most helpful to you during setup and maintenance.
- You must not enable the Translatable option on the context. For details, see *Set Up Extensible Flexfields in Order Management*.

Nomenclature

Make sure the Name attribute and the Code attribute for each segment or context that you create uses the correct nomenclature:

- Begin the name with a letter of the alphabet, A through Z.
- Don't include more than one consecutive space character.
- Don't include spaces in the Code attribute. Spaces will cause a run time error.
- Make sure the name contains only letters and numbers, and only these special characters:

Character	Description
.	Period
-	Hyphen Don't use an em dash or en dash.
_	Underscore
:	Colon
'	Single quotation mark
/	Forward slash

If you encounter an incorrect alias error when you run the *Publish Extensible Flexfield Attributes* scheduled process, then make sure your contexts and segments use the correct nomenclature.

Number of Attributes and Segments

The number of attributes and segments that you can add to the context depends on the data type that you set for the context.

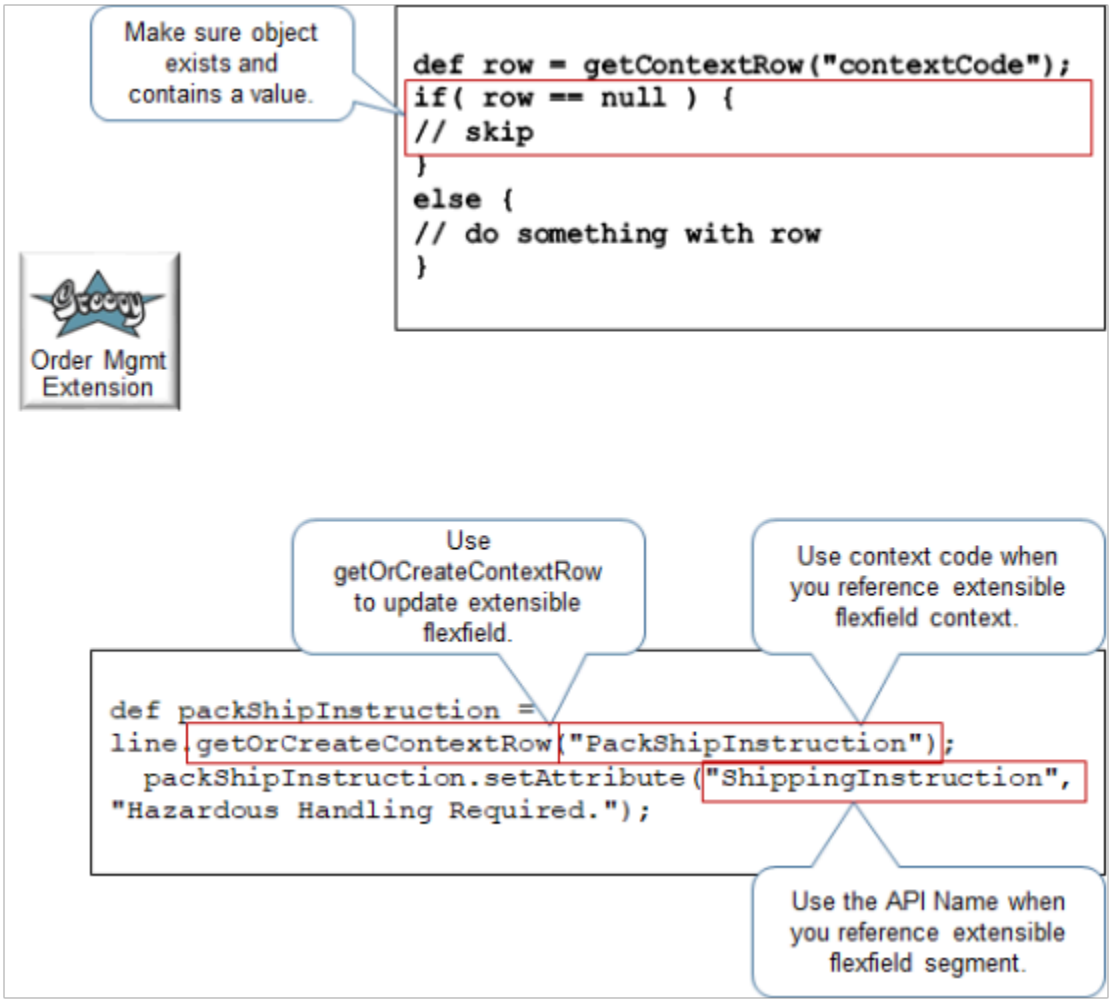
Data Type	Number of Attributes That You Can Add	Number of Segments That You Can Add
Varchar	20	20

Data Type	Number of Attributes That You Can Add	Number of Segments That You Can Add
Number	10	5
Date	5	10
Time	5	10

You can add a maximum of 40 segments to each context.

Order Management doesn't limit how many contexts that you can add, but you must add only the minimum number of contexts that you need because each context can affect performance.

Create Order Management Extensions



Note

- Include an If statement that makes sure each context and segment that your extension references exists and contains a value. If you don't check for missing objects and empty values, then your extension might fail with unpredictable results.

- Use the getOrCreateContextRow method to create or update a value in an extensible flexfield. If the object doesn't exist, then this method creates it, so it isn't necessary to determine whether the object exists before you use this method to do an update. Use getContextRow to only get a row.
- Use Context Code when you reference an extensible flexfield context.
- Use API Name when you reference an extensible flexfield segment. If you don't specify API Name when you set up the extensible flexfield, then use Segment Code but with underscores (_) instead of spaces ( ).
- Your Groovy code must traverse the context and the segment to get all data from the extensible flexfield. Here's an example that uses a `while` statement to traverse all data.

```
def complianceDetails = header.getOrCreateContextRow("ComplianceDetails");
complianceDetails.setAttribute("_ComplianceReason", "This has been set by pre submit extension.");

def lines = header.getAttribute("Lines");
while( lines.hasNext() ) {
 def line = lines.next();
 def inventoryItemId = line.getAttribute("ProductIdentifier");
 def orgId = line.getAttribute("InventoryOrganizationIdentifier");
 def item = getItem(inventoryItemId, orgId);

 String hazardous = item.getAttribute("HazardousMaterialFlag");
 def packShipInstruction = line.getOrCreateContextRow("PackShipInstruction");

 if( "Y".equals(hazardous) ) {
 // get details for fulfill line context PackShipInstruction

 packShipInstruction.setAttribute("_ShippingInstruction", "Hazardous Handling Required.");
 }

}
```

Use Business Rules

If your fulfillment system needs data that the predefined attributes in a sales order don't provide, then use a value from an extensible flexfield as the criteria for setting the default value in some other sales order attribute.

Assume you set up an extensible flexfield named Sample Requested. Here are the values it can contain.

- Complimentary
- Charge
- No Sample

You can create a transformation rule.

- If the value is Complimentary or Charge, then add a fulfillment line for the sample item.

You can then set up a rule that populates price details for the sample lines according to whether the value is Complimentary or Charge.

Make sure you run the defaulting rule during one of these types of transformations when you process the transformation rule. You can't default an attribute value into an extensible flexfield.

Type of Transformation	Description
Pretransformation	Use an extensible flexfield when some other transformation logic requires the default attribute value.

Type of Transformation	Description
	For example, if you must use the defaulted attribute value to determine whether to add a free sample item during transformation, then define a pretransformation rule that sets the default value.
Posttransformation	Use Posttransformation when transformation must happen before you run your defaulting rule. For example, if you plan to add an item during transformation, and you also plan to use an extensible flexfield attribute on the order header to determine the defaulted value to set for the new fulfillment line.

Use an extensible flexfield in a business rule. For details, see *Identify Flexfield Contexts and Category Codes for Your Business Rules*.

Import and Integrate

You can update an extensible flexfield on a closed fulfillment line or closed sales order only in the Order Management work area. You can't through import or integration, such as through file based data import, a web service, REST API, or Oracle Application Development Framework (ADF).

Import Source Orders

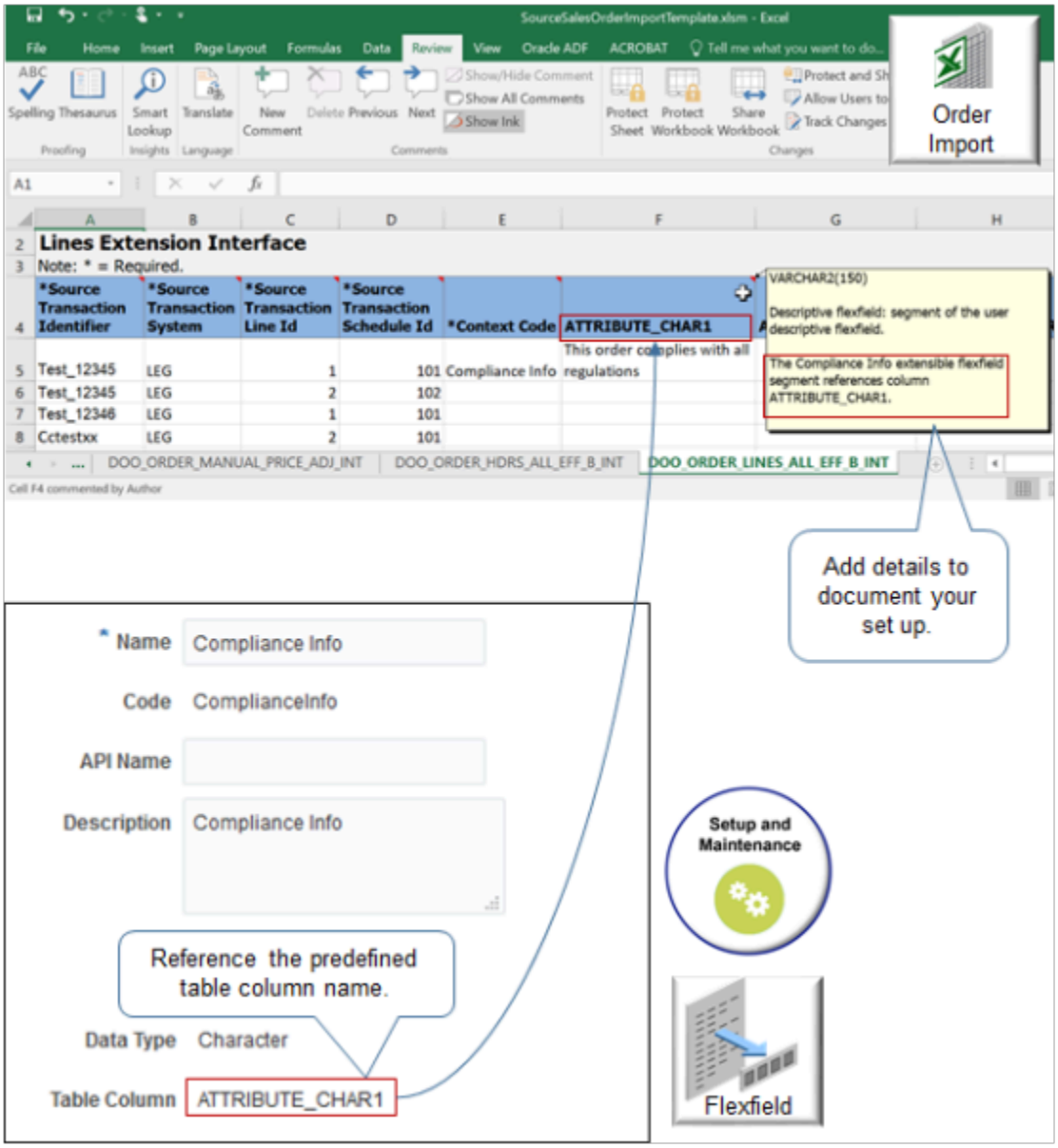
Apply guidelines when you use the order import template to import data for an extensible flexfield.

- Assign the Table Column attribute of each segment so it references the correct column in the order import template.
- Add a comment in Excel to document your set up.
- Use the DOO_ORDER_HDRS_ALL_EFF_B_INT worksheet to capture extensible flexfield details for order headers.
- Use the DOO_ORDER_LINES_ALL_EFF_B_INT worksheet to capture extensible flexfield details for fulfillment lines.
- Use columns, such as ATTRIBUTE_CHAR1, to capture details for each attribute.
- Don't modify worksheet names or column names. Instead, add a comment.

You can't add an extensible flexfield on an order line entity. You can add one only on the order header entity or the fulfillment line entity. The template uses the phrase DOO_ORDER_LINES, but it uses details that you add on the DOO_ORDER_LINES_ALL_EFF_B_INT worksheet to interact with extensible flexfields on the fulfillment line, not on the order line.

For details, see *Import Orders Into Order Management*.

This example illustrates how to set up the `Compliance Info` segment that references the ATTRIBUTE_CHAR1 column.



Use Web Services to Import Extensible Flexfield Details

Add extensible flexfield details to the payload that Order Management uses to communicate with the web service.

- Get an example payload. For details, see *Example Web Service Payloads That Integrate Order Management*.

  A range of examples are available, such returns, coverage, configurations, and so on. These payloads include most of the sales order data that you require.
- If you create your own payload, then create it according to the WSDL definition that the payload requires. For details, see *Use Your Own Payload to Import Extensible Flexfields*.
- Run the Publish Extensible Flexfield Attributes scheduled process.
- Copy extensible flexfield data from the log file that the scheduled process creates into the XSD file that contains your payload. Copy this data onto each object that requires extensible flexfield data. This approach helps to keep your channel system synchronized with the set up in Order Management.

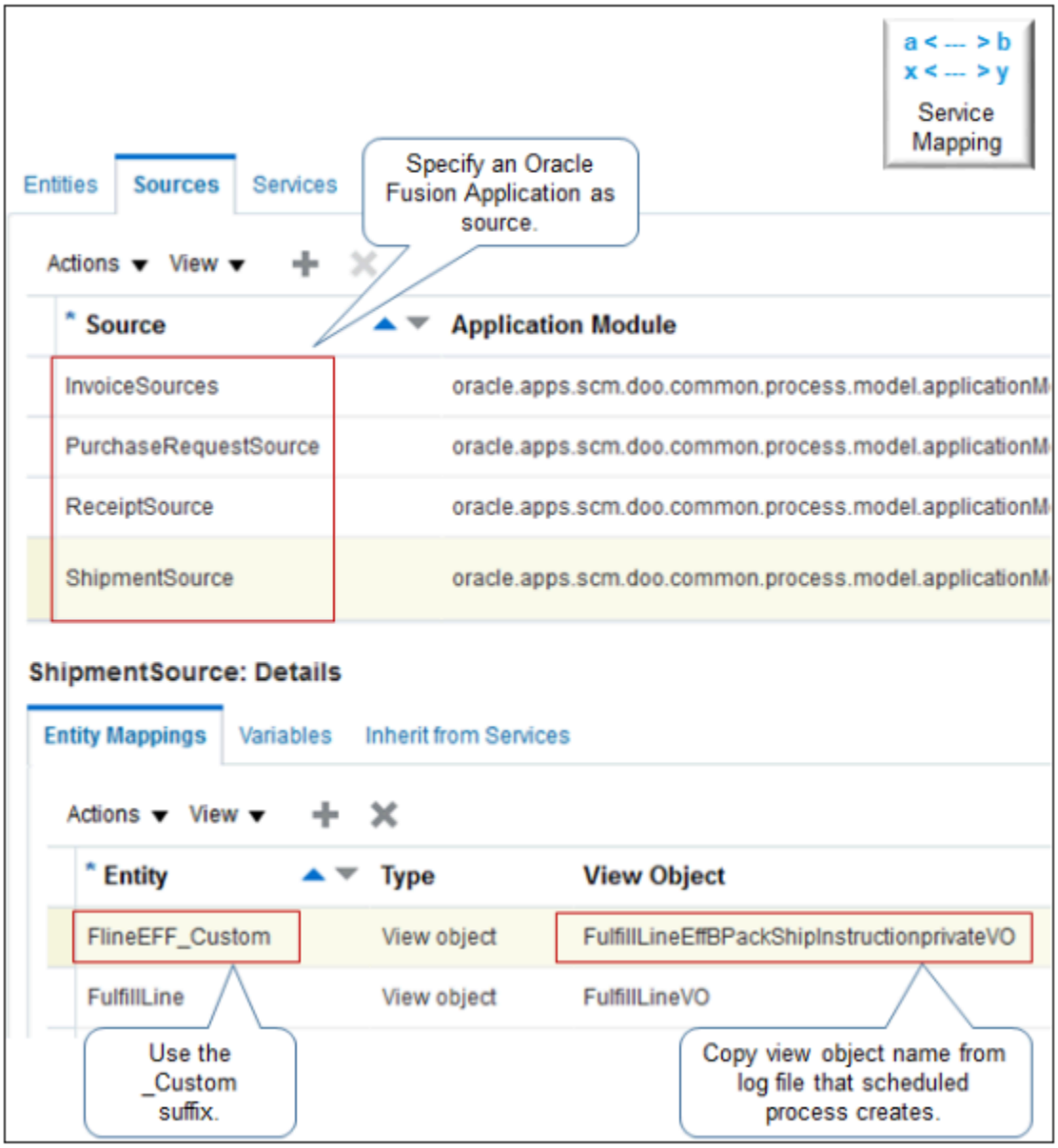
- Use the import service from Application Development Framework (ADF). Don't use the SOA (Service Oriented Architecture) import service.

Here's an example payload that copies one section of code from the log file into the XSD file.

For details, see *Guidelines for Using Web Services to Integrate Order Management*.

Integrate with Other Oracle Applications

Use an extensible flexfield when you integrate with some other Oracle Application, such as Pricing, Receivables, Shipping, Receiving, or Purchasing. You set up a service mapping that's similar to a service mapping that you set up for Oracle Pricing. You use an extensible flexfield to store values that the service mapping requires.



Note

- Copy the name of the view object from the log file. The service mapping uses the view object to get data from your extensible flexfield. Use the log file that the Publish Extensible Flexfield Attributes scheduled process creates.
- Use the _Custom suffix when you define each new entity.
- Use an integration algorithm to implement complex logic. For example, assume you must concatenate an item with the coverage that covers the item, add quantity, then store the results in a descriptive flexfield for Accounts Receivable. You would use an integration algorithm to implement this requirement.

Search for Extensible Flexfields in the Order Management Work Area

Assume you need to search an extensible flexfield on the 56794 sales order.

1. Go to the Order Management work area.

2. On the Overview page, **Tasks > Manage Fulfillment Lines**.
3. Click **Add Fields > Extensible Flexfield Category**.
4. Set the values.

Attribute	Value
Order	56794
Extensible Flexfield Category	Equals Additional Fulfillment Line Information

5. Click **Search**.
6. In the search results, scroll all the way to the right, then examine the extensible flexfield attributes. They will have the names that you set up for the flexfield, such as the context and category.

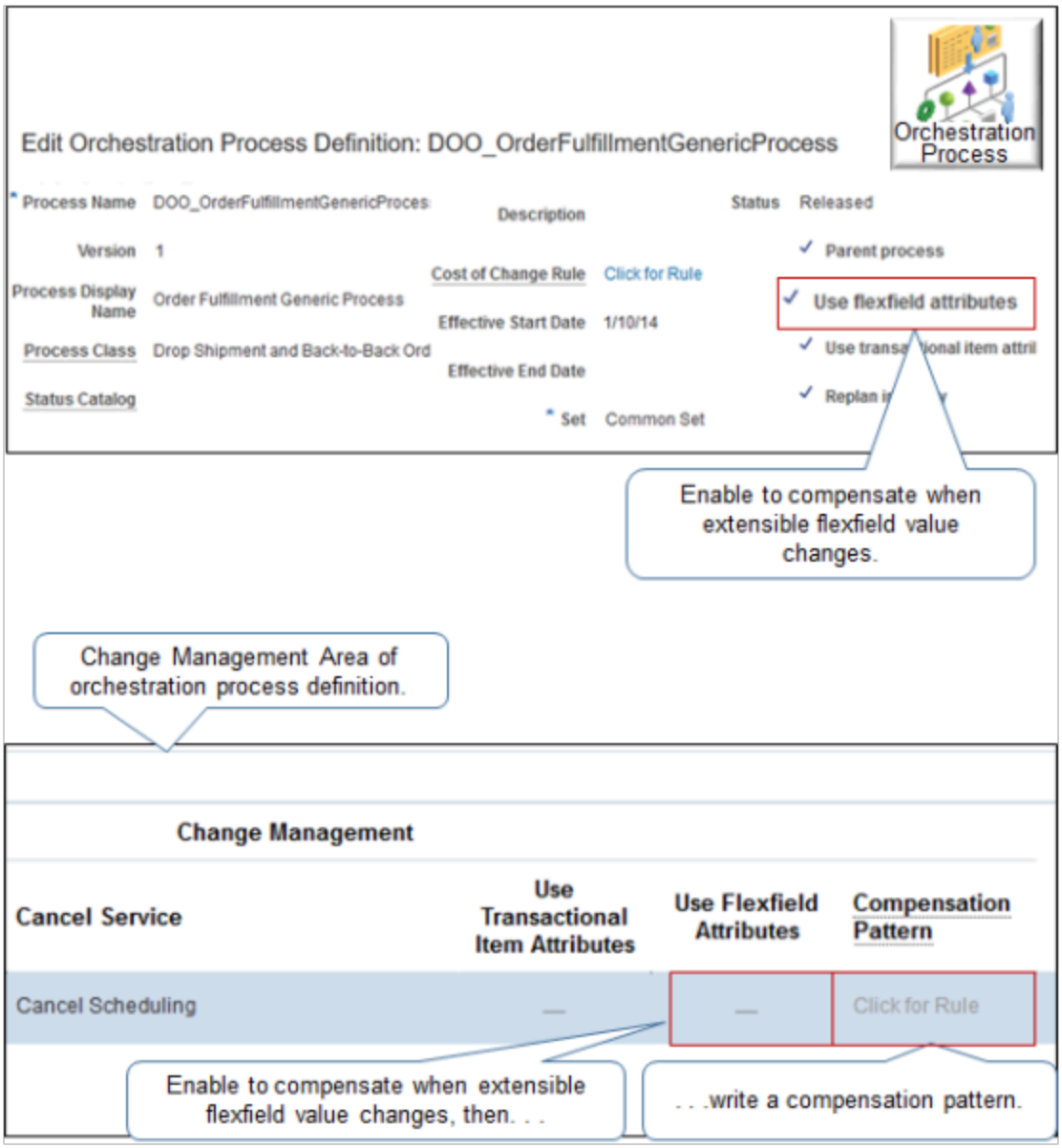
   As an option, click **View > Columns**, then add or remove individual attributes from the search results.
7. Click **Save** to save your search. This way, you can access your saved search the next time you need to see flexfield details.

Here's how you can search for flexfields on the sales order.

1. Go to the Order Management work area.
2. On the Overview page, click **Tasks > Manage Orders**.
3. On the Manage Orders page, click **Add Fields > Extensible Flexfield Category**.
4. Set the attributes.

Attribute	Value
Order	56794
Extensible Flexfield Category	Equals Additional Header Information

Manage Change Orders



Note

- Identify your requirements for change management. For example, determine whether the orchestration process must compensate when you change the value in an extensible flexfield, and the action to take during compensation, such as scheduling, calling Accounts Receivable, and so on.

- Compensate when a value that the orchestration process references changes.
    - **Any extensible flexfield**. Enable the Use Flexfield Attributes option on the header of the orchestration process definition.
    - **A specific extensible flexfield**. Enable the Use Flexfield Attributes option in the Change Management area of the orchestration process definition. Add it to the orchestration process step that references the extensible flexfield. Also define a compensation pattern on this step.

      Use the compensation pattern to specify the action to do, such as Cancel, Create, Update, and so on, depending on the value that changes. You can configure your set up to send an update when a single value changes and to skip sending an update when other values change. This approach provides more detailed control regarding the compensation you do.

      Here's an example compensation pattern.

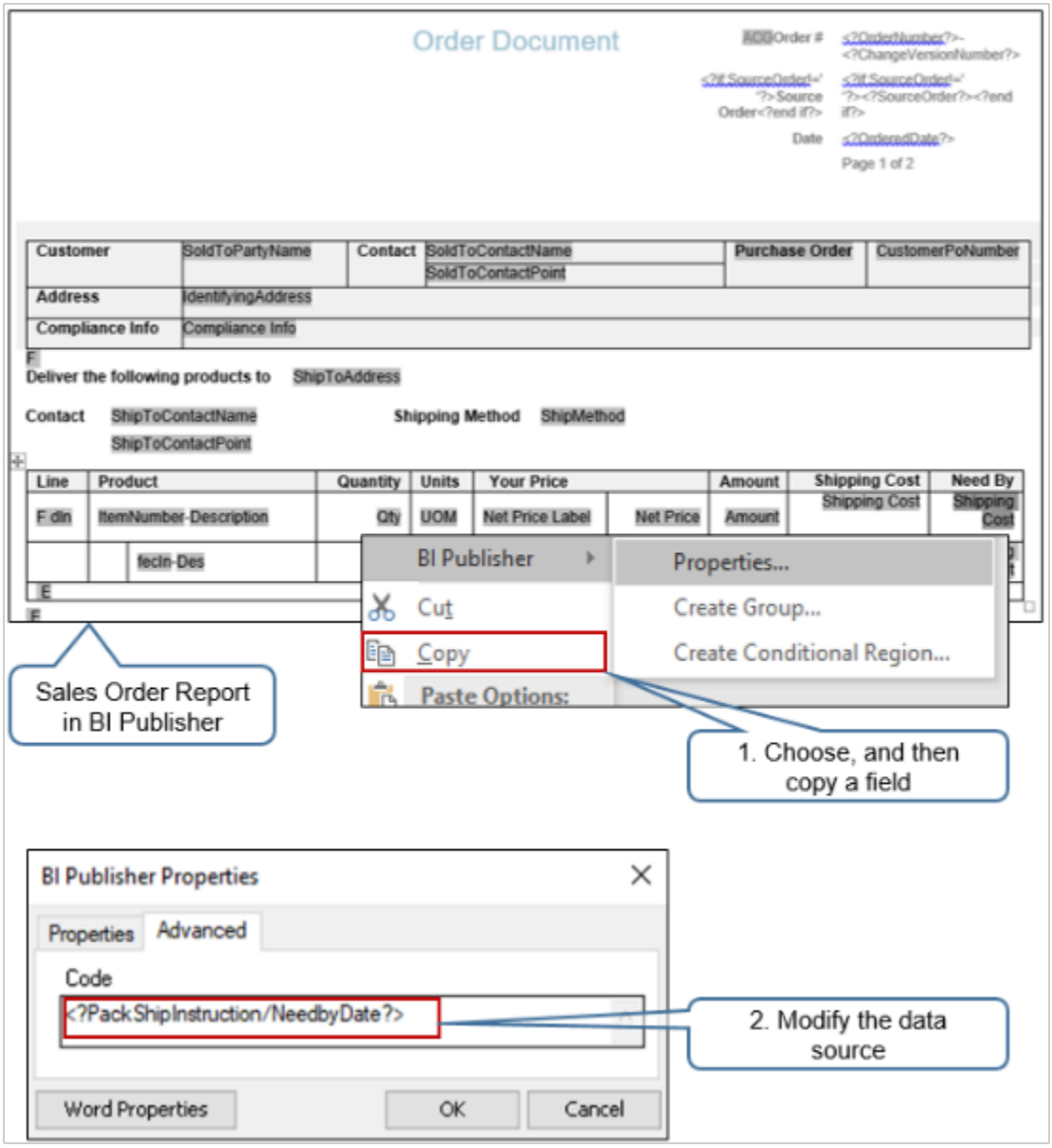
      ```
      pseudocode:
      If segment3 changes, then cancel task x, and then recreate task x.
      If segment4 changes, then do nothing.
      If segment5 changes, then call an update service.
      ```

      Examine a detailed code example. For details, see *Compensate Sales Orders That Change*.
    - To ignore changes to all extensible flexfield values, make sure Use Flexfield Attributes for each step in the Change Management area isn't enabled.

Change management ignores each change that the Order Entry Specialist makes to an extensible flexfield value that you set up on the sales order header.

Use Reports and Analytics

Modify Reports



Note

- The Order Entry Specialist can use the Sales Order report when creating or viewing a sales order.

- Add extensible flexfield data to the report.
    - Install Oracle Analytics Publisher Desktop,
    - Use the Edit Sales Order Report action in Oracle Analytics Publisher. This action displays Sales Order Report in rich text format so you can modify it.
    - Right-click a **field**, then click **Copy**.
    - Use Oracle Analytics Publisher Properties to specify the source that the report uses to display data for the field.

      Use this format.

      ```
      <?context/segment?>
      ```

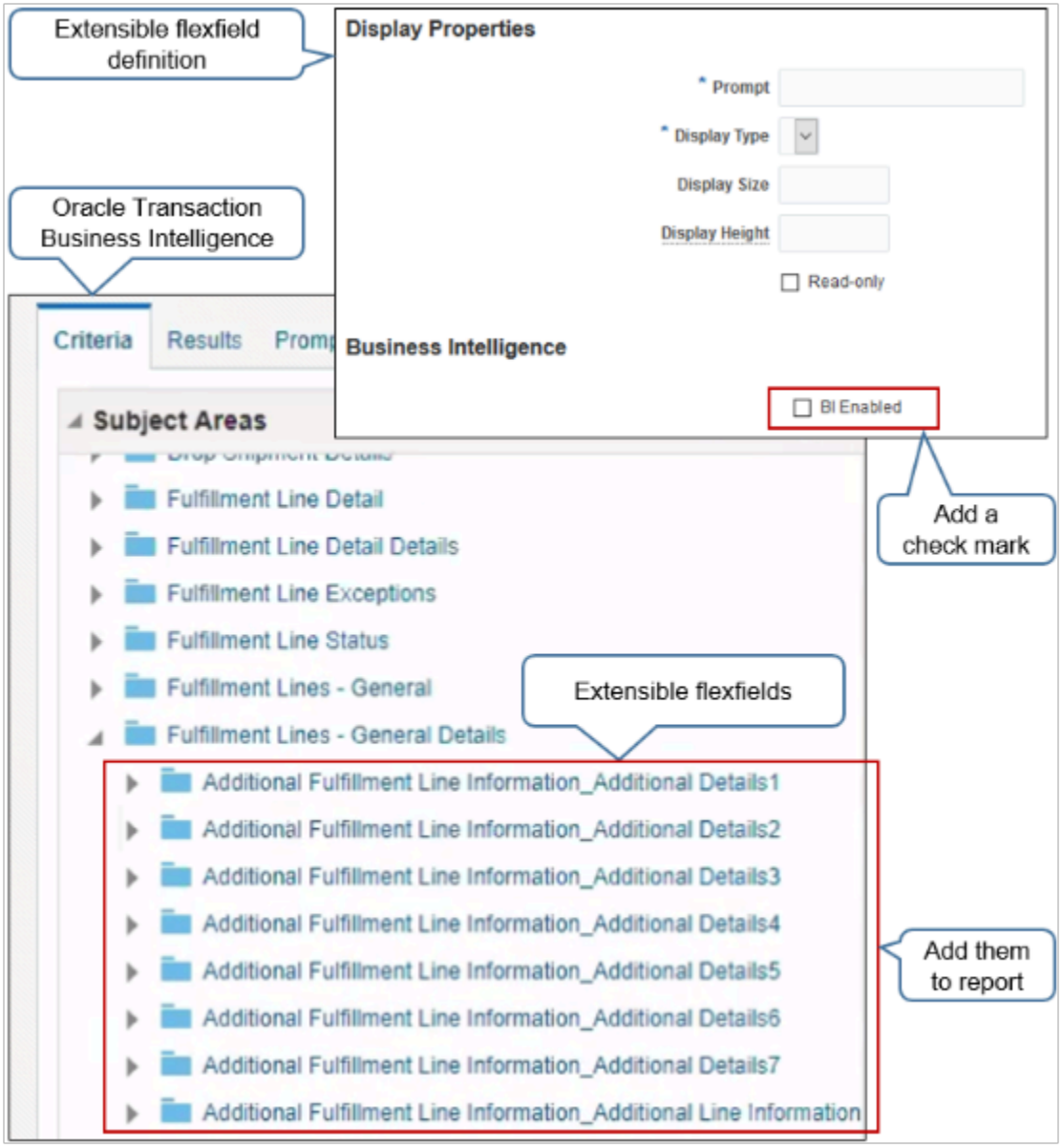
      where
    - **context**. Name of the extensible flexfield context.
    - **/ (forward slash)**. You must use the forward slash between context and segment.
    - **segment**. Name of the extensible flexfield segment.
    - You must use a set of question marks (??) to enclose `context/segment`.
    - You must use a set of angle brackets (< >) to enclose the question marks.

  For example, assume you set up an extensible flexfield that uses Context1 to store details about trade compliance, and SegmentA to store the compliance description. Here's the code you use.

  ```
  <?Context1/SegmentA?>
  ```

Modify Analytics

Oracle Transaction Business Intelligence (OTBI) provides analytic reporting. You can display extensible flexfield data in OTBI.



Note

- Enable the BI Enabled option when you set up your extensible flexfield.
- Run the Import Oracle Fusion Data Extensions for Transactional Business Intelligence scheduled process. This process updates OTBI so it displays your extensible flexfields in area Subject Areas in OTBI. For details, see *Overview of Flexfield Change Import*.
- Add your extensible flexfields from Subject Areas into a report in OTBI.
- Here's the format that OTBI uses.

  `Category_Context`

  For example:

  `Additional Fulfillment Line Information_Additional Details1`

where

- `Additional Fulfillment Line Information` is the category
- `Additional Details1` is the context

And Yet More Guidelines

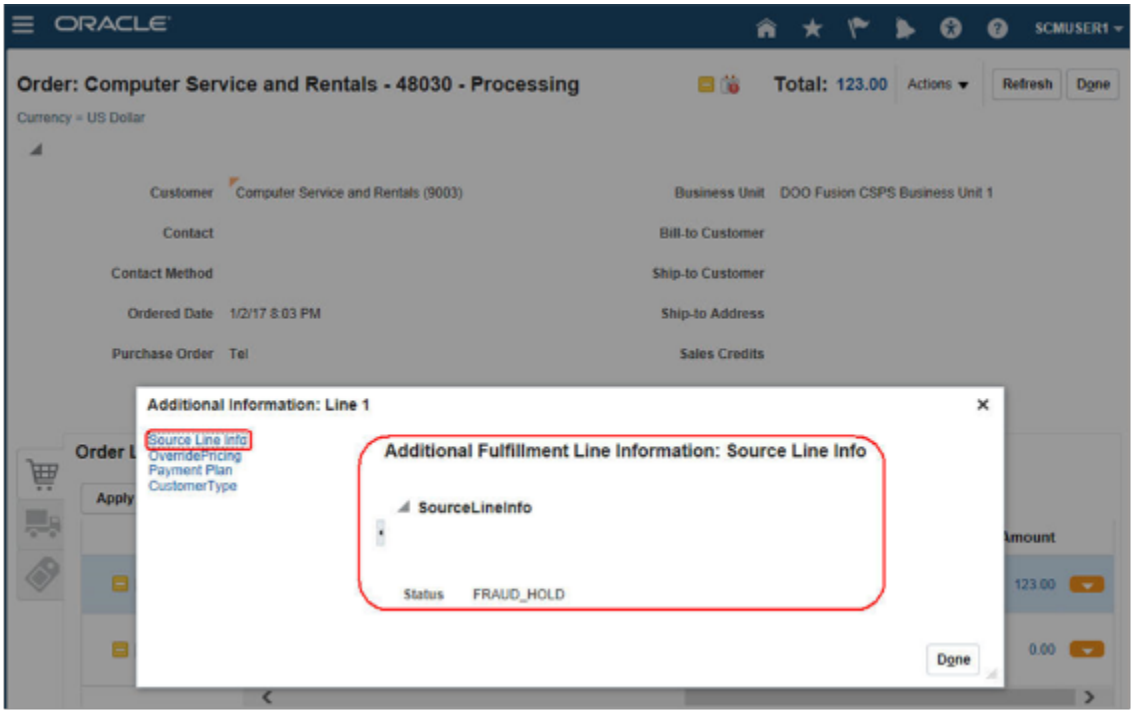
XML doesn't support a context name or segment name that begins with a number. So, if you begin the name of your context or segment with a number, then Order Management automatically prefaces the name with an underscore (_) at run time. For example, if you name your context 1MyContext, then Order Management automatically renames it to _1MyContext.

Set Up

Set Up Extensible Flexfields in Order Management

Set up an extensible flexfield in Oracle Order Management.

In this example you set up an extensible flexfield so the Order Entry Specialist can view the sales order status from the source order in the Order Management work area.



Summary of the Steps

1. Create the context.
2. Associate the context with the category.
3. Add the page to the category.
4. Deploy the extensible flexfield.
5. Publish the extensible flexfield.

6. Test your set up.

For background, see *Overview of Setting Up Extensible Flexfields in Order Management*.

This topic uses example values. You might need different values, depending on your business requirements.

For a detailed example that includes lots of screen prints, see *Implementing and Populating Extensible Flex Fields (EFF) in Order Management (Doc ID 2005737.1)*.

Create the Context

1. Make sure you have the privileges that you need to administer Order Management.

Attribute	Value
Privilege Name	Publish Extensible Flexfield Attributes

   For details, see *Privileges That You Need to Implement Order Management*.
2. Go to the Setup and Maintenance work area, then go to the task.
   - Offering: Order Management
   - Functional Area: Orders
   - Task: Manage Order Extensible Flexfields
3. On the Manage Order Extensible Flexfields page, enter the value, then click **Search**.

Attribute	Value
Name	Fulfillment Line Information

4. In the search results, click **Actions > Edit**.
5. On the Edit Extensible Flexfield page, click **Manage Contexts**.
6. On the Manage Contexts page, click **Search**.

   Scan the predefined contexts in the search results and see if one meets your needs.

   For this example, you will add a new context.
7. Click **Actions > Create**.
8. On the Create Context page, set values.

Attribute	Value
Display Name	SourceLineInfo
Code	SourceLineInfo
API Name	Sourcelineinfo

Attribute	Value
Enabled	Contains a check mark.
Translatable	Leave empty. Don't enable this option. You can't use it with Oracle Order Management.
Behavior	Single Row You must use Single Row for any extensible flexfield that you use in Order Management.

Values for some attributes, such as Code, automatically populate. In general, don't modify the predefined values.

9. On the Context Usages tab, click **Actions > Create**, set the value, then click **Save**.

Attribute	Value
Name	Additional Fulfillment Line Information

10. In the Context Sensitive Segments area, click **Actions > Create**.
11. On the Create Segment page, set values.

Attribute	Value
Name	Status
Code	Status
API Name	status The Create Segment page automatically sets the value for API Name. In this example, it sets the value to `status`, and uses lower case for the first letter. You must use the same value for API Name that the XML uses when it creates the sales order.
Enabled	Contains a check mark.
Data Type	Character
Table Column	ATTRIBUTE_CHAR1

Attribute	Value
	You can use any column. Select a column that you aren't already using to store some other value. To view the values that are available, click the **down arrow**, then click **Search**. In the Search and Select dialog, remove the value that displays in the Name attribute, then click **Search**.
Value Set	10 Characters You can use any value that meets your needs.
Prompt	Status
Display Type	Text Box

For details about these attributes, go to *Configuring and Extending Applications*, then search for Flexfield Segment Properties.

12. Click **Save and Close**. Repeat until you're back on the Edit Extensible Flexfield page.

Associate the Context with the Category

1. On the Edit Extensible Flexfield page, in the Category area, in the Display Name column, click the text **Additional Fulfillment Line Information**.

   Notice that the Associated Contexts tab displays the category details.
2. In the Associated Contexts tab, click **Actions > Select and Add**.
3. In the Select and Add dialog, in the Name attribute, enter `SourceLineInfo`, which is the context you created earlier in this topic, then click **Search**.
4. In the search results, immediately above OK, click the **row**, then click **Apply > OK**.
5. On the Edit Extensible Flexfield page, click **Save**.

Add the Page to the Category

Now let's tell the Order Management work area where to display the extensible flexfield.

1. In the Details area, click **Pages**.

   Note

   - You use the Pages tab to assign each context to a page.
   - The Sequence attribute determines the sequence that Order Management uses when it displays the extensible flexfield.
2. On the Pages tab, click **Actions > Create**.

3. In the Create Page dialog, set values, then click **OK**.

Attribute	Value
Display Name	`Source Line Info`
Code	`SourceLineInfo` Don't include spaces.
Usage	Additional Fulfillment Line Information

4. On the Edit Extensible Flexfield page, click **Save**.
5. In the Details area, click the row that has the value.

Attribute	Value
Display Name	Source Line Info

6. In the Associated Contexts Details area, click **Actions > Select and Add**.
7. In the Select and Add dialog, search for the value.

Attribute	Value
Name	`SourceLineInfo`

8. In the search results, click the **row** that contains `SourceLineInfo` in the Name column, then click **Apply > OK**.
9. On the Edit Extensible Flexfield page, click **Save and Close**.

Deploy the Extensible Flexfield

1. On the Manage Order Extensible Flexfields page, in the search results, click the **row** that includes the value.

Column	Value
Name	Fulfillment Line Information

2. Click **Deploy Flexfield**.
3. In the dialog that displays, wait for the deployment to finish, then click **OK**.
4. On the Manage Order Extensible Flexfields page, verify that the Deployment Status attribute contains a check mark, then click **Done**.

Publish the Extensible Flexfield

You must publish each time you change the set up for your extensible flexfield. If you're using an extensible flexfield in a business rule or processing constraint, then you must publish so you can reference the flexfield from a rule fact or constraint.

1. Go to the Scheduled Processes work area, then click **Actions > Schedule New Process**.
2. In the Schedule New Process dialog, search for *Publish Extensible Flexfield Attributes*, then click **OK**.
3. In the Process Details dialog, click **Submit**.
4. In the Confirmation dialog, copy the Process ID to the clipboard, such as 68721, then click **OK**.
5. Click **Actions > Refresh**, then notice the status of your process.
6. The status is likely Running. Wait a few minutes, then click **Refresh** again until the status is Succeeded.

Example Payload That Includes Extensible Flexfields

extensible_flexfield_example_payload.xml includes code that's part of the payload that Order Management creates for the sales order when it submits a sales order that has an extensible flexfield. It includes the XML that defines the namespace. You can access the file *here*.

Note

- Context Code determines `SourceLineInfo`.
- Segment API Name determines ns22:status.
- The status value is FRAUD_HOLD. You will confirm this value later when you create a sales order.
- Learn how to determine what attribute names you need to use in the namespace for an extensible flexfield. For details, see *How to Include EFF Attributes in the Order Creation Payload Invoking Web Service (Doc ID 2195245.1)*.

Test Your Set Up

Verify that the Order Management work area correctly displays your extensible flexfield.

1. Go to the Order Management work area.
2. Create a new sales order, then add an order line.
3. At the far right of the order line, click the **down arrow**, then click **Edit Additional Information**.
4. In the Edit Additional Information dialog, verify that it displays Source Line Info, and that it includes the FRAUD_HOLD status value that you noticed earlier in this topic when you examined the payload.
5. As an option, use SQL (Structured Query Language) to query extensible_flexfield_example_sql.xml. Verify the values that you defined in this topic. Click *here* to access this file.

   Here are the values that your SQL should return.

Attribute	Value
SOURCE_ORDER_NUMBER	PMC-170113-001
ORDER_NUMBER	48030
FULFILL_LINE_NUMBER	1
CONTEXT_CODE	SourceLineInfo

Attribute	Value
ATTRIBUTE_CHAR1	FRAUD_HOLD

Related Topics

- Privileges That You Need to Implement Order Management

Use Your Own Payload to Import Extensible Flexfields

You typically modify a predefined, example payload that imports an extensible flexfield. But you can also create your own payload that meets your specific requirements.

- You will create an example payload that uses the ReceiveOrderRequestService web service to import a test extensible flexfield.
- This topic describes the minimum details you must include.
- Learn how to modify an example predefined payload instead. For details, see *Example Web Service Payloads That Integrate Order Management*.

Summary of the Steps

1. Create and import the payload.
2. Verify you imported the context code.
3. Verify you imported the flexfield segment.

Create and Import the Payload

1. Create the payload.
    - Add header details.

```
<ns2:AdditionalHeaderInformationCategories xsi:type="ns12:j_HeaderEffDooHeadersAddInfoprivate"
xmlns:ns3=http://xmlns.oracle.com/apps/scm/doo/processOrder/service/
xmlns:ns12=http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerCategories/
xmlns:ns22=http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerContextsB/
xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_HEADERS_ADD_INFO</ns8:Category>
 <ns8:SourceTransactionLineIdentifier>1</ns8:SourceTransactionLineIdentifier>
 <ns8:SourceTransactionScheduleIdentifier>1</ns8:SourceTransactionScheduleIdentifier>
 <ns12:HeaderEffBPREVIOUS_5FSO_5FREFprivateVO>
 <ns8:ContextCode>PREVIOUS_SO_REF</ns8:ContextCode>
 <ns22:locationname>West Coast</ns22:locationname>
 </ns12:HeaderEffBPREVIOUS_5FSO_5FREFprivateVO>
 </ns2:AdditionalHeaderInformationCategories>
```

    - Add fulfillment line details.

```
<ns2:AdditionalFulfillmentLineInformationCategories
xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate"
xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineCategories/"xmlns:ns22=http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineContextsB/
xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
```

```
<ns8:SourceTransactionLineIdentifier>1</ns8:SourceTransactionLineIdentifier>
<ns8:SourceTransactionScheduleIdentifier>1</ns8:SourceTransactionScheduleIdentifier>
<ns12:FulfillLineEffBADD_CONTEXT_HEREprivateVO>
<ns8:ContextCode>ADD_CONTEXT_HERE</ns8:ContextCode>
<ns22:ADD_SEGMENT_HERE>Working</ADD_SEGMENT_HERE>
</ns12:FulfillLineEffBADD_CONTEXT_HEREprivateVO>
</ns2:AdditionalFulfillmentLineInformationCategories>
</ns2:OrchestrationOrderRequestLine>
```

where

- You change ADD_CONTEXT_HERE to the context code.
- You change ADD_SEGMENT_HERE to the segment.
- You add extensible flexfields details as the last line of the order line details, immediately before `</ns2:OrchestrationOrderRequestLine>`.

For example:

```
<ns2:AdditionalFulfillmentLineInformationCategories
xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate"
xmlns:ns12=http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineCategories/
xmlns:ns22=http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <ns8:SourceTransactionLineIdentifier>1</ns8:SourceTransactionLineIdentifier>
 <ns8:SourceTransactionScheduleIdentifier>1</ns8:SourceTransactionScheduleIdentifier>
 <ns12:FulfillLineEffBPCTESTAUGprivateVO>
 <ns8:ContextCode>PCTESTAUG</ns8:ContextCode>
 <ns22:pctestaugseg1>Working</pctestaugseg1>
 </ns12:FulfillLineEffBPCTESTAUGprivateVO>
 </ns2:AdditionalFulfillmentLineInformationCategories>
</ns2:OrchestrationOrderRequestLine>
```

where

- PCTESTAUG is the context.
- pctestaugseg1 is the segment. PCTESTAUG includes only one segment.

Notice the value for the FulfillLineEffBPCTESTAUGprivateVO virtual object. You will verify it later.

2. Use ReceiveOrderRequestService to import the payload. For details, see *Guidelines for Using Web Services to Integrate Order Management*.

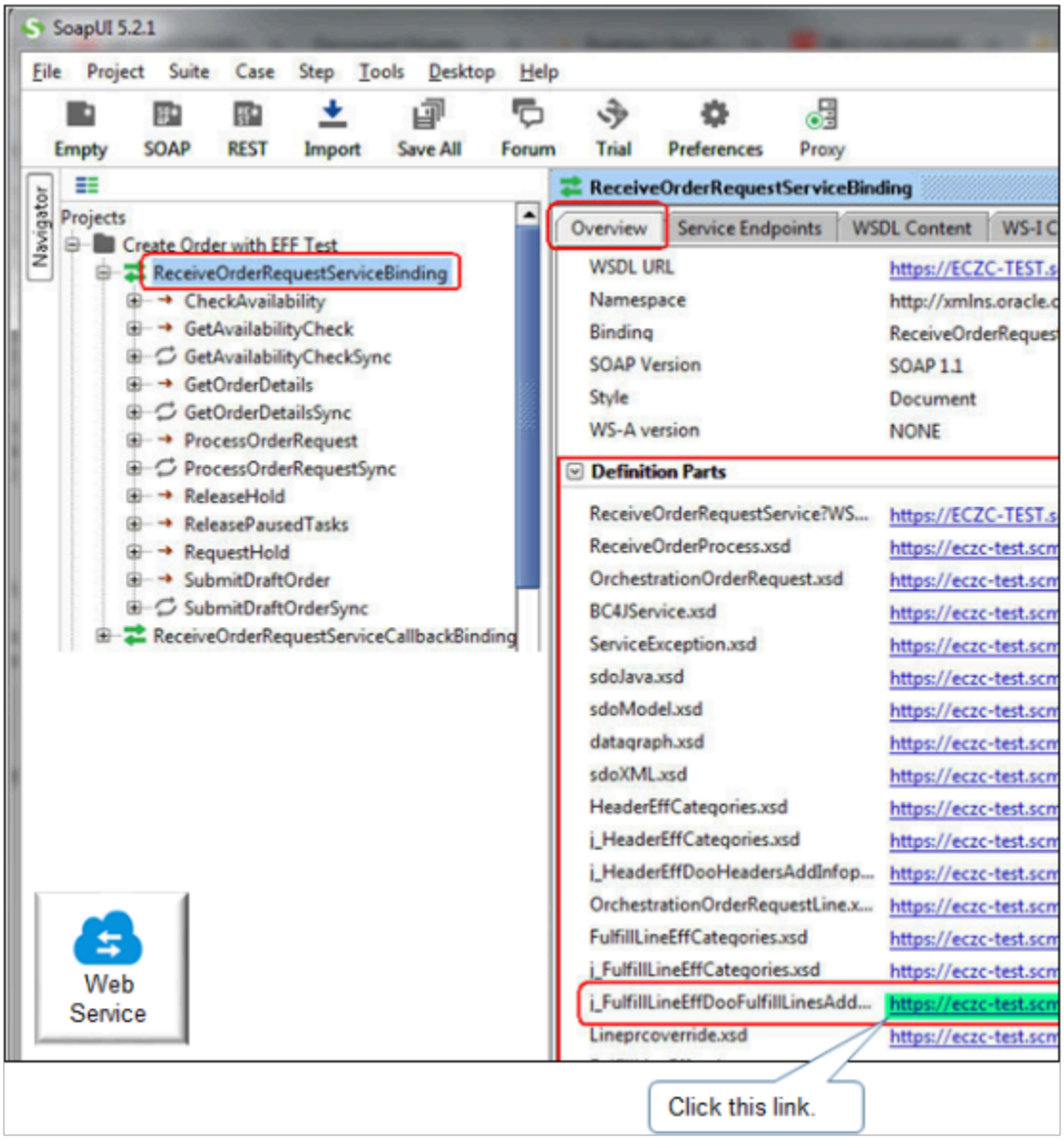
Verify You Imported the Context Code

1. Open SoapUI, then create a project.

Attribute	Value
Project Name	Create Order with EFF Test
Initial WSDL	http://host:port/ soa-infra/services/default/DooDecompReceiveOrderExternalComposite/ReceiveOrderRequestServiceWSDL Replace host:port with your server address.

Attribute	Value
Create Requests	Enabled.

2. Verify the virtual object.
    - In the Projects area, double click **ReceiveOrderRequestServiceBinding**.
    - On the Overview tab, in the Definition Parts area, click the **link** next to J_FulfillLineEFFDooFulfillLinesAdd.



Your browser displays the XSD.

```
<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:ns0="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineContextsB/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:sdoXML="commonj.sdo/
xml" xmlns:sdo="commonj.sdo" xmlns="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineCategories/" targetNamespace="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineCategories/" elementFormDefault="qualified">
 <xsd:import schemaLocation="../fulfillLineContextsB/Lineprcoverride.xsd" namespace="http://
xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/"></xsd:import>
```

```
 <xsd:import schemaLocation="../fulfillLineContextsB/Pctestaug.xsd" namespace="http://
xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/"></xsd:import>
 <xsd:import schemaLocation="https://eczc-test.scm.em2.oraclecloud.com:443/soa-infra/services/
default/DooDecompReceiveOrderExternalComposite/soa-cp/xml/sdoXML.xsd" namespace="commonj.sdo/
xml"></xsd:import>
 <xsd:import schemaLocation="https://eczc-test.scm.em2.oraclecloud.com:443/soa-infra/services/
default/DooDecompReceiveOrderExternalComposite/soa-cp/xml/sdoModel.xsd" namespace="commonj.sdo"></
xsd:import>
 <xsd:include schemaLocation="j_FulfillLineEffCategories.xsd"></xsd:include>
 <xsd:complexType name="j_FulfillLineEffDooFulfillLinesAddInfoprivate">
 <xsd:annotation>
 <xsd:appinfo source="http://xmlns.oracle.com/adf/svc/metadata/">
 <key xmlns="http://xmlns.oracle.com/adf/svc/metadata/">
 <attribute>FulfillLineId</attribute>
 </key>
 </xsd:appinfo>
 </xsd:annotation>
 <xsd:complexContent>
 <xsd:extension base="j_FulfillLineEffCategories">
 <xsd:sequence>
 <xsd:element name="FulfillLineEffBLinePrcOverrideprivateVO" type="ns0:Lineprcoverride"
 minOccurs="0" sdoXML:dataType="sdo:DataObject"></xsd:element>
 <xsd:element name="FulfillLineEffBPCTESTAUGprivateVO" type="ns0:Pctestaug" minOccurs="0"
 sdoXML:dataType="sdo:DataObject"></xsd:element>
 </xsd:sequence>
 </xsd:extension>
 </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="j_FulfillLineEffDooFulfillLinesAddInfoprivate"
 type="j_FulfillLineEffDooFulfillLinesAddInfoprivate"></xsd:element>
</xsd:schema>
```

- In `dataType="sdo:DataObject"`, verify that `xsd:element name=` contains FulfillLineEffBPCTESTAUGprivateVO.

  This value must match the value you included earlier in your import payload. For example, `<ns12:FulfillLineEffBPCTESTAUGprivateVO>`.

3. Verify the context code.
   - Make sure you have the privileges that you need to administer Order Management.
   - In the Setup and Maintenance work area, go to the task.
     - Offering: Order Management
     - Functional Area: Orders
     - Task: Manage Order Extensible Flexfields
   - On the Manage Extensible Flexfields page, search for the value.

Attribute	Value
Flexfield Code	DOO_FULFILL_LINES_ADD_INFO

Attribute	Value

- In the search results, click **Actions > Edit**.
- On the Edit Extensible Flexfield page, click **Manage Contexts**.
- On the Manage Contexts page, search for the value.

Attribute	Value
Code	PCTESTAUG

- Verify the search results display your context, then click **Actions > Edit**.
- On the Edit Context page, verify the context contains the values you imported.

On the header.

Attribute	Value
Code	PCTESTAUG
API Name	`Pctestaug` Note this value. You will use it later when you verify the segment.

In the Context Sensitive Segments area.

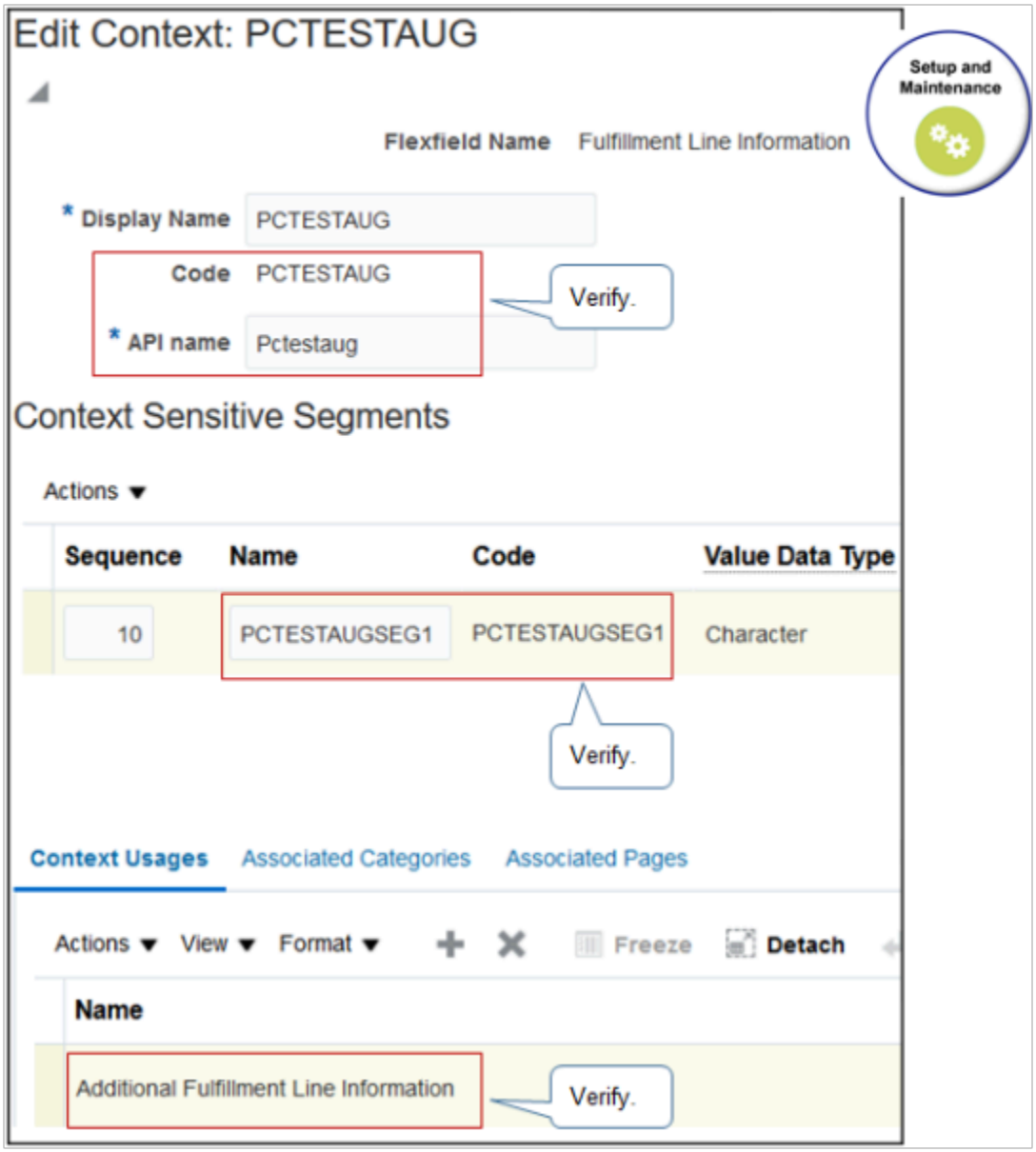
Attribute	Value
Name	PCTESTAUGSEG1
Code	PCTESTAUGSEG1

On the Context usages tab, make sure a row exists that contains this value.

Attribute	Value
Name	Additional Fulfillment Line Information

Attribute	Value

For example:



If you prefer not to use the Manage Order Extensible Flexfields page, then as an alternative, run a SQL query against the Oracle database.

```
select
 fdct.application_ID,
 fdct.descriptive_flexfield_code,
 fdct.context_code,
 fdcb.context_identifier,
```

```
 fdcb.enabled_flag,
 fdct.description
from
 fnd_df_contexts_TL fdct,
 fnd_df_contexts_B fdcb
where
 fdct.context_code = fdcb.context_code and
 fdct.application_id = fdcb.application_id and
 fdct.language = 'US' and
 fdct.descriptive_flexfield_code = 'DOO_FULFILL_LINES_ADD_INFO';
```

Here's an example of data the query might return.

APPLICATION_ID	DESCRIPTIVE_ FLEXFIELD_CODE	CONTEXT_CODE	CONTEXT_IDENTIFIER	ENABLED_FLAG
10008	DOO_FULFILL_LINES_ ADD_INFO	LinePrcOverride	Lineprcoverride	Y
10008	DOO_FULFILL_LINES_ ADD_INFO	Accounting_Rule	AccountingRule	N
10008	DOO_FULFILL_LINES_ ADD_INFO	PCTESTAUG	`Pctestaug`	Y

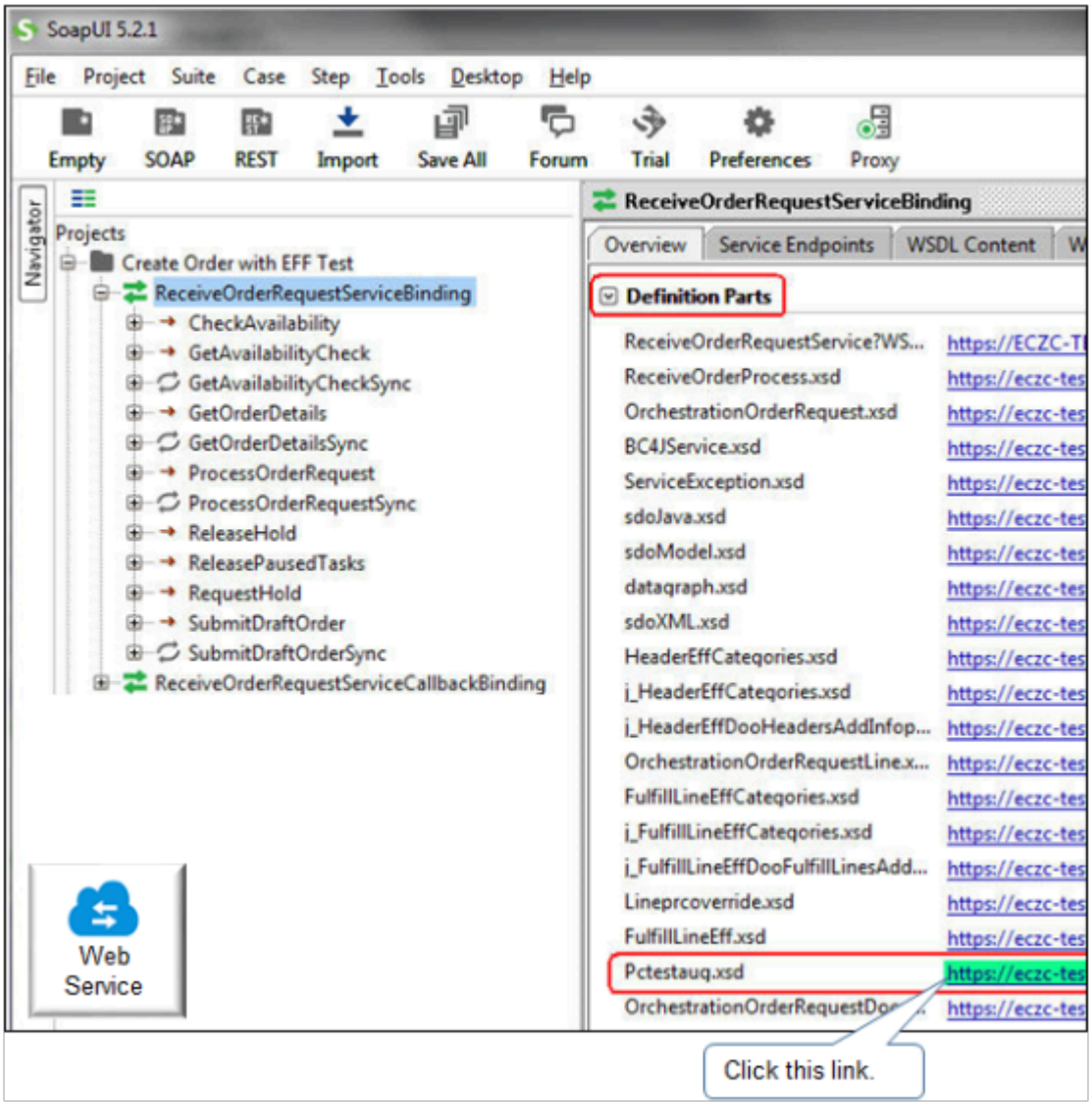
PCTESTAUG is the CONTEXT_CODE for this example.

Verify You Imported the Flexfield Segment

1. In SoapUI, in the Projects area, double click **ReceiveOrderRequestServiceBinding**.

2. On the Overview tab, in the Definition Parts area, click the **link** next to Pctestaug.xsd.

   `Pctestaug` is the value of attribute API Name you noted earlier on the Edit Context page.



Your browser displays the XSD.

```
<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:ns0="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineContextsB/" targetNamespace="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineContextsB/" elementFormDefault="qualified"><xsd:include xmlns:id="Pctestaug-xsd"
 schemaLocation="Pctestaug.xsd"></xsd:include>
 <xsd:import schemaLocation="../../model/FulfillLineEff.xsd" namespace="http://xmlns.oracle.com/apps/
scm/doo/processOrder/model/"></xsd:import>
 <xsd:complexType name="Pctestaug">
 <xsd:annotation>
 <xsd:appinfo source="http://xmlns.oracle.com/adf/svc/metadata/">
 <key xmlns="http://xmlns.oracle.com/adf/svc/metadata/">
 <attribute>EffLineId</attribute>
 </key>
```

```
 </xsd:appinfo>
 </xsd:annotation>
 <xsd:complexContent>
 <xsd:extension base="ns0:FulfillLineEff">
 <xsd:sequence>
 <xsd:element name="pctestaugseg1" type="xsd:string" minOccurs="0" nillable="true"></xsd:element>
 </xsd:sequence>
 </xsd:extension>
 </xsd:complexContent>
 </xsd:complexType>
 <xsd:element name="pctestaug" type="Pctestaug"></xsd:element>
</xsd:schema>
```

3. In `xsd:extension base="ns0:FulfillLineEff"`, verify that `xsd:element name=` contains pctestaugseg1.

   This value must match the value you included earlier in your import payload.

   If you prefer not to use SoapUI to verify the flexfield segment, then as an alternative, run a SQL query against the Oracle database.

```
select
 fdst.DESCRIPTIVE_FLEXFIELD_CODE,
 fdst.CONTEXT_CODE,
 fdst.SEGMENT_CODE,
 fdst.NAME,
 fdsb.SEGMENT_IDENTIFIER,
 fdsb.COLUMN_NAME,
 fdsb.SEQUENCE_NUMBER
from
 fnd_df_segments_tl fdst,
 fnd_df_segments_b fdsb
where
 fdst.APPLICATION_ID = fdsb.APPLICATION_ID and
 fdst.ENTERPRISE_ID = fdsb.ENTERPRISE_ID and
 fdst.DESCRIPTIVE_FLEXFIELD_CODE = fdsb.DESCRIPTIVE_FLEXFIELD_CODE and
 fdst.CONTEXT_CODE = fdsb.CONTEXT_CODE and
 fdst.SEGMENT_CODE = fdst.SEGMENT_CODE and
 fdst.language = 'US' and
 fdst.descriptive_flexfield_code = 'DOO_FULFILL_LINES_ADD_INFO'
order by
 fdst.CONTEXT_CODE,
 fdsb.SEQUENCE_NUMBER;
```

   Here's an example of data the query might return.

DESCRIPTIVE_ FLEXFIELD_CODE	CONTEXT_CODE	SEGMENT_CODE	NAME	SEGMENT_ IDENTIFIER	COLUMN_NAME
DOO_FULFILL_ LINES_ADD_INFO	Accounting_Rule	PC 1	pc1	PC 1	ATTRIBUTE_CHAR1
DOO_FULFILL_ LINES_ADD_INFO	LinePrcOverride	SalePrcOverrideVal	SalePrcOverrideVal	saleprcoverrideval	ATTRIBUTE_ NUMBER1
DOO_FULFILL_ LINES_ADD_INFO	PCTESTAUG	PCTESTAUGSEG1	PCTESTAUGSEG1	pctestaugseg1	ATTRIBUTE_CHAR7

DESCRIPTIVE_ FLEXFIELD_CODE	CONTEXT_CODE	SEGMENT_CODE	NAME	SEGMENT_ IDENTIFIER	COLUMN_NAME

The pctestaugseg1 value in the SEGMENT_IDENTIFIER column must match the value in `<ns22:pctestaugseg1>` from your import payload.

Related Topics

- Use SQL to Query Order Management Data
- Example Web Service Payloads That Integrate Order Management
- Guidelines for Setting Up Extensible Flexfields in Order Management
- Guidelines for Using Web Services to Integrate Order Management

Another Example of Importing Flexfield Data Through Web Services

Assume you add an extensible flexfield that you use to store a status on the order line, and you plan to import it through a web service.

1. Add the SourceLineInfo context and Status segment to the Fulfillment Line Information extensible flexfield.
   For details, see *Set Up Extensible Flexfields in Order Management*.
2. On the Manage Order Extensible Flexfields page, search for the value.

Attribute	Value
Name	Fulfillment Line Information

3. In the search results, click the **row** that contains Fulfillment Line Information in the Name column, then click **Actions > Download Flexfield Archive**.
4. In the Confirmation dialog, click **Download**, then save the zip file to any folder.
   The file manager for your operating system opens. For example, if you're using Microsoft Windows, then Windows Explorer opens.
5. In your file manager, double-click **10008_DOO_FULFILL_LINES_ADD_INFO.zip**, then expand it to the view folder.
   For example:
   ```
   C:\downloads\10008_DOO_FULFILL_LINES_ADD_INFO.zip\oracle\apps\scm\doo\processOrder\flex
   \fulfillLineContextsB\view
   ```
   The FulfillLineEffBSourceLineInfoprivateVO.xml file contains the extensible flexfield segment that you must reference.
6. Use an xml editor to open FulfillLineEffBSourceLineInfoprivateVO.xml.
7. Identify the values that you will need in your web service payload.
   - Search for the attribute name.

```
<ViewAttribute Name="status" EntityUsage="FulfillLineEffEO" EntityAttrName="status"
 AliasName="status">
```

- Search for the context code.

```
<Property Name="FND_ACFF_EFF_CONTEXT_CODE" Value="SourceLineInfo"/>
```

8. Create your web service payload.

```
<mod:AdditionalFulfillmentLineInformationCategories
 xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineCategories/"
 xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/"
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <ns8:SourceTransactionLineIdentifier>1</ns8:SourceTransactionLineIdentifier>
 <ns8:SourceTransactionScheduleIdentifier>1</ns8:SourceTransactionScheduleIdentifier>
 <ns8:SourceTransactionLineNumber>1</ns8:SourceTransactionLineNumber>
 <ns8:SourceTransactionScheduleNumber>1</ns8:SourceTransactionScheduleNumber>
 <ns12:FulfillLineEffBSourceLineInfoprivateVO>
 <ns8:ContextCode>SourceLineInfo</ns8:ContextCode>
 <ns22:status>Working</ns22:status>
 </ns12:FulfillLineEffBSourceLineInfoprivateVO>
 </mod:AdditionalFulfillmentLineInformationCategories>
```

where

- `ContextCode` contains the value you found for the context code in FulfillLineEffBSourceLineInfoprivateVO.xml.
- `status` contains the value you found for the attribute in FulfillLineEffBSourceLineInfoprivateVO.xml

Here's your WSDL.

```
https://
server:port/soa-infra/services/default/DooDecompReceiveOrderExternalComposite/ReceiveOrderRequestService
```

We recommend that you use Order Import Service instead of Receive Order Request Service. For details, see *Web Services That You Can Use to Integrate Order Management*.

Use the ProcessOrderRequest operation. You can also use the ProcessOrderRequestSync operation to get a response in SOAP UI for testing purposes.

As an alternative to examining the zip file, you can get the attribute name and context code in the output file when you run the *Publish Extensible Flexfield Attributes* scheduled process. For details, see *Set Up Extensible Flexfields in Order Management*.

Import a Flexfield on the Order Header

The procedure that you use to Import a flexfield that's on the order header is similar, with a few differences.

Assume you need a flexfield that your users can use to store a value that identifies the bill-to customer.

- Edit the Header Information extensible flexfield instead of Fulfillment Line Information.
- Add a context named PMC Header and segment named customerBillToId to the Header Information extensible flexfield.
- Download and open 10008_DOO_HEADERS_ADD_INFO.zip.
- Navigate to `oracle/apps/scm/doo/processOrder/flex/headerContextsB/view` in the zip file.

- Open the HeaderEffBPMC__HeaderprivateVO.xml file.
- Identify the context code. Search for the value.

```
<Property Name="FND_ACFF_EFF_CONTEXT_CODE" Value="PMC Header"/>
```

For example:

```
<mod:AdditionalHeaderInformationCategories
 xsi:type="ns12:j_HeaderEffDooHeadersAddInfoprivate"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ns3="http://xmlns.oracle.com/apps/scm/doo/processOrder/service/"
 xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerCategories/"
 xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerContextsB/"
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_HEADERS_ADD_INFO</ns8:Category>
 <ns12:HeaderEffBPMC__HeaderprivateVO>
 <ns8:ContextCode>PMC Header</ns8:ContextCode>
 <ns22:specialHandlingInstructions>West Coast Packers</ns22:specialHandlingInstructions>
 </ns12:HeaderEffBPMC__HeaderprivateVO>
 </mod:AdditionalHeaderInformationCategories>
```

Here's the entire payload.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:dood="http://
xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/receiveSalesOrder/
DooDecompReceiveOrderExternalComposite" xmlns:mod="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/model/" xmlns:mod1="http://xmlns.oracle.com/apps/scm/doo/
processOrder/model/" xmlns:xsi="xsi">
 <soapenv:Header/>
 <soapenv:Body>
 <dood:process>
 <dood:OrchestrationOrderRequest>
 <mod:SourceTransactionNumber>PMC-160902-016</mod:SourceTransactionNumber>
 <mod:SourceTransactionSystem>OPS</mod:SourceTransactionSystem>
 <mod:SourceTransactionIdentifier>PMC-160902-016</mod:SourceTransactionIdentifier>
 <mod:FreezePriceFlag>false</mod:FreezePriceFlag>
 <mod:FreezeShippingChargeFlag>false</mod:FreezeShippingChargeFlag>
 <mod:FreezeTaxFlag>false</mod:FreezeTaxFlag>
 <mod:BuyingPartyId>300000074725229</mod:BuyingPartyId>
 <mod:BuyingPartyContactName/>
 <mod:ShipToPartyType>ORGANIZATION</mod:ShipToPartyType>
 <mod:ShipToPartyIdentifier>300000074725229</mod:ShipToPartyIdentifier>
 <mod:ShipToPartySiteIdentifier>300000074725231</mod:ShipToPartySiteIdentifier>
 <mod:BillToPartyType>ORGANIZATION</mod:BillToPartyType>
 <mod:BillToCustomerIdentifier>300000075240955</mod:BillToCustomerIdentifier>
 <mod:BillToAccountSiteUseIdentifier>300000075240957</mod:BillToAccountSiteUseIdentifier>
 <mod:TransactionalCurrencyCode>USD</mod:TransactionalCurrencyCode>
 <mod:TransactionOn>2020-04-07T10:10:10</mod:TransactionOn>
 <mod:RequestingBusinessUnitIdentifier>300000001616323</mod:RequestingBusinessUnitIdentifier>
 <mod:OrigSystemDocumentReference/>
 <mod:AdditionalHeaderInformationCategories xsi:type="ns12:j_HeaderEffDooHeadersAddInfoprivate"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns3="http://xmlns.oracle.com/apps/
scm/doo/processOrder/service/" xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
headerCategories/" xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/headerContextsB/"
 xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_HEADERS_ADD_INFO</ns8:Category>
 <ns12:HeaderEffBPMC__HeaderprivateVO>
 <ns8:ContextCode>PMC Header</ns8:ContextCode>
 <ns22:specialHandlingInstructions>West Coast Packers</ns22:specialHandlingInstructions>
 </ns12:HeaderEffBPMC__HeaderprivateVO>
 </mod:AdditionalHeaderInformationCategories>
 <mod:OrchestrationOrderRequestLine>
```

```
 <mod:SourceTransactionLineIdentifier>1</mod:SourceTransactionLineIdentifier>
 <mod:SourceTransactionScheduleIdentifier>1</mod:SourceTransactionScheduleIdentifier>
 <mod:SourceTransactionLineNumber>1</mod:SourceTransactionLineNumber>
 <mod:SourceTransactionScheduleNumber>1</mod:SourceTransactionScheduleNumber>
 <mod:ProductNumber>PMC - Std Item</mod:ProductNumber>
 <mod:OrderedQuantity>10</mod:OrderedQuantity>
 <mod:OrderedUOM>Each</mod:OrderedUOM>
 <mod:PaymentTerms>NG_IMMEDIATE</mod:PaymentTerms>
 <mod:InventoryOrganizationIdentifier>300000001621747</mod:InventoryOrganizationIdentifier>
 <mod:ShipToPartyIdentifier>300000074725229</mod:ShipToPartyIdentifier>
 <mod:ShipToPartySiteIdentifier>300000074725231</mod:ShipToPartySiteIdentifier>
 <mod:BillToPartyType>ORGANIZATION</mod:BillToPartyType>
 <mod:BillToCustomerIdentifier>300000075240955</mod:BillToCustomerIdentifier>
 <mod:BillToAccountSiteUseIdentifier>300000075240957</mod:BillToAccountSiteUseIdentifier>
 <mod:ShippingInstructions>Handle with care</mod:ShippingInstructions>
 <mod:PackingInstructions>Pack with shockproof material</mod:PackingInstructions>
 <mod:RequestedShipDate>2020-04-07T10:10:10</mod:RequestedShipDate>
 <mod:PaymentTerms>NG_IMMEDIATE</mod:PaymentTerms>
 <mod:TransactionCategoryCode>ORDER</mod:TransactionCategoryCode>
 <mod:PartialShipAllowedFlag>false</mod:PartialShipAllowedFlag>
 <mod:OrigSysDocumentReference>ORIGSYS</mod:OrigSysDocumentReference>
 <mod:OrigSysDocumentLineReference>ORIGSYSLINE</mod:OrigSysDocumentLineReference>
 <mod:AdditionalFulfillmentLineInformationCategories
 xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineCategories/" xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/
fulfillLineContextsB/" xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/">
 <ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <ns8:SourceTransactionLineIdentifier>1</ns8:SourceTransactionLineIdentifier>
 <ns8:SourceTransactionScheduleIdentifier>1</ns8:SourceTransactionScheduleIdentifier>
 <ns8:SourceTransactionLineNumber>1</ns8:SourceTransactionLineNumber>
 <ns8:SourceTransactionScheduleNumber>1</ns8:SourceTransactionScheduleNumber>
 <ns12:FulfillLineEffBSourceLineInfoprivateVO>
 <ns8:ContextCode>SourceLineInfo</ns8:ContextCode>
 <ns22:status>Working</ns22:status>
 </ns12:FulfillLineEffBSourceLineInfoprivateVO>
 </mod:AdditionalFulfillmentLineInformationCategories>
 </mod:OrchestrationOrderRequestLine>
 <mod:OrderProcessingPreferences>
 <mod:CreateCustomerInformationFlag>true</mod:CreateCustomerInformationFlag>
 <mod:SubmitFlag>true</mod:SubmitFlag>
 </mod:OrderProcessingPreferences>
 </dood:OrchestrationOrderRequest>
 </dood:process>
 </soapenv:Body>
</soapenv:Envelope>
```

Related Topics

- Use SQL to Query Order Management Data
- Example Web Service Payloads That Integrate Order Management
- Guidelines for Setting Up Extensible Flexfields in Order Management
- Guidelines for Using Web Services to Integrate Order Management
- Set Up Extensible Flexfields in Order Management

Update Extensible Flexfields During Fulfillment

Call the FulfillmentResponse web service to update an extensible flexfield on a sales order after your user already submitted the sales order to fulfillment.

Make the call only when the orchestration process is on a wait step.

Here's an example payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns1:FulfillmentRequest
xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/taskLayer/fulfillOrder/
DooTaskFulfillOrderResponseInterfaceComposite">
 <ns1:FLine
xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/common/process/model/">
 <!-- Mandatory attributes 3 below, make sure you provide the correct value-->
 <ns2:FulfillLineId>300100095720462</ns2:FulfillLineId>
<ns2:SourceOrderSystem>LEG</ns2:SourceOrderSystem>
<ns2:TaskType>FulfillOrder</ns2:TaskType>
 <!-- Optional attributes -->
 <ns2:OrderedQuantity unitCode="">11</ns2:OrderedQuantity>
 <ns2:RecordNumber>1</ns2:RecordNumber>
 <!-- Mandatory attributes assuming you want to update EFFs -->
 <ns2:AdditionalFulfillLineInformationCategories
xsi:type="ns12:j_FulfillLineEffDooFulfillLinesAddInfoprivate"
xmlns:ns12="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineCategories/"
xmlns:ns22="http://xmlns.oracle.com/apps/scm/doo/processOrder/flex/fulfillLineContextsB/"
xmlns:ns8="http://xmlns.oracle.com/apps/scm/doo/processOrder/model/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ns8:Category>DOO_FULFILL_LINES_ADD_INFO</ns8:Category>
 <!-- Nodes below repeats for each of the context that you want to update -->
 <!-- The node itself contains the name of the virtual object for the context-->
 <!-- To get these details, run the Publish Extensible Flexfields scheduled process-->
 <!-- This example assumes you are updating two contexts -->
 <ns12:FulfillLineEffBPackShipInstructionprivateVO>
<ns8:ContextCode>PackShipInstruction</ns8:ContextCode>
<ns22:_PackingInstruction>NKResponsePack</ns22:_PackingInstruction>
<ns22:_ShippingInstruction>NKResponseShip</ns22:_ShippingInstruction>
<ns22:_ShippingCost>31</ns22:_ShippingCost>
<ns22:_NeedbyDate>2016-04-21</ns22:_NeedbyDate>
<ns22:_PickDate>2016-04-21T12:12:12</ns22:_PickDate>
</ns12:FulfillLineEffBPackShipInstructionprivateVO>
<ns12:FulfillLineEffBFulfillLineContext1privateVO>
<ns8:ContextCode>FulfillLineContext1</ns8:ContextCode>
<ns22:_FL1AttributeChar1>FLC1-Resp</ns22:_FL1AttributeChar1>
<ns22:_FL1AttributeChar2>FLC2-Resp</ns22:_FL1AttributeChar2>
<ns22:_FL1AttributeNum1>620</ns22:_FL1AttributeNum1>
<ns22:_FL1AttributeDate1>2016-03-13</ns22:_FL1AttributeDate1>
<ns22:_FL1AttributeDateTime1>2016-03-13T12:12:12</ns22:_FL1AttributeDateTime1>
</ns12:FulfillLineEffBFulfillLineContext1privateVO>
</ns2:AdditionalFulfillLineInformationCategories>
 </ns1:FLine>
 </ns1:FulfillmentRequest>
 </soap:Body>
</soap:Envelope>
```

where

- FulfillLineId identifies your fulfillment line, such as 300100095720462.
- SourceOrderSystem identifies your source system, such as LEG.
- Replace values for other attributes, as necessary, such as PackingInstruction, dates, times, and so on.

Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management
- Guidelines for Integrating Order Management

Use Order Management Extensions to Get Values from Extensible Flexfields

Use the getOrCreateContextRow method to add new data or update data through a flexfield. Use getContextRow to get data that you already added.

Use `getOrCreateContextRow("context code")` or `getAttribute("API name")`, where `context code` is the name of the context code of the extensible flexfield and `API name` is the name of the API (application programming interface) that you use to call the method:

Entity	Value to Use in Code	Parent	Read During All Events	Write During Save or Start of Submission Request	Write During End of Submission Request
Extensible flexfield on order header	Not applicable	Header	Yes	Yes	Yes
Extensible flexfield on fulfillment line	Not applicable	Lines	Yes	Yes	Yes

If you use an extension to get a value from an extensible flexfield, then the extension behaves differently depending on the method that you use and whether the context that you specify in the extension exists.

Method	If the Context Doesn't Exist
getContextRow("*EFFContextName*")	This method doesn't create a new context.
getorCreateContextRow("*EFFContextName*"	This method creates a new context.

Assume you use this code.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");
if (poNumber != null && poNumber.equals("test")) {
 def lines = header.getAttribute("Lines");
 while (lines.hasNext()) {
 def line = lines.next();
 def context = line.getOrCreateContextRow("My_Context");
 def effVal = context.getAttribute("eligibleforprime");
 // throw new ValidationException("Eff value effVal:: "+effVal);
 if (effVal.equals("Y")) {
 line.setAttribute("OrderedQuantity", 0);
 }
 }
}
```

where

- `getOrCreateContextRow("My_Context")` attempts to find a context named My_Context. If that context doesn't exist, then the extension creates it.

You use getOrCreateContextRow to add new data or to update data through the flexfield on your sales order.

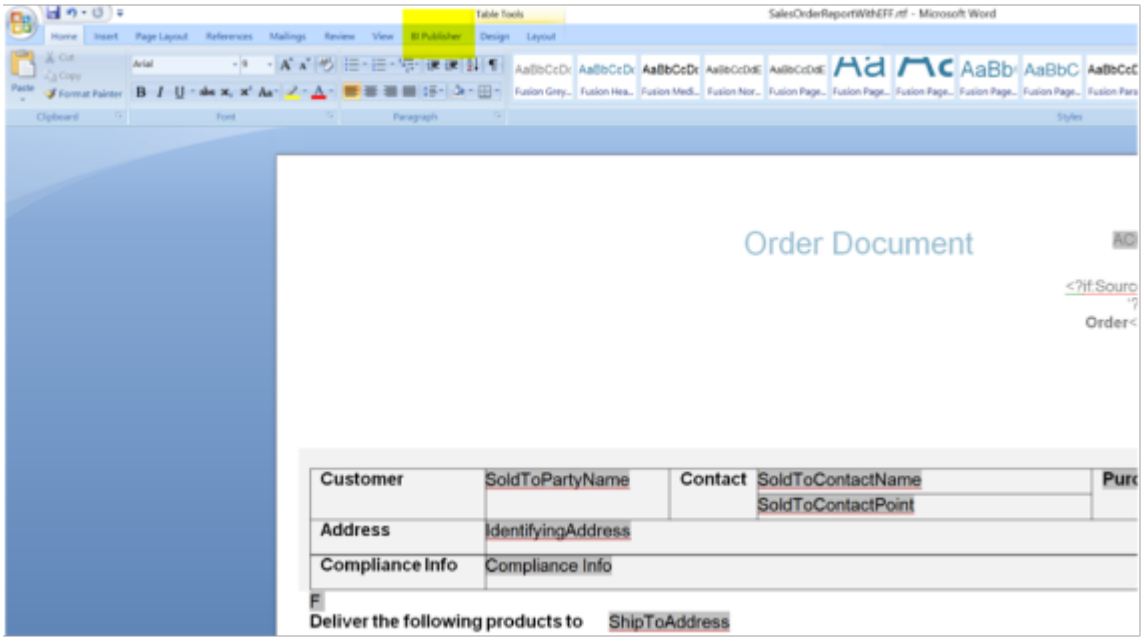
If you only need to get data that you already added through the flexfield on the sales order, then use getContextRow instead of getOrCreateContextRow. For example:

```
import oracle.apps.scm.doo.common.extensions.ValidationException;
def poNumber = header.getAttribute("CustomerPONumber");
if (poNumber != null && poNumber.equals("test")) {
 def lines = header.getAttribute("Lines");
 while (lines.hasNext()) {
 def line = lines.next();
 def context = line.getContextRow("My_Context");
 if (context == null) {
 return; // if we don't find a context, then don't create one.
 }
 def effVal = context.getAttribute("eligibleforprime");
 // throw new ValidationException("Eff value effVal:: "+effVal);
 if (effVal.equals("Y")) {
 line.setAttribute("OrderedQuantity", 0);
 }
 }
}
```

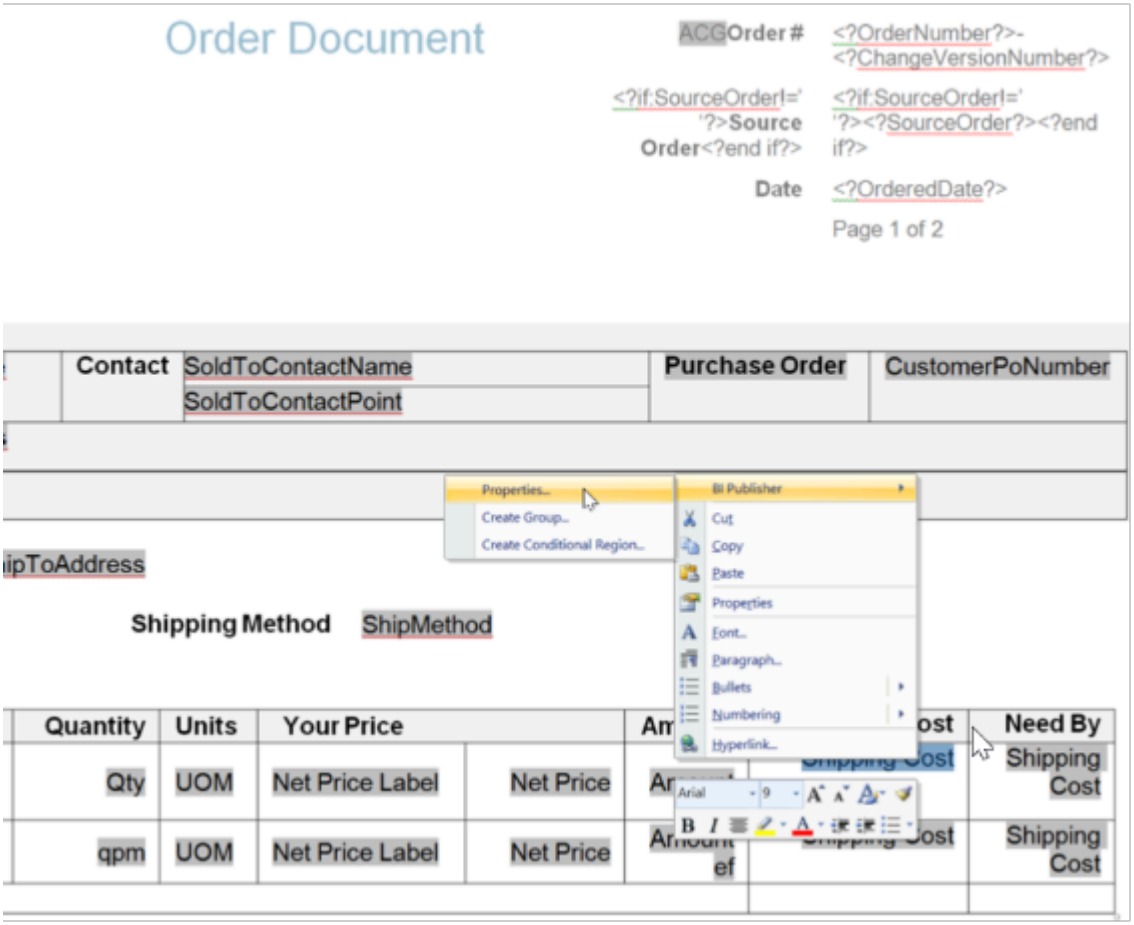
Use Rich Text Files to Print Extensible Flexfield Data in Order Headers

Use an RTF file (Rich Text File) to print extensible flexfield data in the order header.

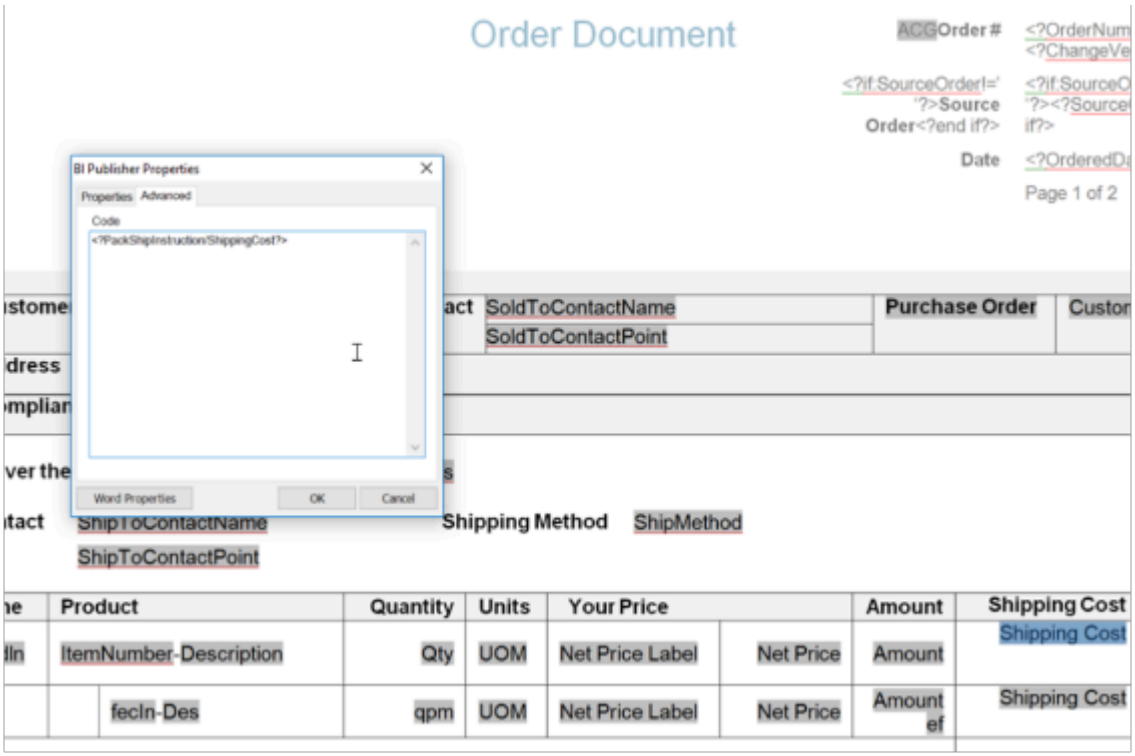
1. Download, then install the Oracle Analytics Publisher for Microsoft Word plugin so you can edit the RTF layout. Select a version that's compatible with your version of Microsoft, such as Oracle Analytics Publisher 11.1.1.9.0 for 32 bit Office on Windows.
2. Notice the Oracle Analytics Publisher option in the menu bar.



3. Set the Oracle Analytics Publisher properties in the RTF template. See the Oracle Analytics Publisher documentation for details.

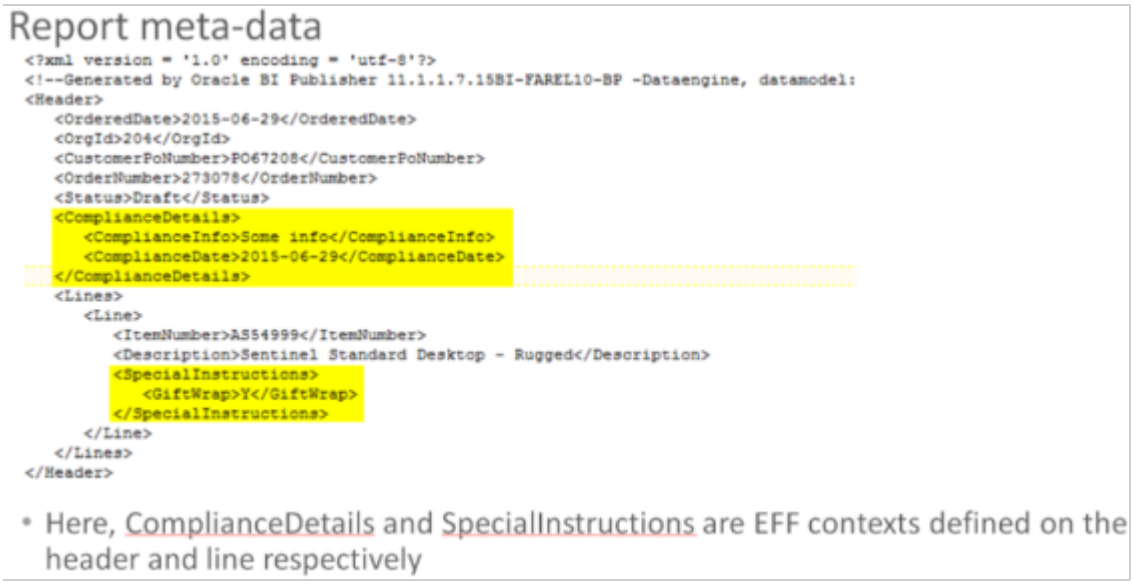


4. Access the line extensible flexfield.



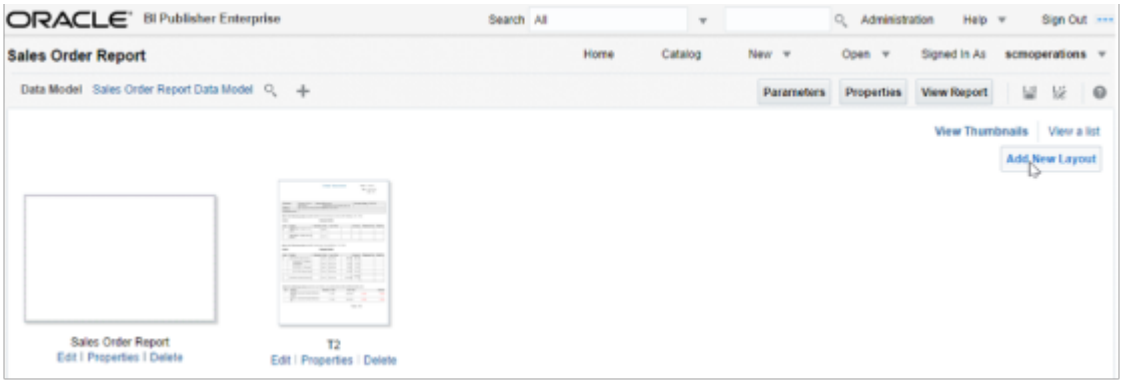
To view an example `rtf` file, click *here*, then open sales_order_report_with_extensible_flexfield.rtf.

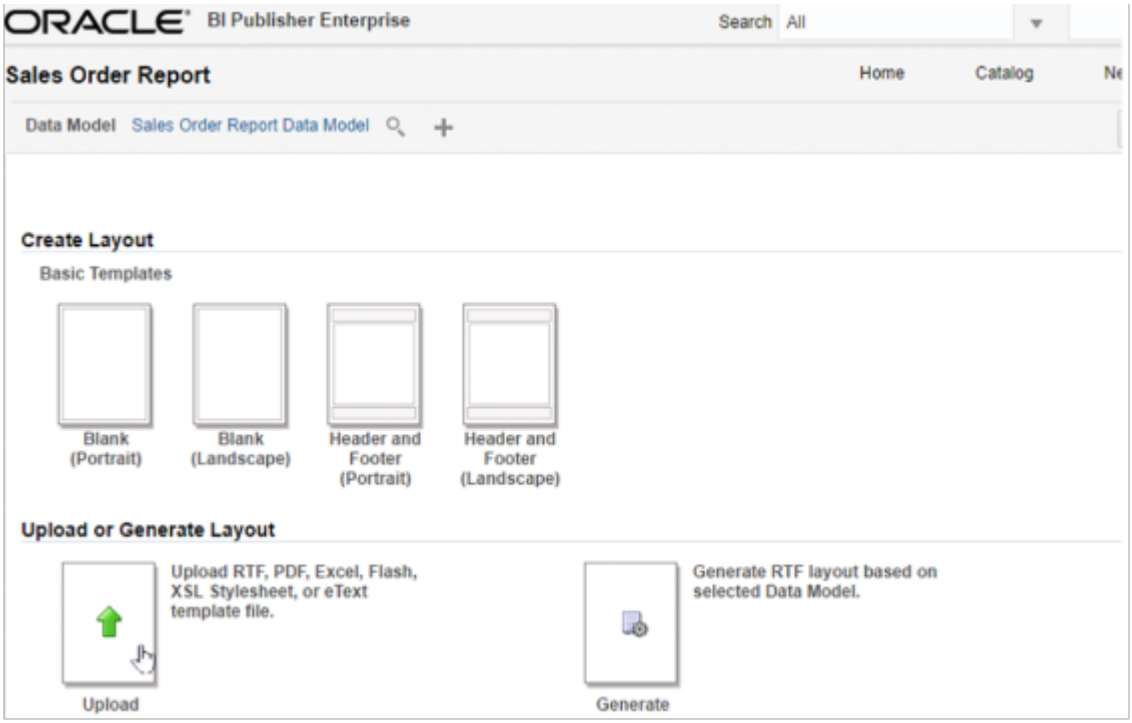
5. Examine the example XML payload that represents a sales order that you send to Oracle Analytics Publisher to create the report.

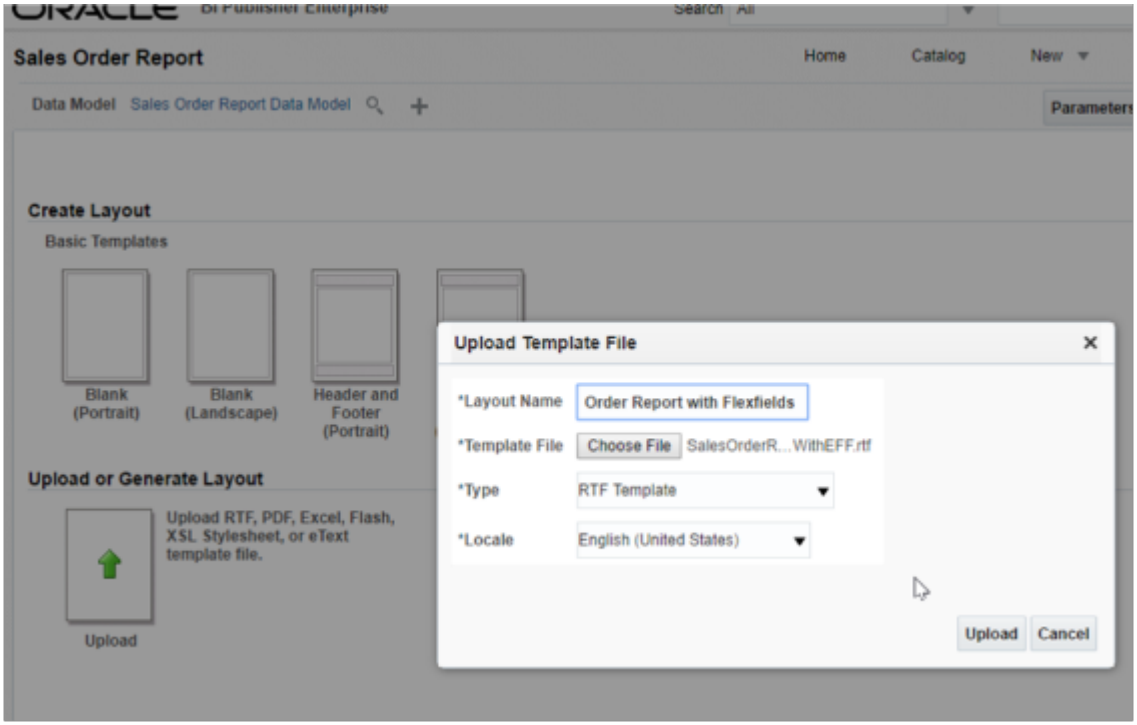


6. Create a new template.
7. Edit the Sales Order Report.
8. Upload the new template to Sales Order Report.
9. Notice that the new template is now available in the Template menu on the view document dialog.

Set the new template as the default template in Oracle Analytics Publisher.







Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management

More Set Up Details for Extensible Flexfields

Get more details about how to create extensible flexfields in Order Management.

Set Up Extensible Flexfields When You Use Oracle Analytics Publisher

If you use Oracle Analytics Publisher, then make sure you do these steps when you set up your extensible flexfield. For assistance, contact Oracle Support.

1. Run the *Publish Extensible Flexfield Attributes* scheduled process.
2. On the Manage Extensible Flexfield page, click **Actions > Refresh and Deploy Offline**.
3. Deploy your extensible flexfield. For details, see *Overview of Using Extensible Flexfields in Order Management*.
4. Download the flexfield archive.
    - On the Manage Extensible Flexfield page, click **Actions > Download Flexfield Archive**.
    - Wait for the dialog to display 100%, then click **Download**.
    - In the Opening dialog, click **OK**.
    - Save the file to a folder of your choice.
    - Unzip the file.
5. Reset the Financial Analytics container.

Extensible Flexfields That You Can Publish

You can publish only the extensible flexfield attributes that come predefined with Order Management and that are part of the predefined extensible categories.

Entity	Predefined Category
Headers	DOO_HEADERS_ADD_INFO
Lines	DOO_LINES_ADD_INFO
FulfillLines	DOO_FULFILL_LINES_ADD_INFO
Payments	DOO_PAYMENTS_ADD_INFO
PriceAdjustments	DOO_PRICE_ADJUSTMENTS_ADD_INFO
SalesCredits	DOO_SALES_CREDITS_ADD_INFO
OrchestrationTaskActivity	DOO_ACTIVITIES_ADD_INFO
FulfillLineDetails	DOO_FULFILL_LINE_DTLS_ADD_INFO
LotSerialNumber	DOO_LOT_SERIAL_NUM_ADD_INFO

The Category Hierarchy and Preconfigured Context Values aren't available for these extensible flexfields.

Masking Data

Assume you need to display your own postal code on the order header, the postal code has seven digits, and you need to use an extensible flexfield to do it because these postal codes aren't part of Order Management's predefined data. The data contains sensitive information, so you also want to use asterisks to mask the value except for the last two characters. For example, the user enters 5674577, and you want to mask that value to *****77.

Note that you can't mask data in a value set in Order Management, but we can still meet the rest of your requirement. Here's how.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage SCM Common Value Sets
2. On the Manage SCM Common Value Sets page, click **Actions > Create**, then set the values.

Attribute	Value
Value Set Code	My Postal Codes

Attribute	Value
Module	Common
Validation Type	Format Only
Value Data Type	Character
Value Subtype	Text
Maximum Length	7

Setup varies for different types of value sets. For details, see *Set Attribute Values Before You Transform Source Orders*.

3. Create an extensible flexfield, then set the Value Set attribute on your flexfield segment to My Postal Codes. For details, see *Overview of Using Extensible Flexfields in Order Management*.

Related Topics

- Processing Constraints

Troubleshoot Extensible Flexfields in Order Management

Use this topic to help you troubleshoot problems with your extensible flexfields.

- Examine the setups for your contexts and categories. See the Create Segments and Contexts subtopic in *Guidelines for Setting Up Extensible Flexfields in Order Management*.
- Examine your value sets. Setup varies for different types of value sets. For details, see *Set Attribute Values Before You Transform Source Orders*. Also, see the Masking Data subtopic in *More Set Up Details for Extensible Flexfields*.

Deploy and Publish

Make sure you deploy and publish your extensible flexfield every time you modify the flexfield's set up:

- Verify that the Deployment Status attribute contains a check mark.
- Make sure the Publish Extensible Flexfield Attributes scheduled process finishes successfully.

For details, see the Deploy the Extensible Flexfield subtopic and the Publish the Extensible Flexfield subtopic in *Set Up Extensible Flexfields in Order Management*.

Here's an easy to make sure you have correctly deployed all of your extensible flexfields.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields

2. On the Manage Extensible Flexfields page, search for the value.

Attribute	Description
Flexfield Code	DOO%

3. In the search results, select the row that has your flexfield, then click **Actions > Deploy Flexfield**.

   You can also use **Actions > Refresh & Deploy Offline**. Refresh & Deploy Offline allows you to continue working without having to wait for the deployment to finish. For details see the Deployment section in *Considerations for Managing Extensible Flexfields*.
4. Repeat the deployment for each flexfield that you have created or modified since the last time you deployed.
5. If you continue to encounter problems during testing, log out and log back in after you deploy, and log out and log back in again after you publish.

For more tips, see *Considerations for Planning Extensible Flexfields*.

Troubleshoot Specific Problems

Trouble	Shoot
The Category area of the Edit Extensible Flexfield page doesn't have any rows, so I can't edit the category or category details, such as associated contexts and pages.	Run the *Publish Extensible Flexfield Attributes* scheduled process. For details, see *Set Up Extensible Flexfields in Order Management*.
I encounter an error. `500-Internal Server Error` This problem might happen because you updated an extensible flexfield but didn't deploy your update. It typically happens with DOO_HEADERS_ADD_INFO, DOO_LINES_ADD_INFO, or DOO_FULFILL_LINES_ADD_INFO.	Deploy your extensible flexfields. For details, see *Set Up Extensible Flexfields in Order Management*.
I click Additional Information on the order header or order line when creating a sales order in the Order Management work area, then encounter an error that's similar to: `ADF_FACES-60097:For more information, please see the server's error log for an entry beginning with: ADF_FACES-60096:Server exception during PPR` This problem happens when you include a space in the Code attribute of the extensible flexfield segment.	Remove the space from the Code attribute, deploy, then publish the extensible flexfield. For details, *Guidelines for Setting Up Extensible Flexfields in Order Management*.

Business Rules

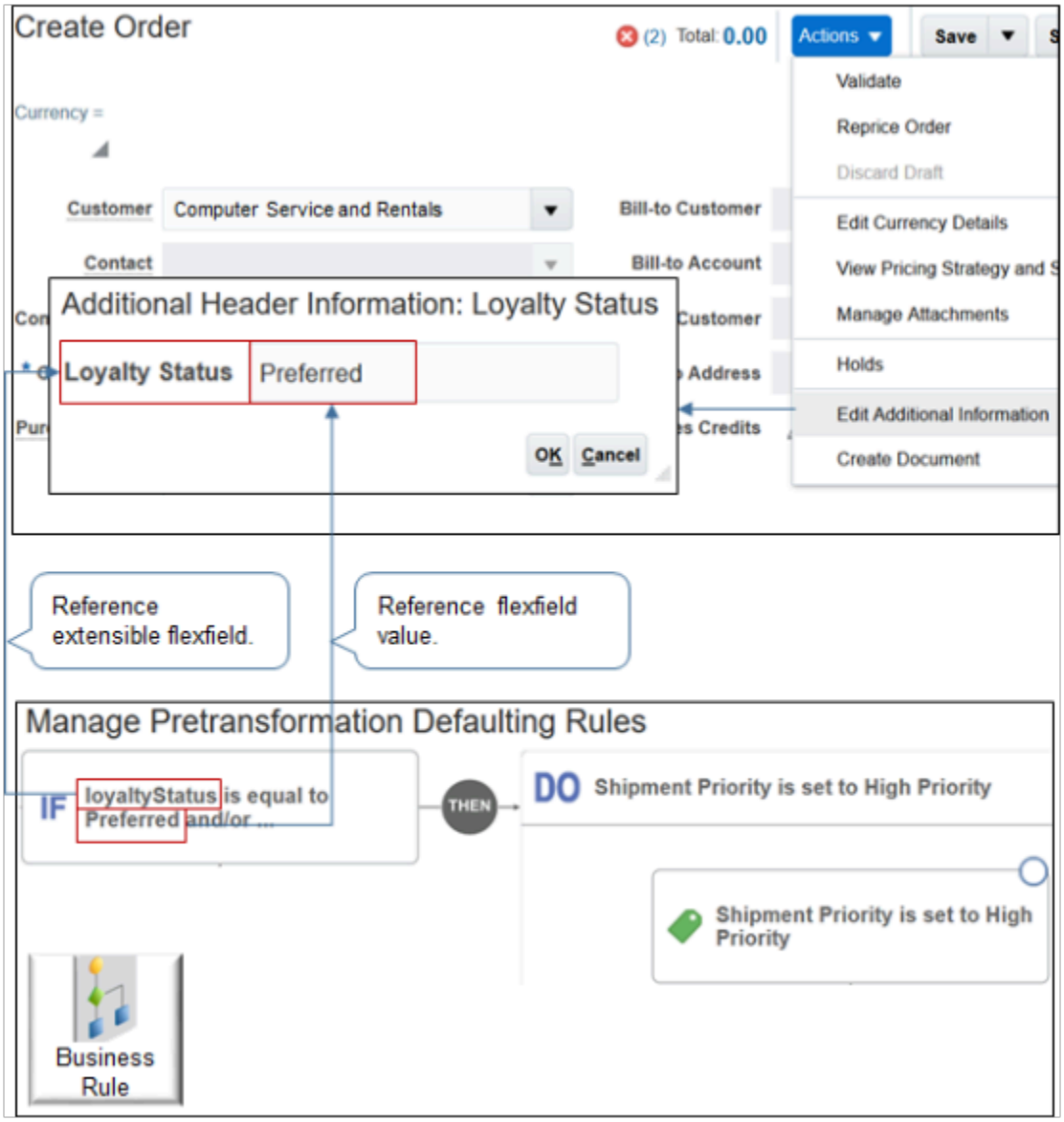
Use Extensible Flexfields in Transformation Rules

Use a transformation rule to add or change data in a fulfillment line. The rule determines information to add or change according to details that already exist in the fulfillment order.

You will create a rule.

- If Preferred customer places a sales order that includes AS54888 Desktop Computer on or before 01/01/2019 12:00 AM, then set Shipment Priority to High Priority, and add a free printer to the shipment.

Create a rule that references an extensible flexfield that stores loyalty details.



Summary of the Set Up

1. Create pretransformation rule.
2. Create product transformation rule.
3. Create posttransformation rule.

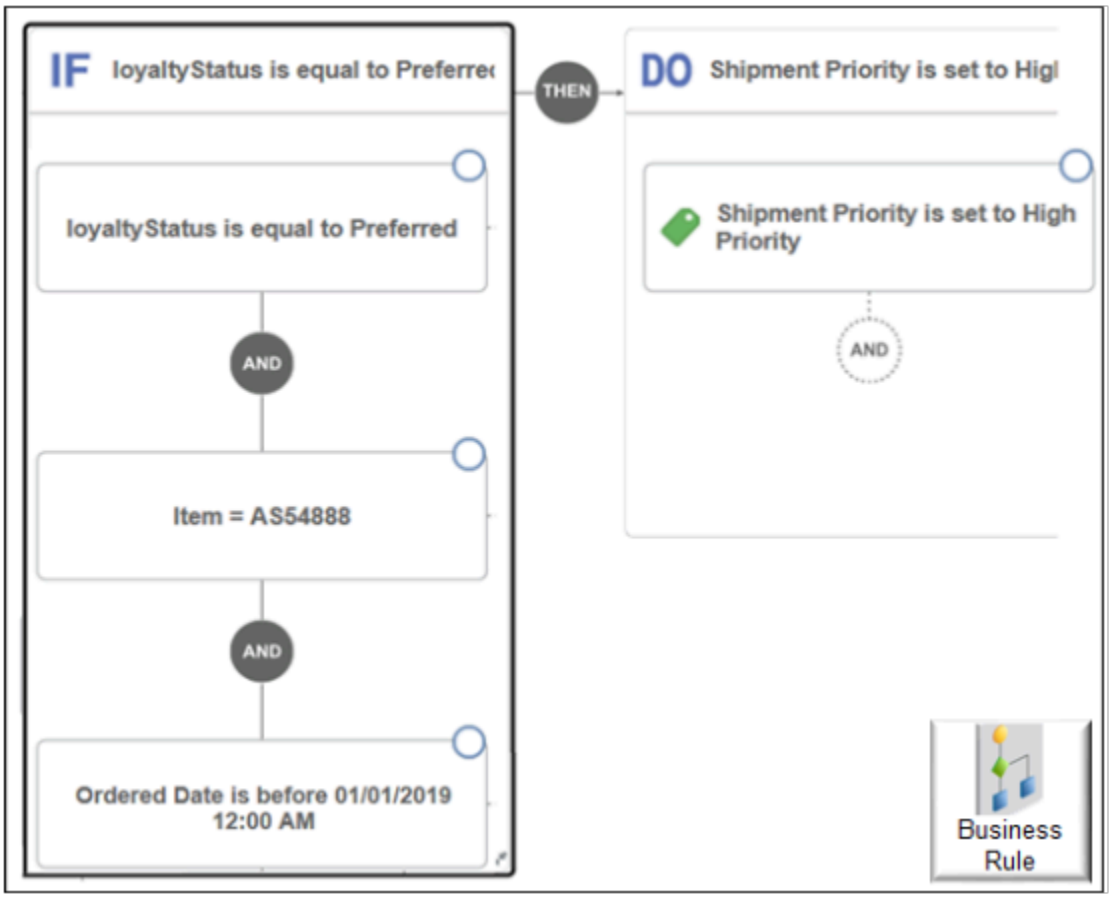
Assume you already created the Loyalty Status extensible flexfield.

For details, see *Overview of Using Business Rules With Order Management*.

This topic uses example values. You might need different values, depending on your business requirements.

Create Pretransformation Rule

Here's the rule you will create.

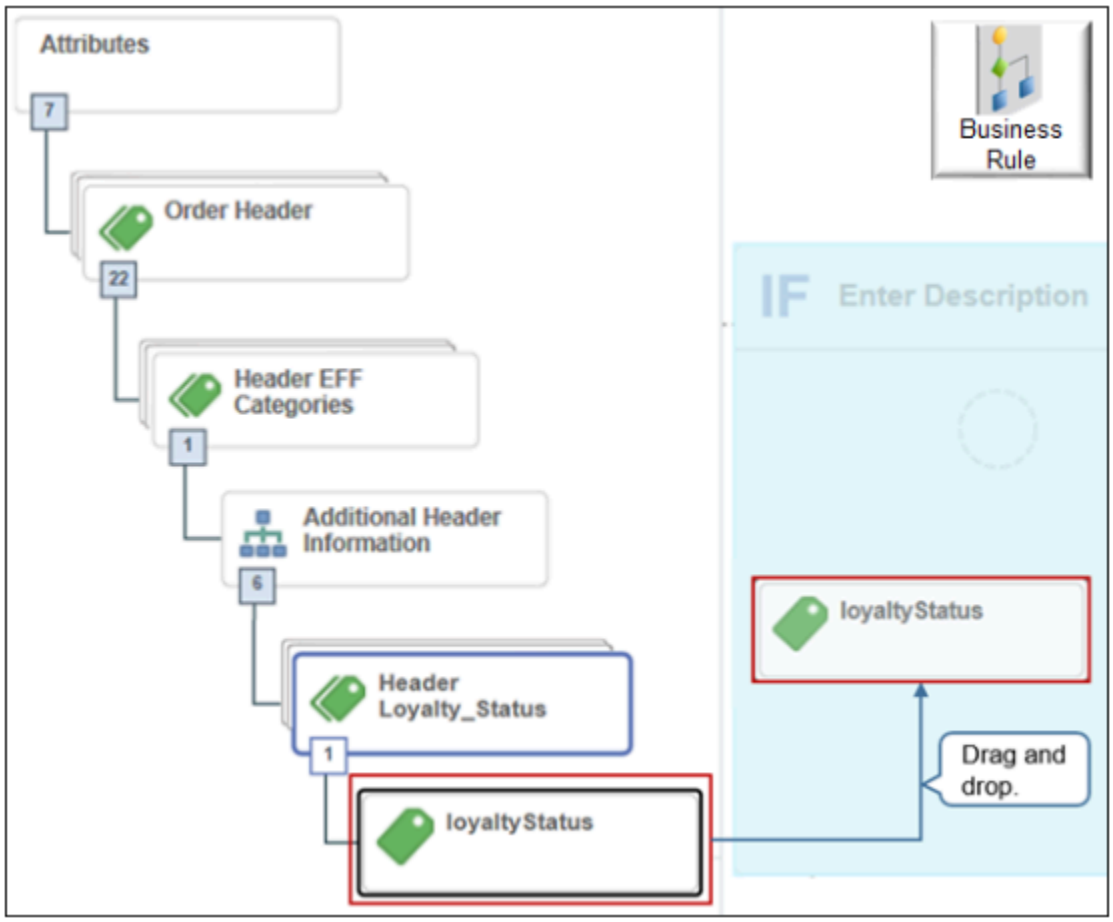


Do it.

1. Create the rule.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Pretransformation Rules for Sales Orders
    - On the Manage Pretransformation Defaulting Rules page, click **Create New Rule**.
    - Set the values.

Attribute	Value
Name	Expedite Shipping for Loyal Customer
Description	If loyal customer places a sales order that includes AS54888 Desktop Computer on or before promotion date, then add a free printer to the sales order.

2. Create the If statement.
    - Click in the **IF area** to expand it.
    - On the Attributes tab, expand **Order Header > Header EFF Categories > Additional Header Information > Header Loyalty Status**.
    - Click `Loyalty Status`, drag it, then drop it into the IF area.



    - In the Create Condition dialog, set the operator to `Is Equal To`.
    - Enter `Preferred`, then click **OK**.
3. Add the And condition for the computer.
    - Click **And**.
    - In the Create Condition dialog, enter `item`, wait a moment, then click **Item (Item Definition)**.
    - Set the operator to `=`.
    - Click Search.
    - In the Search dialog, search for AS54888, then click **OK**.

4. Add the And condition for the date.
    - Click **And**.
    - In the Create Condition dialog, enter `ordered`, wait a moment, then click **Ordered Date (Order Header)**.
    - Set the operator to `is before`.
    - Click **Select Date and Time**, set it to `01/01/2019 12:00 AM`, then click **OK > OK**.
5. Create the Do statement.
    - Click **Then > Do > New Action > Set a Value**, then click .
    - In the Create Action dialog, enter `ship`, wait a moment, then click **Shipment Priority (Order Header)**.
    - Click **Search**.
    - In the Search dialog, click **Advanced > Search**, then notice the dialog displays values you can set for this attribute.

      This functionality helps make sure you specify a value that Order Management can use and understand for the attribute. For example, Shipment Priority can contain only alphabetic data.
    - In the Meaning list, click **High Priority > OK > OK > Save and Close**.
6. Activate and publish your rule.

Create Product Transformation Rule

Use the value in the Loyalty Status extensible flexfield as criteria to make other fulfillment changes, such as add a fulfillment line for the free printer.

Use attributes to set up the object hierarchy of the sales order.

Attribute Usage	Hierarchy

Determine whether customer is preferred according to the value of the Loyalty Status extensible flexfield on the order header.	`Header`
	`Header EFF Category`
	`Header EFF Context`

Use the InventoryItemId attribute on the fulfillment line to identify the computer.	`Header`
	`Line`
	`Fulfillment Line`

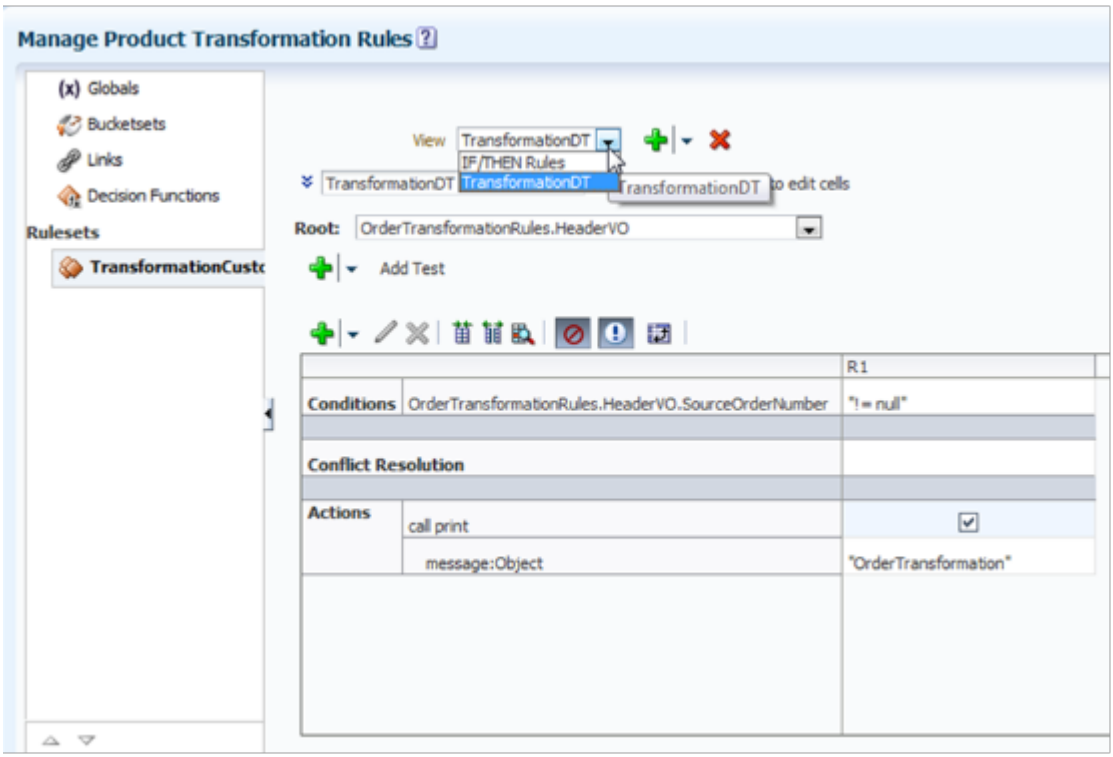
Use the OrderedDate attribute on the order header to identify the date.	`Header`
Add a new fulfillment line to the sales order for the printer.	`Header` `Fulfillment Line`

Create the Rule

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders

- Task: Manage Product Transformation Rules

2. On the Manage Product Transformation Rules page, in the View list, click **TransformationDT**.



3. To the right of View TransformationDT, click **down arrow > Expand > Add Rule > Properties**, then set the values.

Attribute	Value
Name	Add Printer for Loyal Customers
Description	If loyal customer places a sales order that includes AS54888 Desktop Computer, on or before promotion date, then add a free printer to the sales order.
Effective Date	Always
Priority	Medium
Active	Contains a check mark
Advanced Mode	Contains a check mark

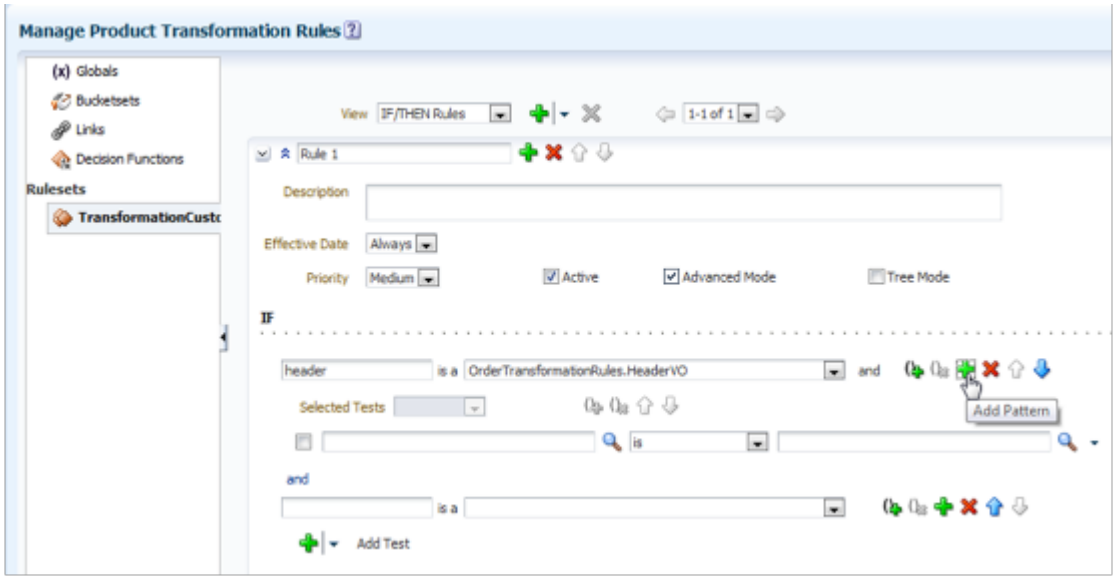
4. Click **Add Pattern** to add each pattern.

Pattern	Operand	Fact
header	is a	OrderTransformationRules.HeaderVO
line	is a	OrderTransformationRules.LineVO
fline	is a	OrderTransformationRules.FulfillLineVO
headerEFFcat	is a	OrderTransformationRules.j_HeaderEffDooHeadersAddInfoprivateVO
headerEFFCtxt	is a	OrderTransformationRules.headerEFFcat.HeaderEffBL 5FStatusprivateVO

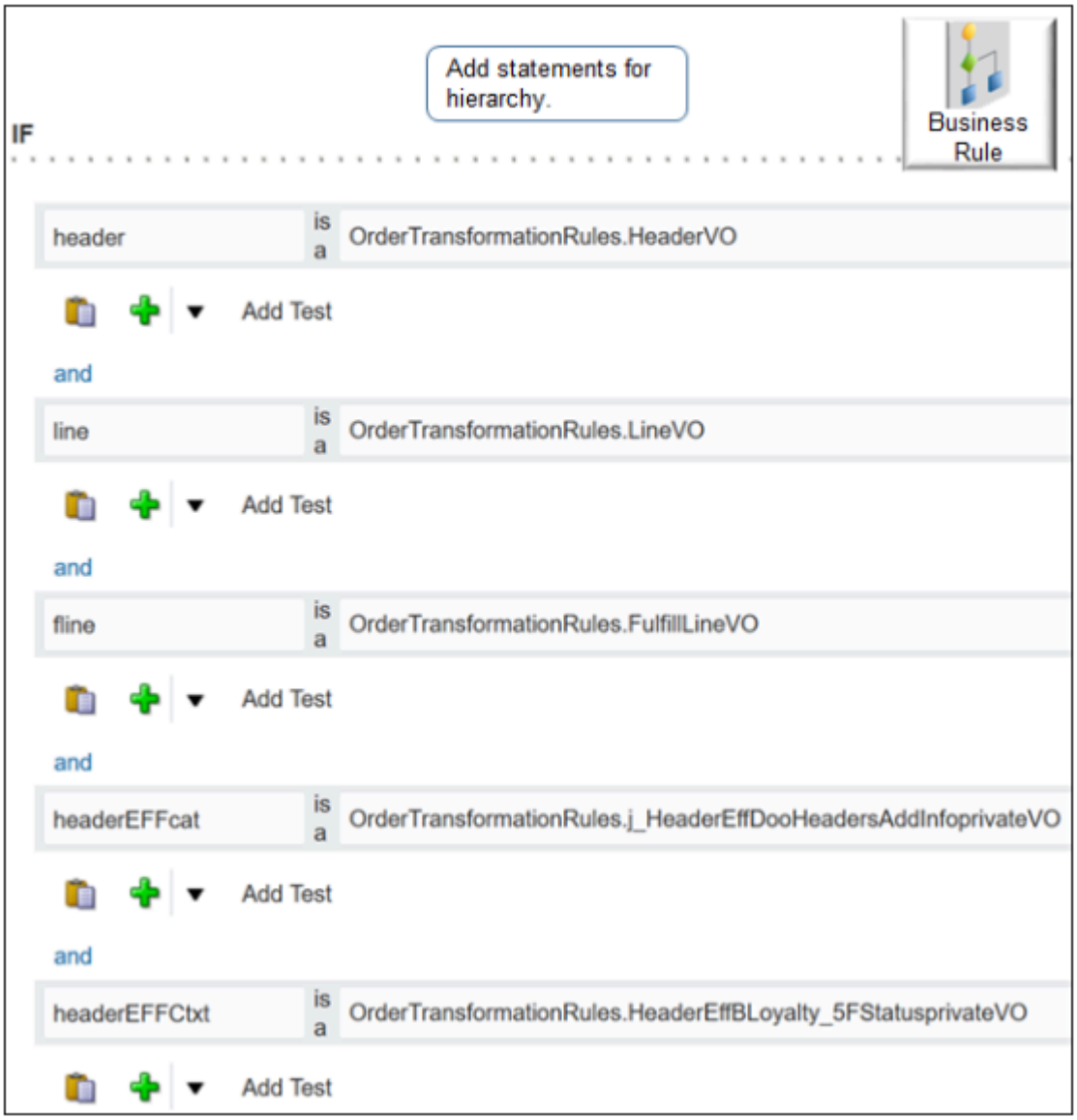
Pattern	Operand	Fact

Each pattern creates one level of the hierarchy.

For example, here's the rule that creates the first pattern, and the cursor is positioned to begin creating the second pattern.



Here's how your set up should look.



Create the Hierarchy

You will create this hierarchy.

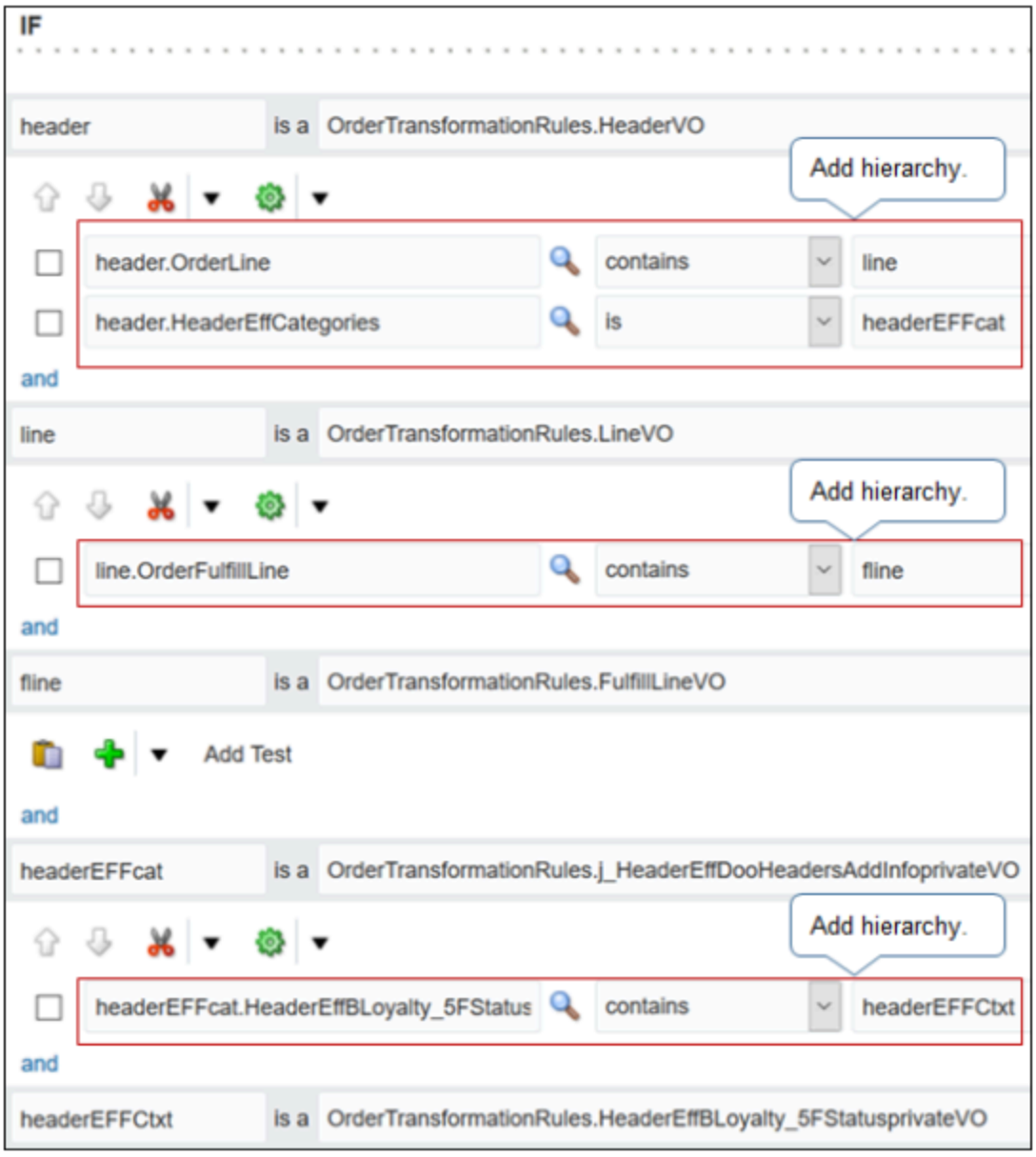
```
header
 line
 fline
 headerEFFcat
 headerEFFctxt
```

To create the hierarchy, create a test for each patterns.

Parent Pattern	Fact	Operand	Child Pattern
header	header.OrderLine	contains	line

Parent Pattern	Fact	Operand	Child Pattern
header	header.HeaderEffCategories	is	headerEFFcat
line	line.OrderFulfillLine	contains	fline
headerEFFcat	headerEFFcat.HeaderEffBLoyalty_5FStatusprivateVO	contains	headerEFFCtxt

To create a test, click **Add Test** (green icon underneath the pattern), then click **Simple Test**.



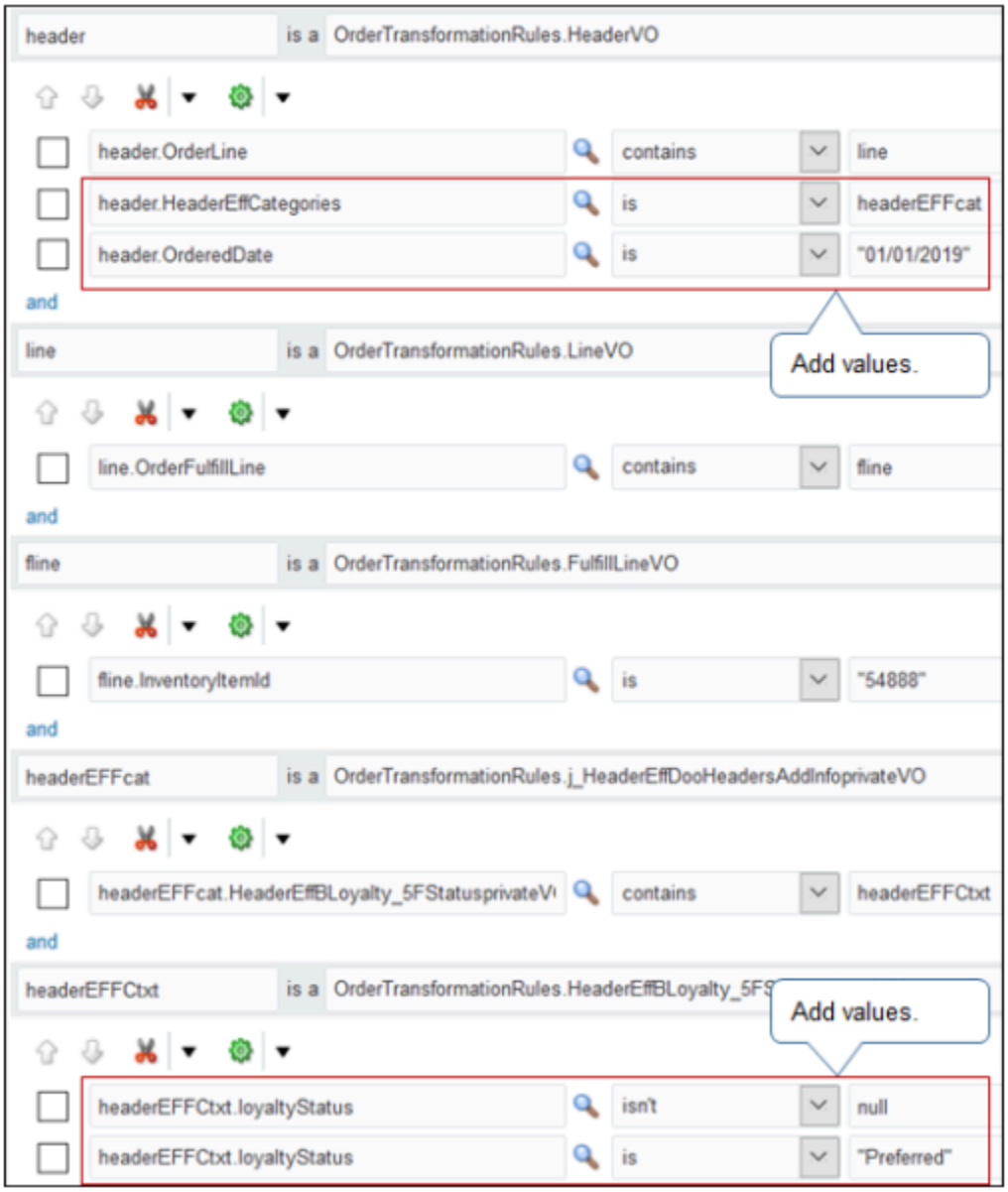
Create Test Criteria

Create the test criteria that identifies AS54888 Desktop Computer. If you include optional attributes, then make sure you include a NOT NULL check for each optional attribute to avoid the NullPointerExceptions runtime error.

Create tests.

Pattern	Fact	Operand	Value
header	header.OrderedDate	is	"01/01/2019"
fline	fline.InventoryItemId	is	"54888"
headerEFFctxt	headerEFFCtxt.loyaltyStatus	isn't	null
headerEFFctxt	headerEFFCtxt.loyaltyStatus	is	"Preferred"

Create them in the same way you created tests when you defined the hierarchy. You must include the double quotation marks (") as part of the value, where indicated. For example:



Create the Action

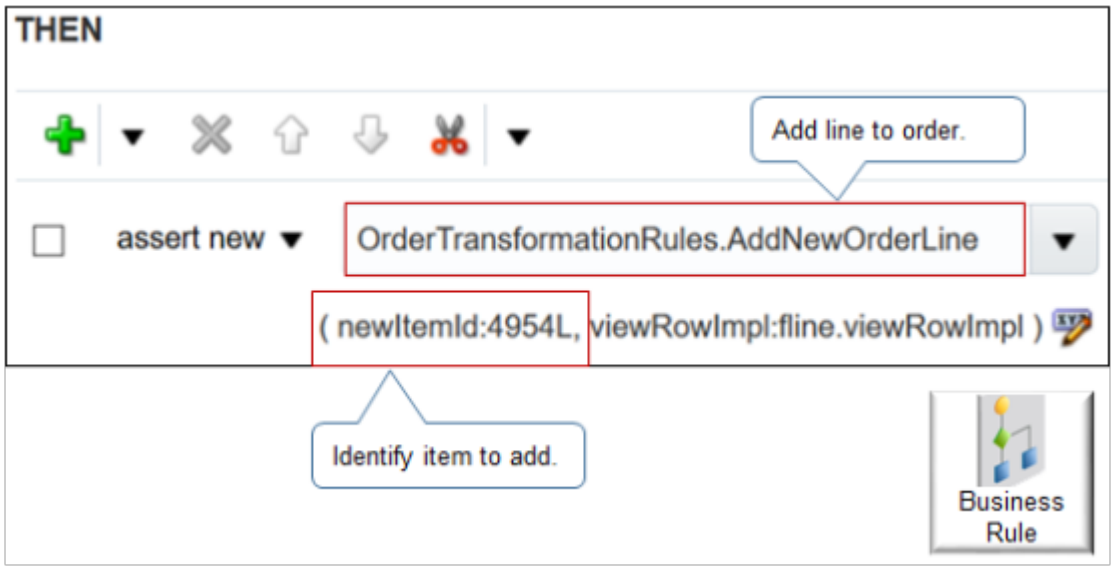
Create the action that adds a fulfillment line for the free printer.

1. In the Then area, click **Add Action > Assert New**.
2. You must add a new fulfillment line, so enter `AddNew > OrderTransformationRules.AddNewOrderLine`.
3. In the Then area, click **Edit Properties**.
4. In the Properties dialog, set the values, then click **OK**.

Name	Value
newItemId	4954L Assume 4954L identifies the printer.

Name	Value
viewRowImpl	fline.viewRowImpl

5. Make sure your set up looks like this.

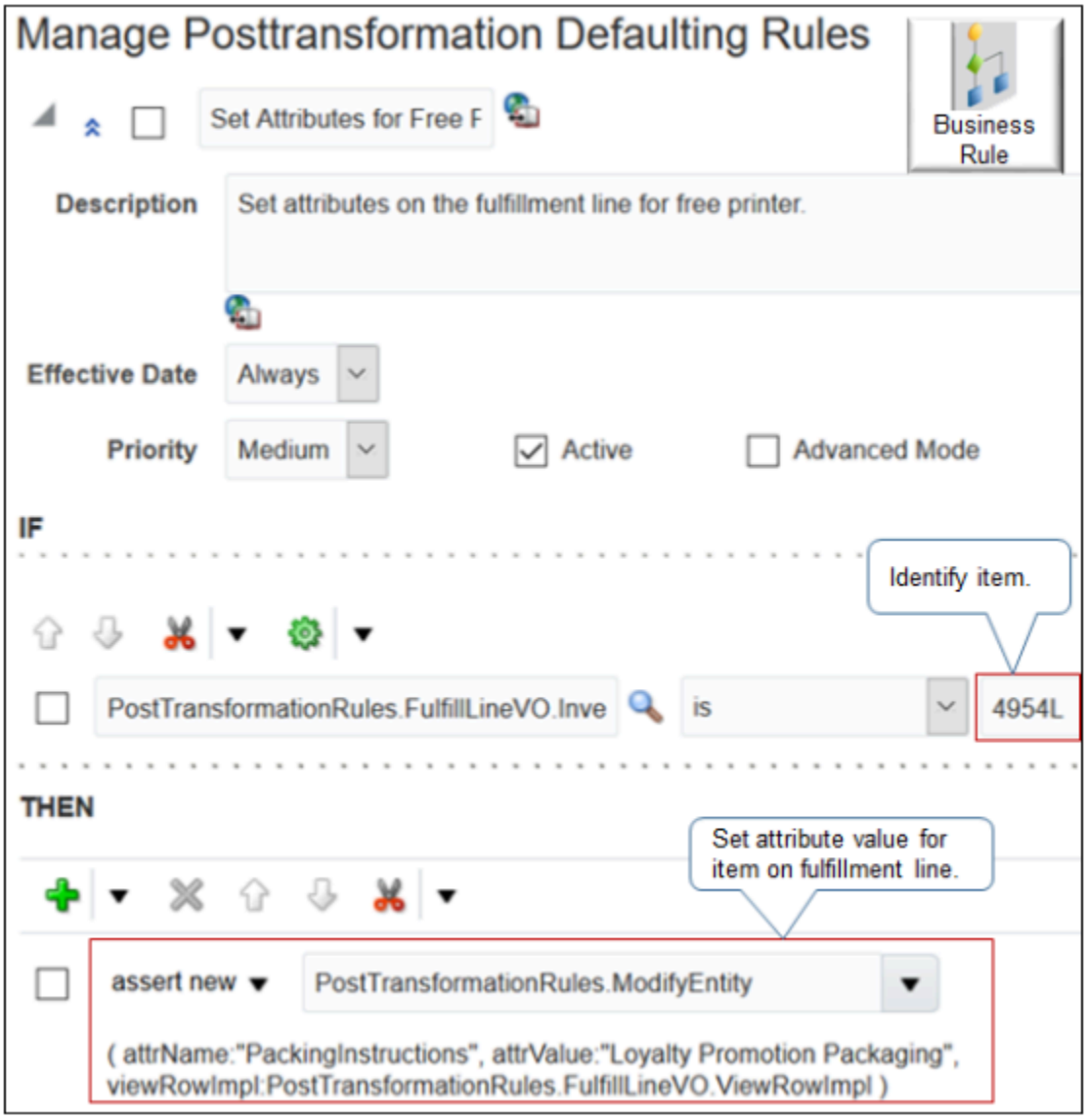


6. Click **Save and Close**.

Create Posttransformation Rule

Order Management must add values to some fulfillment line attributes when it creates the fulfillment line for the printer. You use a posttransformation rule to add the values.

Here's the rule you will create.



Do it.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Posttransformation Defaulting Rules

1. On the Manage Posttransformation Defaulting Rules page, click **Add > Add Rule**.
2. Click **Properties**, then set the values.

Attribute	Value
Name	Set Attributes for Free Printer

Attribute	Value
Description	Set attributes on the fulfillment line for free printer.
Effective Date	Always
Priority	Medium
Active	Contains a check mark
Advanced Mode	Does not contain a check mark

3. In the If area, create a statement.

```
PostTransformationRules.FulfillLineVO.InventoryItemId is 4954L
```

4. Add the Then statement.
    - In the Then area, click **Add Action > Assert New**.
    - In the empty window, enter `modify`, wait a moment, then click **PostTransformationRules.ModifyEntity**.
    - In the Then area, click **Edit Properties**.
    - In the Properties dialog, set the values, then click **OK**.

Attribute	Value
attrName	"PackingInstructions"
attrValue	"Loyalty Promotion Packaging"
viewRowImpl	PostTransformationRules.FulfillLineVO.ViewRowImpl

5. Repeat step 4 for each attribute you must set.
6. Click **Save and Close**.

Related Topics

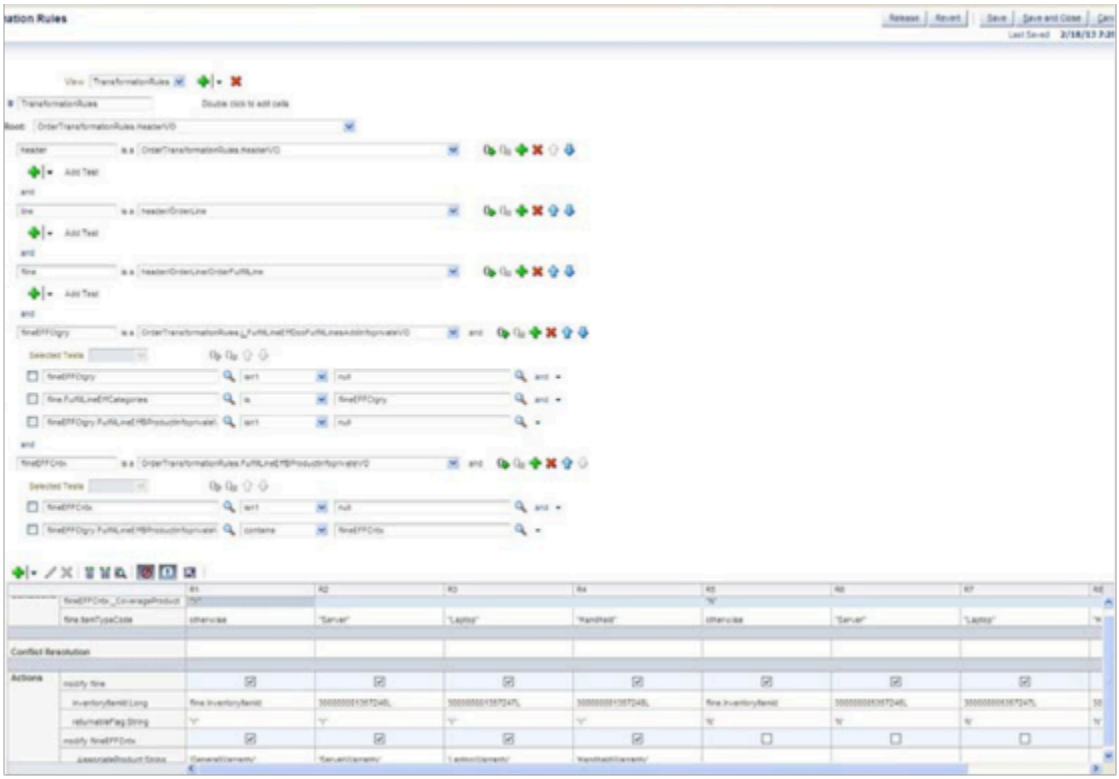
- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management
- Overview of Using Business Rules With Order Management
- How Order Management Transforms Source Orders Into Sales Orders

Use Extensible Flexfields in Advanced Transformation Rules

Use an extensible flexfield in a decision table to implement more complex logic.

This example uses business rules to transform an item. It replaces a general item with a server, laptop, and handheld device. It makes the replacement according to the product type and a Boolean value in an extensible flexfield segment.

Here's the rule you create. For details, see *Overview of Using Business Rules With Order Management*.



This example includes.

- A decision table that manages a set of rules
- **`header`** as an alias for the order header object
- **`line`** as an alias for the order line object
- **`fline`** as an alias for the fulfillment line object

This example includes extensible flexfield details.

- Extensible flexfield named ProductInfo
- flineEFFctgry for the extensible flexfield category
- flineEFFcntx for the extensible flexfield context
- An underscore (_) before the name of the extensible flexfield segment

You will use Root Mode to set up the hierarchy.

```
Order Header
 orderLine
 orderFulfillLine
```

where

- Order Header is the root
- orderLine is a child of Order Header
- orderFulfillLine is a child of orderLine and a grandchild of Order Header

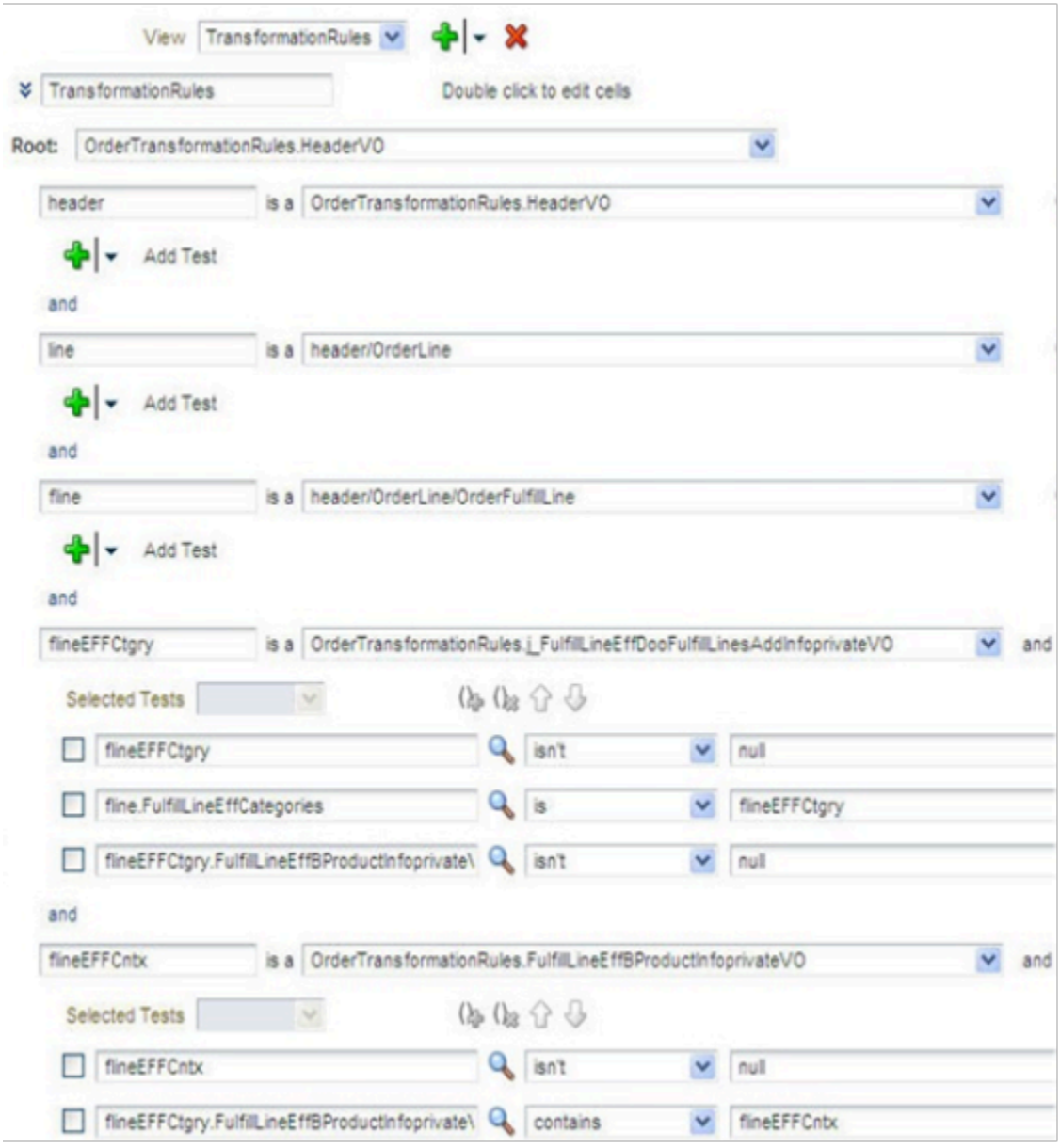
Summary of the Set Up

1. Create the IF statements and tests.
2. Create the decision table.

This topic uses example values. You might need different values, depending on your business requirements.

Create the IF Statements and Tests

You will create IF statements and tests.



Do it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Product Transformation Rules
2. On the Manage Product Transformation Rules page, in the View list, click **IF/THEN Rules**.
3. Click **Properties**.
4. Create a rule.

Attribute	Value
Effective Date	Always
Priority	Medium
Active	Contains a check mark
Advanced Mode	Contains a check mark
Tree Mode	Contains a check mark

5. Set the root.

Attribute	Value
Root	DooSeededOrchestrationRules.DOOHeader

6. Create IF statements.
    - header is a OrderTransformationRules.HeaderVO
    - line is a header/OrderLine
    - fline is a header/OrderLine/OrderFulfillLine
    - flineEFFCtgry is a OrderTransformationRules.j_FulfillLineEffDooFulfillLinesAddInfoprivateVO
    - flineEFFCnts is a OrderTransformationRules.FulfillLineEffBProductInfoprivate

7. Create tests.
    - flineEFFCtgry isn't null
    - flineEFFCntx isn't null
    - fline.FulfillLineEffCategories is flineEFFCtgry
    - flineEFFCtgry.FulfillLineEffBProductInfoprivateVO isn't null
    - flineEFFCtgry.FulfillLineEffBProductInfoprivateVO contains flineEFFCntx

Make sure you include a hierarchy test when you create extensible flexfield variables. The extensible flexfield in this example is ProductInfo. It contains the segment _CoverageProduct that you will use as a condition in the decision table.

Create the Decision Table

Here's the decision table you will create.

Note

- Create a condition that examines the _CoverageProduct extensible flexfield segment and fline.ItemTypeCode attribute.
- The actions modify the fline.InventoryItemId attribute to transform one product to another product, and also modify the fline.ReservableFlag attribute.
- A business rule can't modify the value of an extensible flexfield.
- The ModifyEntity function modifies attribute values.

If each of your business rules.

- **Use the same value**. You can create them as part of the condition.
- **Don't use the same value**. You must set option Parameterized for the condition so the rule can set a different value for each rule. Some actions in the decision table in this example return the existing product value instead of a substitution. Also, some actions aren't active for some rules. To make an action active, make sure the option above the value contains a check mark.

Related Topics

- Overview of Integrating Order Management with Accounts Receivable
- Use Extensible Flexfields to Integrate Order Management with Other Oracle Applications
- Pricing Algorithms
- Service Mapping
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together

Use Extensible Flexfields in Assignment Rules

Use an extensible flexfield to assign the fulfillment steps that your orchestration process does for each fulfillment line.

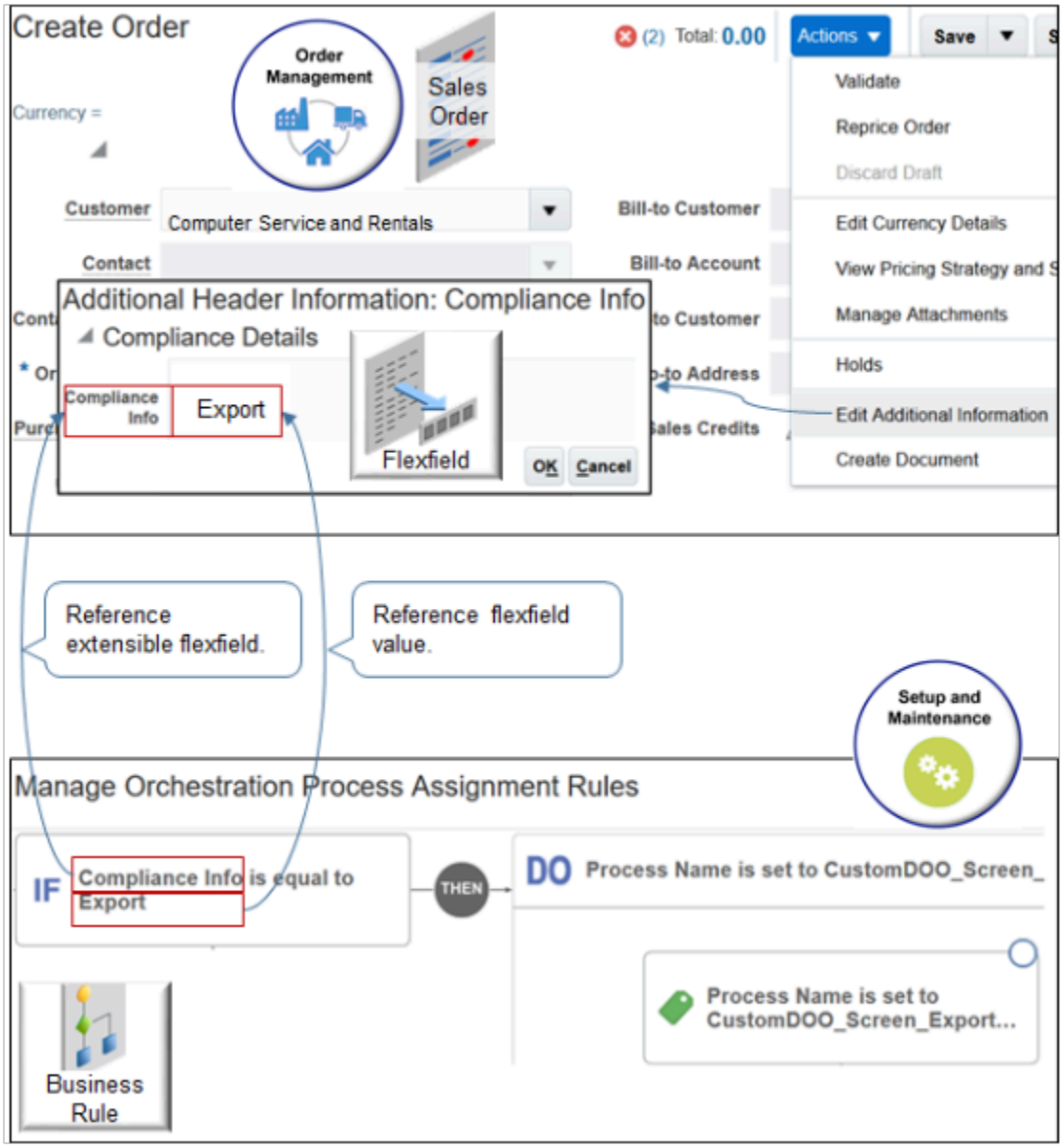
The fulfillment steps you use to fulfill an item might be different for each sales order depending on your business requirements.

An item attribute typically determines the orchestration process that runs and how to run orchestration process steps. If no predefined attributes meet your requirements, then you can create your own extensible flexfield, use it to capture important details about the item, then use it to select the fulfillment steps that run. You can write an assignment rule that uses extensible flexfield data as part of the selection criteria.

Assume.

- You sell an item in your own country and to several other countries throughout the world.
- The item includes details that are protected through international trade agreements.
- If you ship the item to a location that's outside of your own country, then the trade agreements require you to consider it an export and screen it for trade compliance.
- You do a needs analysis and determine to create the Trade Compliance Details extensible flexfield and add it to the order header so the Order Entry Specialist can enter the value Export to indicate whether the sales order is an export.

You will create an assignment rule that references the extensible flexfield.



You will create an assignment rule.

- If the Trade Compliance Details extensible flexfield contains the value Export, then assign the fulfillment line to the Screen_Exports_for_Trade_Compliance orchestration process.

This topic uses example values. You might need different values, depending on your business requirements.

Assign an orchestration process according to the value of an extensible flexfield.

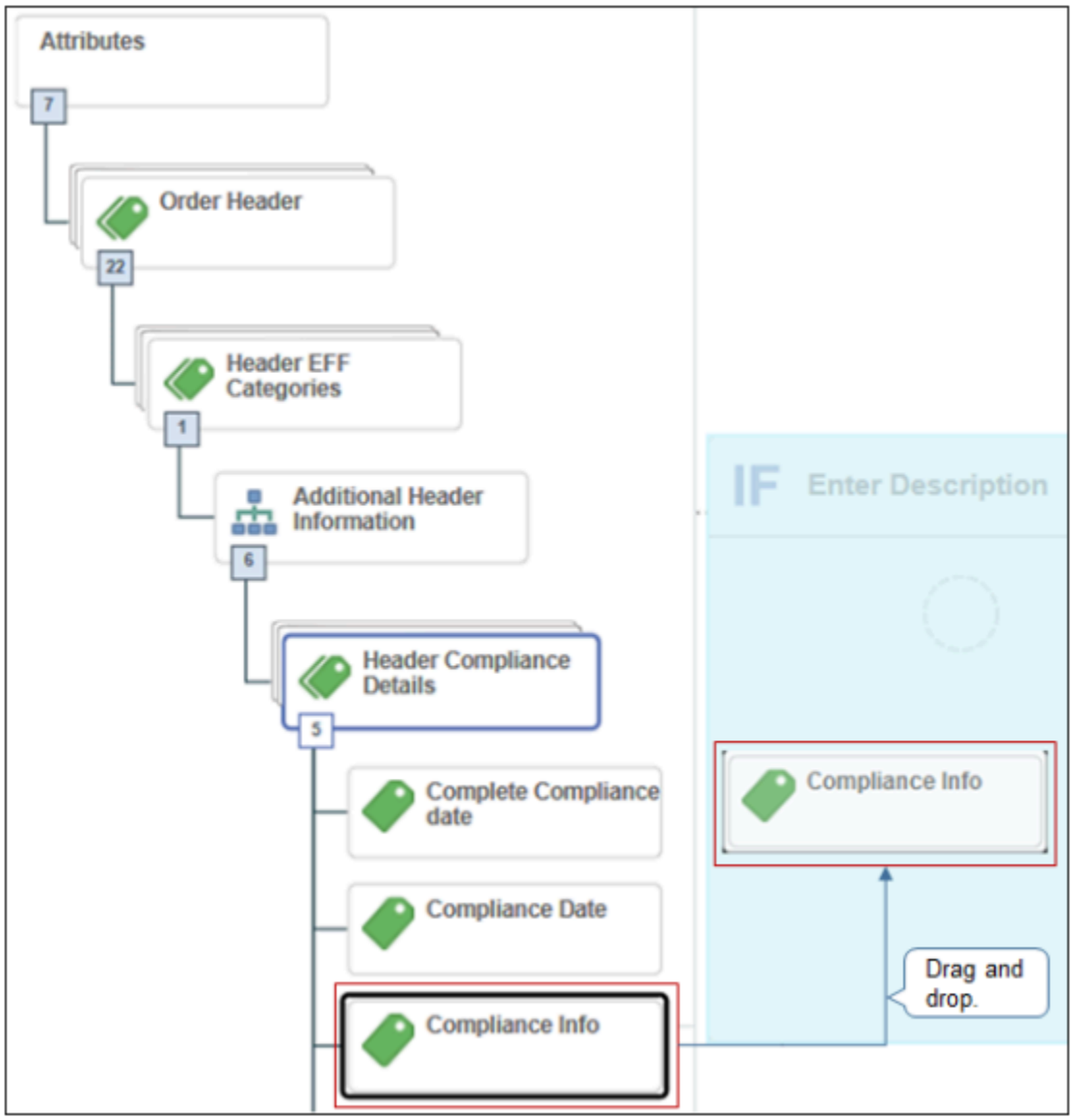
1. Set up the extensible flexfield and the orchestration process. Make sure you publish the extensible flexfield. If you don't publish, it won't show up the Manage Orchestration Process Assignment Rules page.

   For details, see *Overview of Using Extensible Flexfields in Order Management*.

2. Create the assignment rule.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Order Management
        - Functional Area: Orders
        - Task: Manage Process Assignment Rules for Sales Orders
    - On the Manage Orchestration Process Assignment Rules page, click **Create New Rule**.
    - Set the values.

Attribute	Value
Name	Assign for Trade Compliance
Description	Assign orchestration process depending on export.

3. Create the If statement.
    - Click in the If area to expand it.
    - On the Attributes tab, expand **Order Header > Header EFF Categories > Additional Header Information > Header Compliance Details**.
    - Click `Compliance Info`, drag it, then drop it into the IF area.



    - In the Create Condition dialog, set the operator to `Is Equal To`.
    - Enter `Export`, then click **OK**.

4. Create the Do statement.
    - Click **Then > Do > New Action**.
    - In the Create Action dialog, enter `process`, wait a moment, then click **Process Name (Order Fulfill Line)**.

      The phrase Order Fulfill Line indicates that the orchestration process you set will process order fulfillment lines.
5. Search for your orchestration process.
    - Click **Search > Advanced**.
    - Set Process Name to Contains.
    - Enter `Screen_Exports_for_Trade_Compliance`.

      The search is case sensitive.
    - Click **Search**.

      For this example, assume you already created and deployed this orchestration process. If you didn't deploy, then Search won't find it.
    - Click the **row** in the search results.
    - Click **OK**.
    - In the Create Action dialog, click **OK**.
6. Activate and publish your rule.

Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management
- Overview of Using Business Rules With Order Management

Use Extensible Flexfields in Line-Selection Rules

Use the DooSeededOrchestrationRules object to reference an orchestration process when you use an extensible flexfield in a line-selection rule.

For example, assume you must set up a rule.

- If the order line contains a reward item, then use it when running this orchestration process step.

Here's the logic you would use.

Object	Description
Pattern	Use patterns. • Root: DooSeededOrchestrationRules.DOOHeader • header is a DooSeededOrchestrationRules.DOOHeader • fline is a header/childFLines • flineEFF is a header/childFLines/flexContexts

Object	Description
Test	Use tests. • flineEFF.context.equalsIgnoreCase("Item_Information") is DooSeededOrchestrationRules.Boolean.TRUE • flineEFF.getFlexAttributeValue("_Reward_Item") isn't null Note • The equalsIgnoreCase function is optional if you know the absolute case of the argument. • The getFlexAttributeValue() function on the flexContexts object looks up the extensible flexfield value.
Action	Use an Assert New action. • DooSeededOrchestrationRules.Result (resultObjKey:FLine.fulfillLineId)

This logic creates a hierarchy.

```
Order Header
 childFLines
 flexContext
```

where

- Order Header is the root
- childFLines is a child of Order Header
- flexContext is a child of childFLines and a grandchild of Order Header

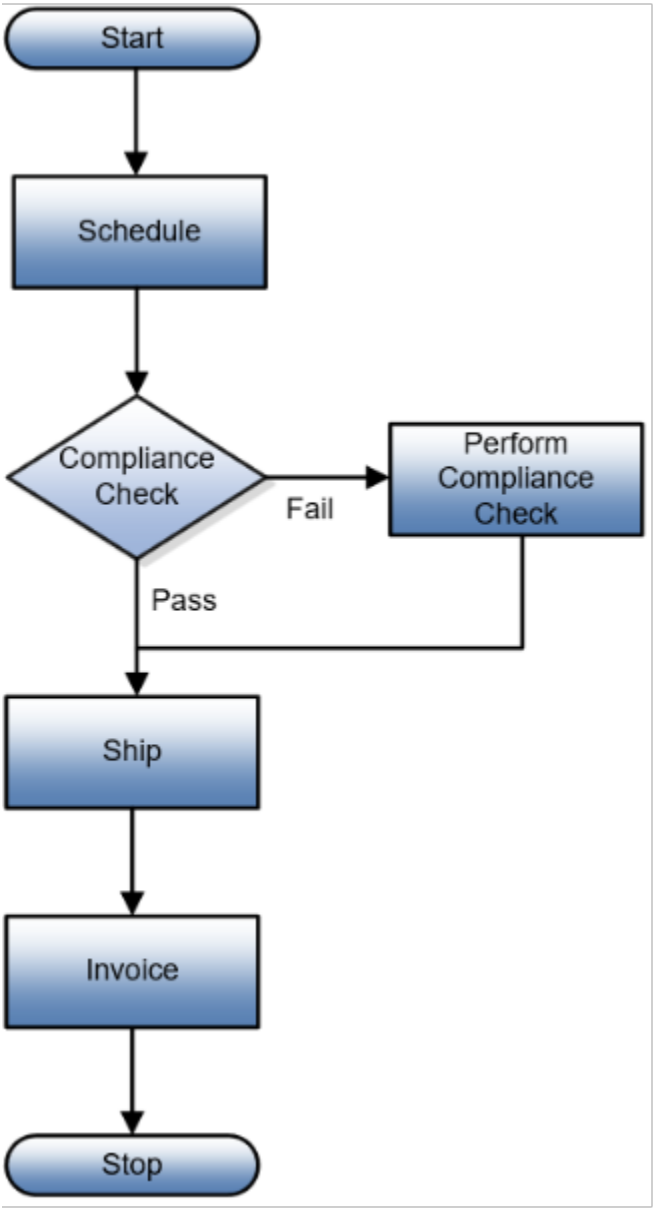
This example uses Tree Mode to maintain the hierarchy rather than explicitly creating the hierarchy. The forward slash (/) specifies the hierarchy. For example:

- `header/childFLines/flexContexts`

In this example, you will define line selection criteria in an orchestration process to specify which orchestration process steps to run for each fulfillment line. You define this criterion according to the value that an extensible flexfield contains. You create a rule.

- If the Export Compliance status is Not Passed for an order line, then do a manual examination. Send other order lines directly to shipping.

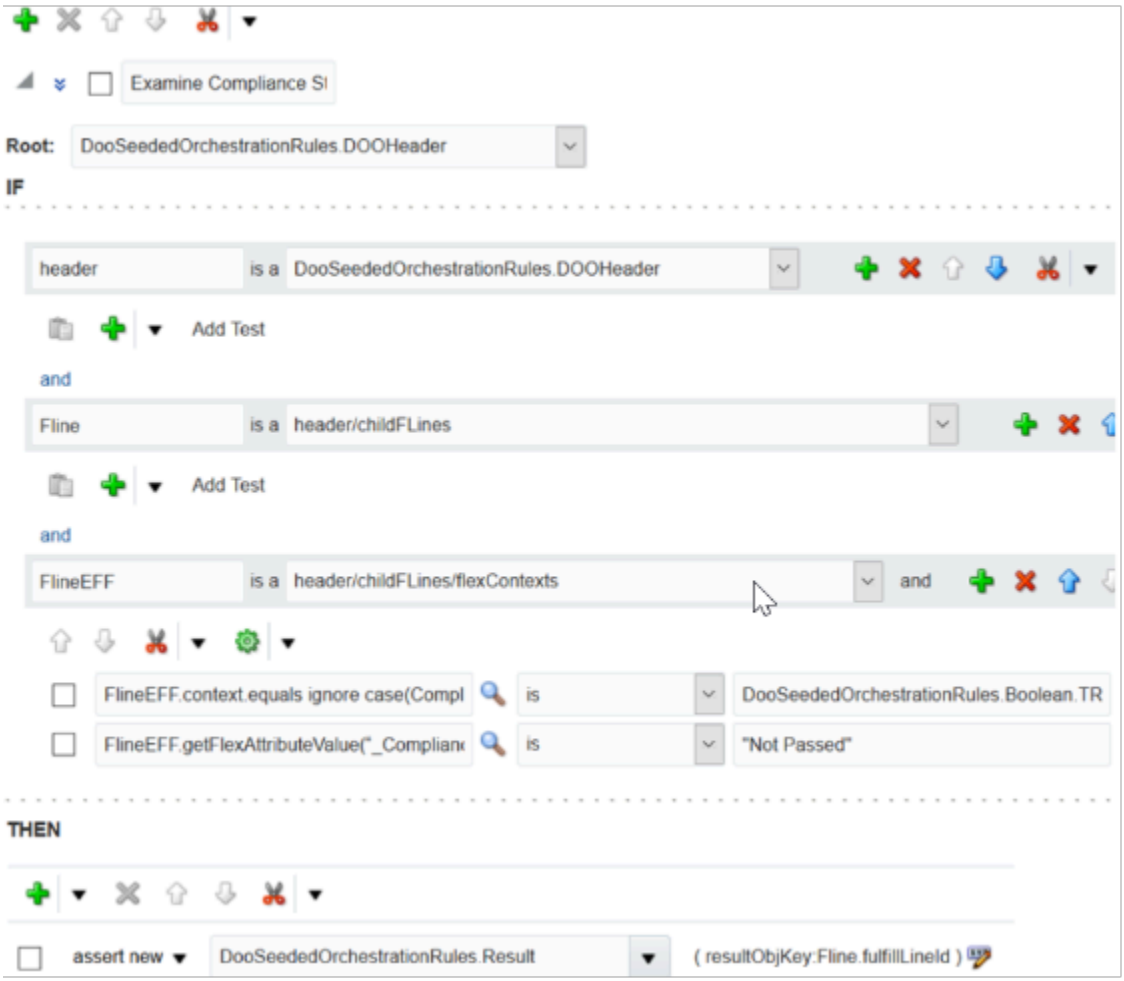
Here's the flow you create.



You will use an extensible flexfield to store the value of the compliance status.

Context	Extensible Flexfield	Object	Value
Compliance	Compliance_Status	Fulfillment Line EFF (DOO_ FULFILL_LINES_ADD_INFO)	Not Passed

You will create a line-selection rule.



Summary of the Steps

Fulfill order lines according to the value of an extensible flexfield.

1. Create the rule.
2. Establish the object hierarchy.
3. Select lines according to flexfield context.
4. Select lines according to compliance status.
5. Identify the fulfillment line that requires manual examination.

For details about how to create a business rule, see *Overview of Using Business Rules With Order Management*.

This topic uses example values. You might need different values, depending on your business requirements.

Create the Rule

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions

2. On the Manage Orchestration Process Definitions page, locate the orchestration process you must edit, then click **Actions > Edit**.
3. On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, add a step.

Attribute	Value
Step Name	Compliance Check
Step Type	Service
Task Type	DOO_TradeCompliance
Task	DOO_TradeCompliance
Service	Request Screening for Trade Compliance

4. In the row you just added, in the Line-Selection Criteria column, click **Click for Rule**.
5. In the Line-Selection Criteria dialog, click **Add Rule > Properties**, then set the values.

Attribute	Value
Name	Examine Compliance Status
Description	If the Export Compliance status is Not Passed for an order line, then perform a manual examination.
Effective Date	Always
Priority	Medium
Active	Contains a check mark.
Advanced Mode	Contains a check mark.
Tree Mode	Contains a check mark. Use tree mode so you can include an extensible flexfield in the root hierarchy of an orchestration rule.

6. Set the root.

Attribute	Value
Root	DooSeededOrchestrationRules.DOOHeader

Establish the Object Hierarchy

You must establish an object hierarchy so the rule filters the data it processes.

```
Order Header
 childFLines
 flexContexts
```

where

- Order Header is the root
- childFLines is a child of Order Header
- flexContexts is a child of childFLines and a grandchild of Order Header

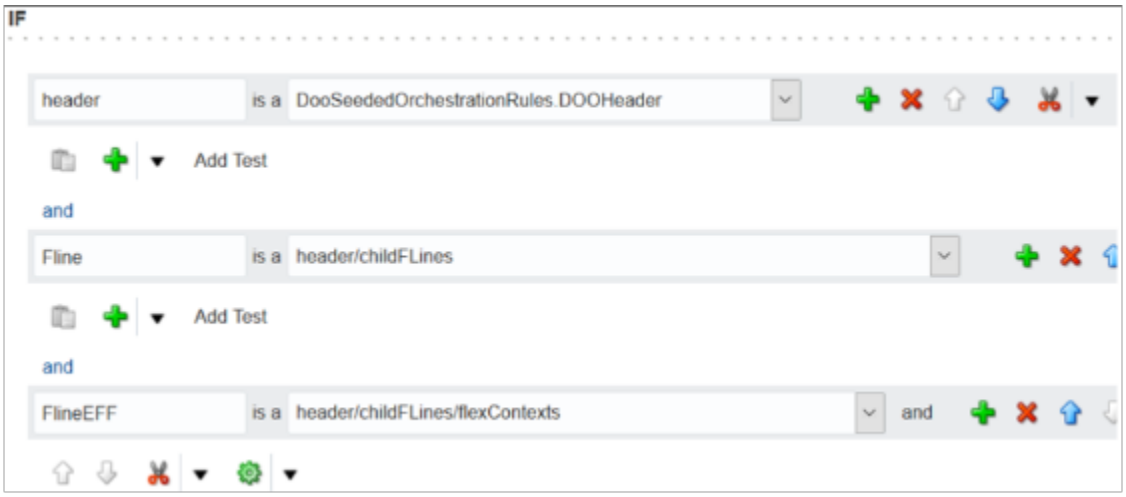
This example uses Tree Mode to maintain the hierarchy rather than explicitly creating the hierarchy. Each forward slash (/) specifies a hierarchy level.

```
header/childFLines/flexContexts
```

where

- header is the root. The root makes sure the rule processes data only in the child object that references the root.
- childFLines is the child. The child makes sure the rule processes data only in the grandchild object that references the child.
- flexContexts is the grandchild.

You will create an If statement. It establishes the object hierarchy.



Establish the object hierarchy.

1. In the If area, delete the value in the field to the left of Is A, and then enter this value.

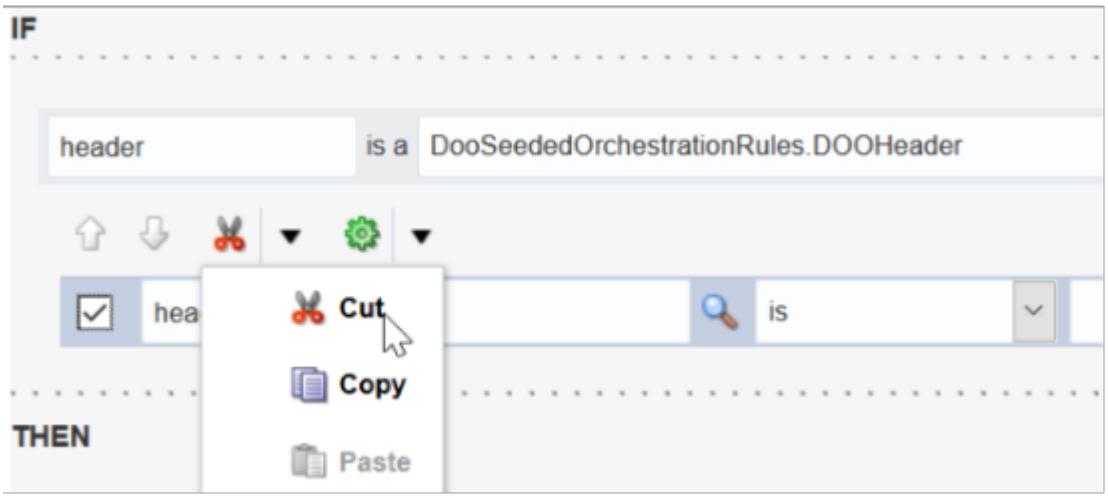
Attribute	Value
Field to the left of Is A	Header

2. In the field to the right of Is A, set the value.

Attribute	Value
Field to the right of Is A	DooSeededOrchestrationRules.DOOHeader

3. Under the field that contains `header`, add a check mark to the option that selects the test, click **Cut > Cut**.

   For example:



   The hierarchy includes the extensible flexfield variables, so its not necessary to include tests when you define the hierarchy.
4. Click **Add Pattern**.
5. In the field to the left of Is A, enter the value.

Attribute	Value
Field to the left of Is A	`Fline`

6. In the field to the right of Is A, set the value.

Attribute	Value
Field to the right of Is A	header/childFLines

7. Click **Add Pattern**.
8. In the field to the left of Is A, enter the value.

Attribute	Value
Field to the left of Is A	FlineEFF

9. In the field to the right of Is A, set the value.

Attribute	Value
Field to the right of Is A	header/childFLines/flexContexts

Select Lines According to Flexfield Context

You will add a test.

- If the flexfield context is Compliance

This text will select lines only where the flexfield context is Compliance.

You will create an expression.

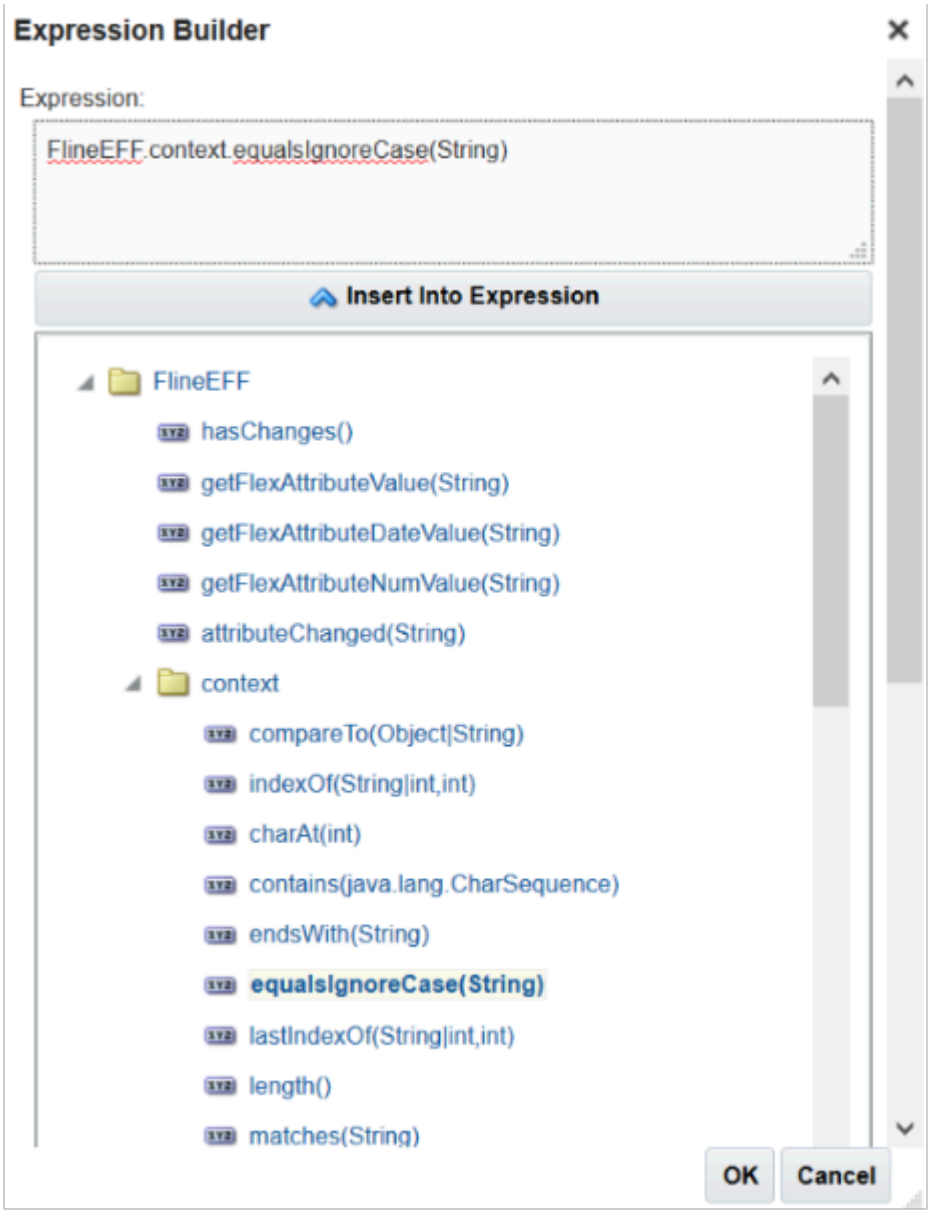
Left Expression	Operand	Right Expression
FLineEFF.context.equalsIgnoreCase("Compliance	Is	DooSeededOrchestrationRules.Boolean.TRUE

Select lines according to flexfield context.

1. Under the pattern you just added, click **Add Test**.

   You will add the test criteria that identifies fulfillment lines that proceed through compliance testing. If the attributes you reference are optional, then you include null value checks to avoid a null pointer exception at run time.
2. Click **Left Value**.
3. In the Condition Browser, click **Expression Builder**.
4. In the Expression Builder, click **Functions**, expand **FLineEFF > Context**, click **equalsIgnoreCase(String),** then click **Insert Into Expression**.

   For example:



To navigate the tree structure, expand the folders until you can view the functions of the context object. FlineEFF is the pattern you declared earlier.

The context you expand is a fact in the FlineEFF pattern. It includes functions you can use to perform a variety of calculations. Therefore, the expression you create includes the pattern, fact, and function.

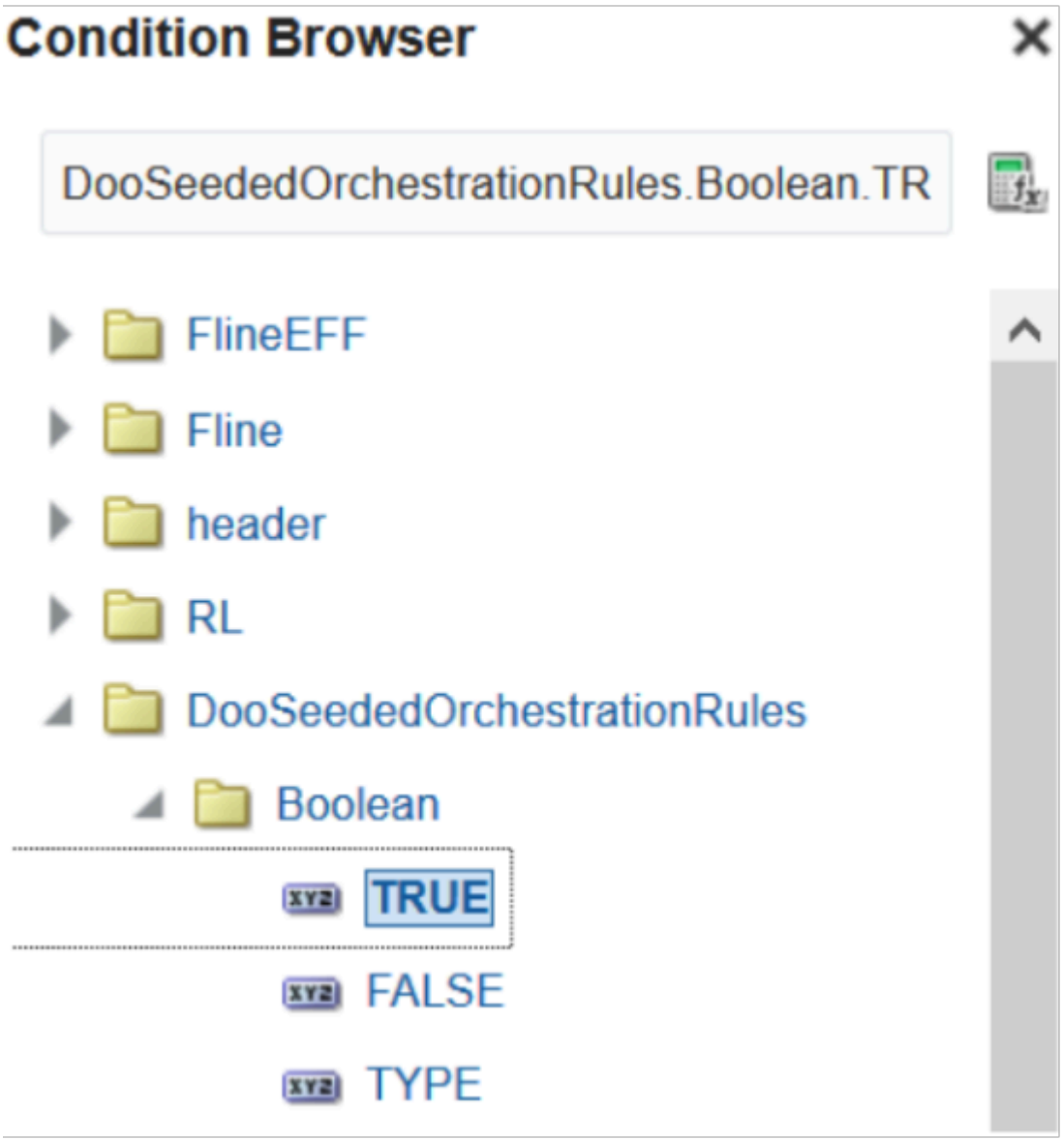
5. In the window near the top of the dialog, replace the word `String` with the context for your extensible flexfield condition. In this example, replace `String` with `Compliance`.

   For example:

6. Click **OK > OK**.
7. Click **Right Value**.

8. In the Condition Browser, expand **DooSeededOrchestrationRules > Boolean**, then click **TRUE > OK**.

   For example:



Select Lines According to Compliance Status

You will add a test.

- If the value of the flexfield attribute named Compliance Status is Not Passed

You will create an expression.

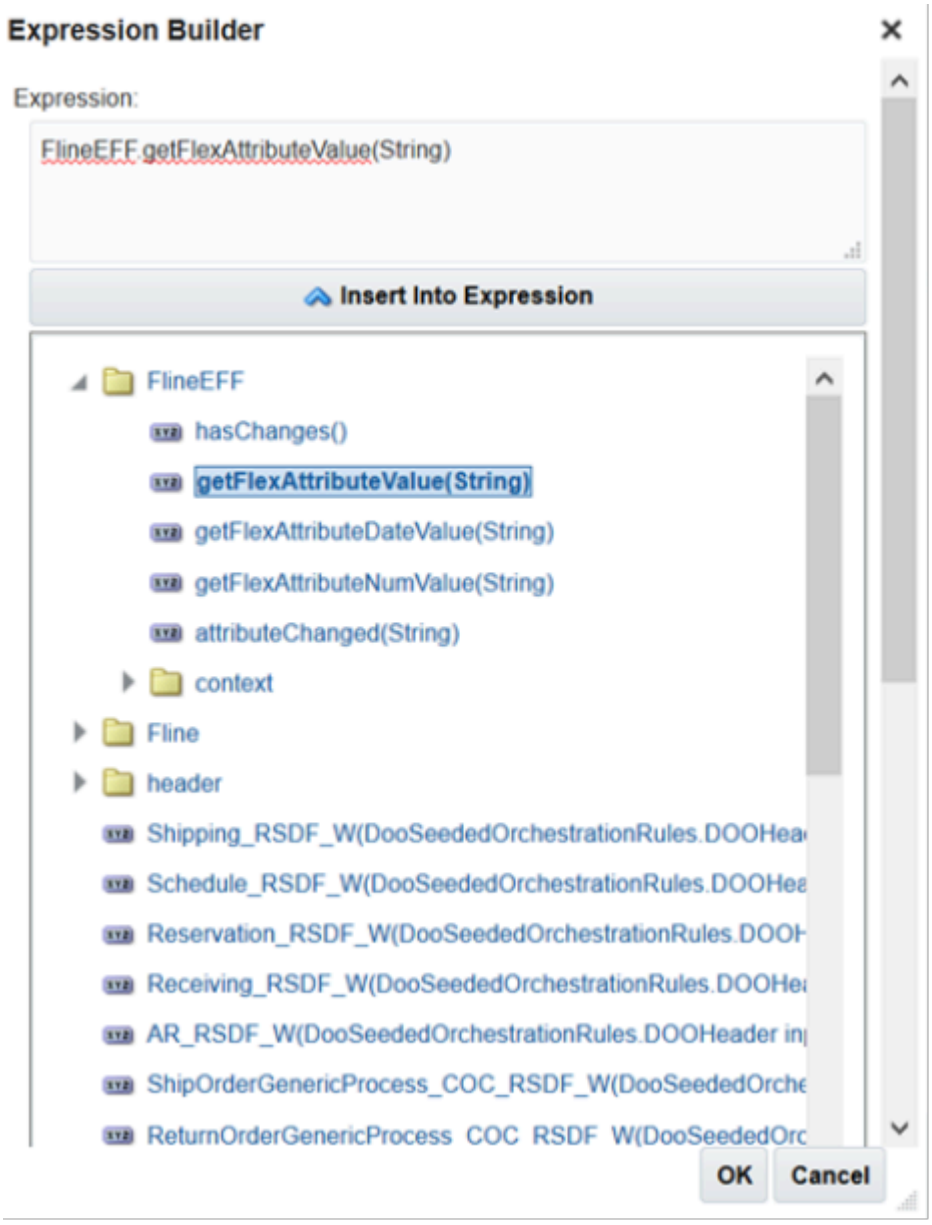
Left Expression	Operand	Right Expression
flineEFF.getFlexAttributeValue("_Compliance_Status")	Is	"Not Passed "

Select lines according to Compliance Status.

1. In the test you just added, immediately to the right of the Right Value magnifying glass, click the **down arrow**, then click **Simple Test**.
2. Click **Left Value**.
3. In the Condition Browser, click **Expression Builder**.

4. In the Expression Builder, click **Functions**, click **FlineEFF.getFlexAttributeValue(String)**, then click **Insert Into Expression**.

   For example:



5. In the window near the top of the dialog, replace the word String with the name of the flexfield attribute.

   You must use an underscore to prefix the attribute name in a function. The attribute name is Compliance_Status, so you use _Compliance_Status.

   For example:

6. Click **OK > OK**.
7. In the field to the right of Is, enter `"Not Passed"`.

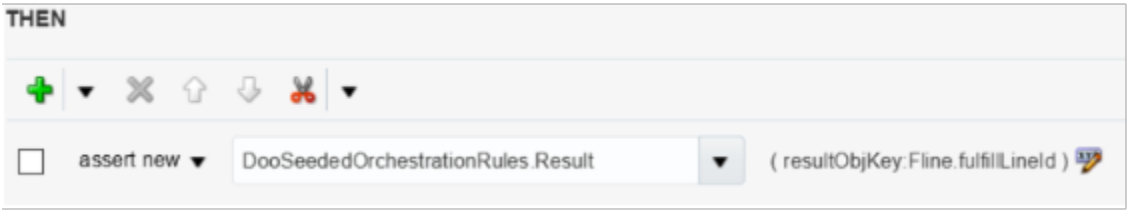
   You must include the double quotation marks (").

   Make sure your rule contains tests.

Left Expression	Operand	Right Expression
FLineEFF.context.equalsIgnoreCase("Compli	Is	DooSeededOrchestrationRules.Boolean.TRUE
flineEFF.getFlexAttributeValue("_ Compliance_Status")	is	"Not Passed"

Identify the Fulfillment Line That Requires Manual Examination

You will create a statement.



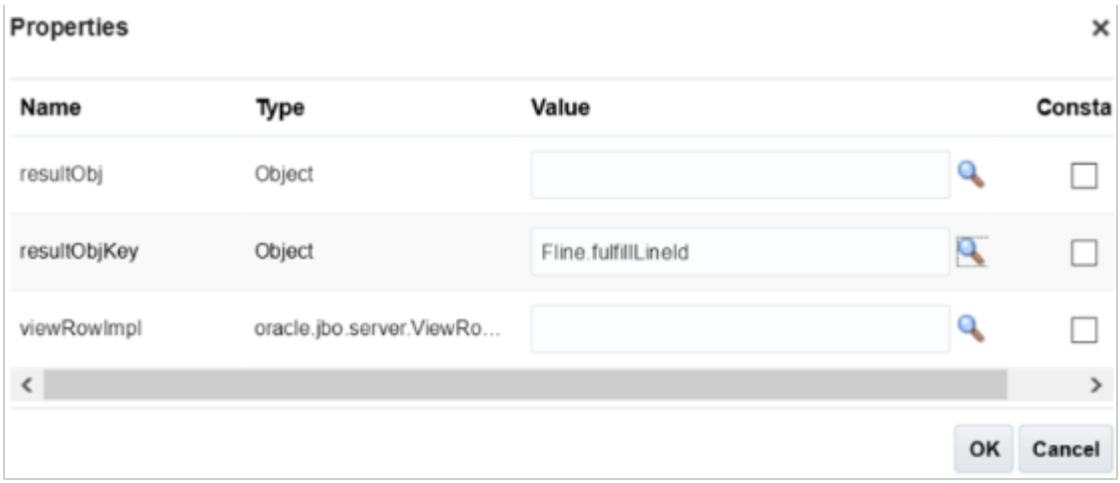
where

- Result is a fact in the DooSeededOrchestrationRules dictionary.
- resultObjKey is a property of the Result fact.
- resultObjKey stores the value that this rule uses to identify the fulfillment line that requires manual examination.
- `Fline` is a fact in the DooSeededOrchestrationRules dictionary that contains fulfillment line attributes.

- fulfillLineID is a fulfillment line attribute.

Identify the fulfillment line that requires manual examination.

1. In the Then area, click **Add Action > Assert New**.
2. Click **Select a Target > DooSeededOrchestrationRules.Result**.
3. Click **Edit Properties**.
4. In the Properties dialog, in the resultObjKey row, click **Value**.
5. In the Condition Browser, expand `Fline`, click **fulfillLineID**, then click **OK**.
6. Make sure the Properties dialog looks like this:



7. Click **OK**
8. In the Line-Selection Criteria dialog, click **Save**.
9. On the Edit Orchestration Process Definition page, click **Save**.

Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management
- Overview of Using Business Rules With Order Management

Another Example of Using Extensible Flexfields In Line-Selection Rules

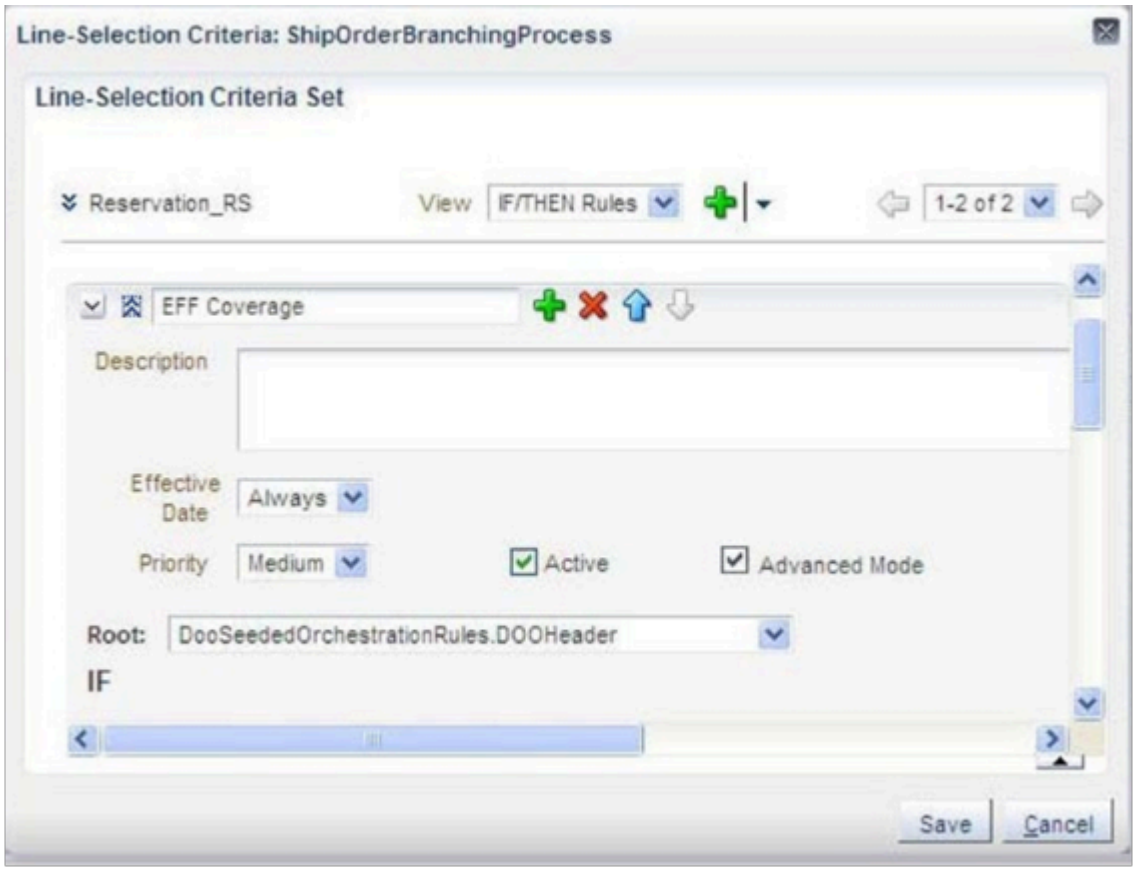
Create a line selection rule that references an extensible flexfield.

- If the value in the CoverageProduct extensible flexfield doesn't contain a check mark, then reserve the item and process the fulfillment line.

Fulfill order lines according to the value of an extensible flexfield.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Orchestration Process Definitions
2. On the Manage Orchestration Process Definitions page, locate the orchestration process you must edit then click **Actions > Edit**.

3. On the Edit Orchestration Process Definitions page, in the Process Details area, in the Step Definition list, in the row that contains the reservation step, in the Line-Selection Criteria column, click **Click for Rule**.
4. In the Line-Selection Criteria dialog, add values.



Use tree mode to include an extensible flexfield in the root hierarchy of an orchestration rule. The hierarchy includes extensible flexfield variables so it isn't necessary to add more tests when you define the hierarchy.

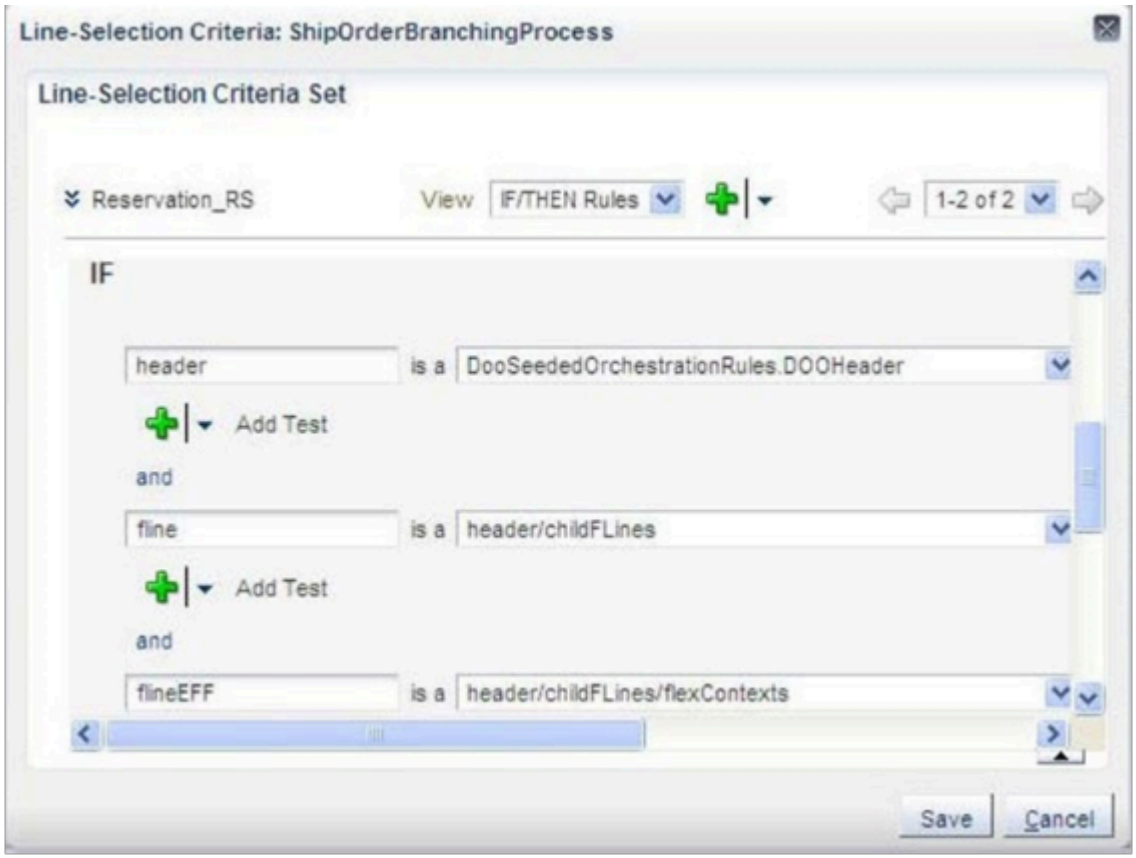
5. Define IF statements.

    - `header is a DOOSeededOrchestrationRules.DOOHeader`
    - `fline is a header/childFlines`
    - `flineEFF is a header/childFLines/flexContexts`

    where

    - `header` is a variable that stores the order header.
    - `fline` is a variable that stores the fulfillment line.
    - `flineEFF` is a variable that stores the extensible flexfield.

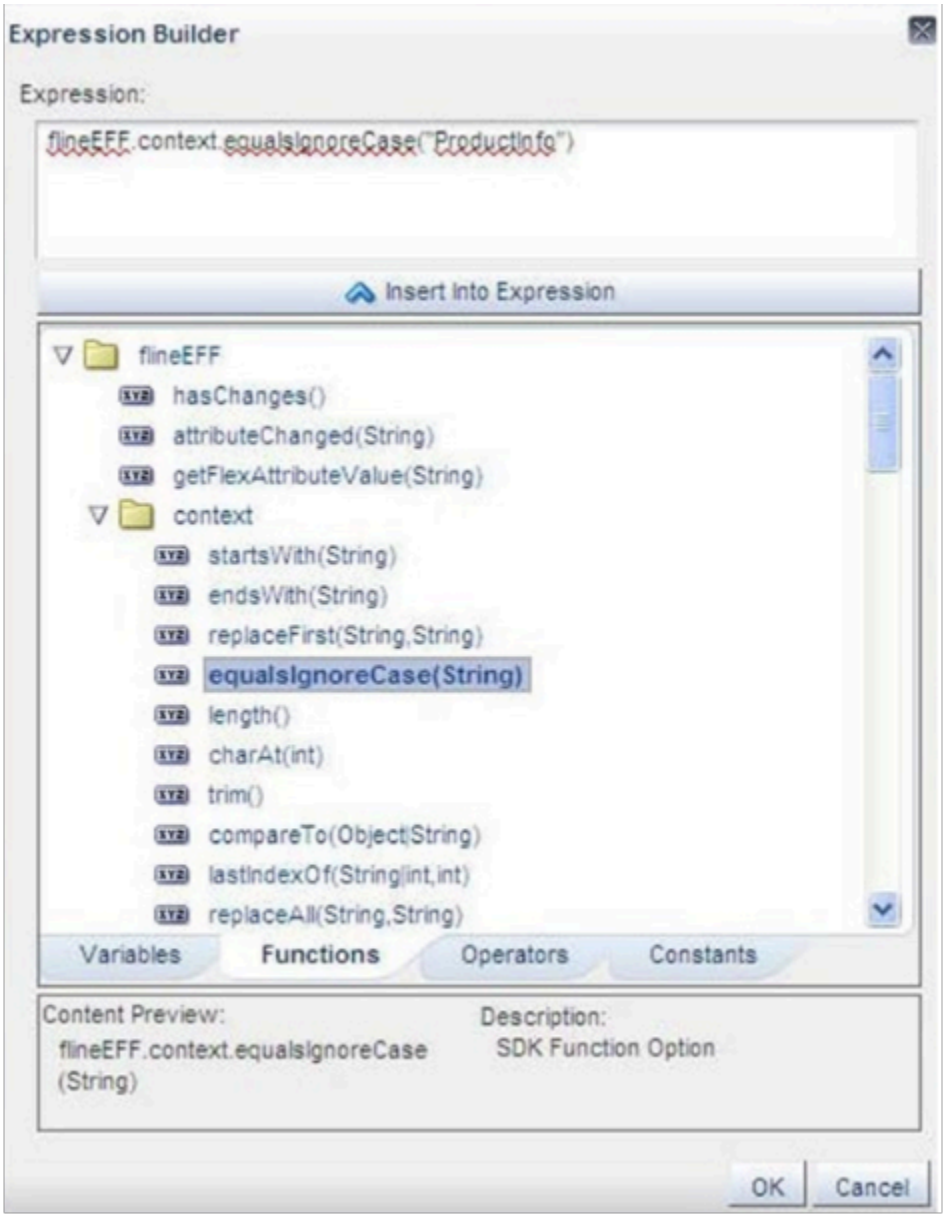
    For example:



6. Create an expression that looks up the extensible flexfield value in the tests.

    - `flineEFF.context.equalsIgnoreCase("ProductInfo")`

      For example:

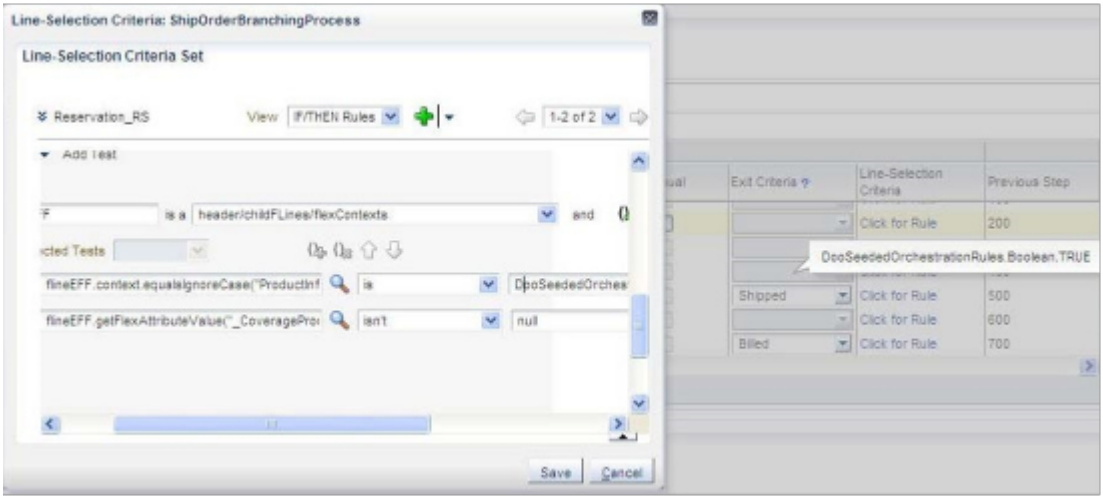


7. Create tests.

Code	Description
`flineEFF.context.equalsIgnoreC is DooSeededOrchestration Rules.Boolean.TRUE`	Make sure the item isn't a coverage item.

Code	Description
`flineEFF.getFlexAttributeValue isn't null`	Make sure the value in the flexfield isn't empty.

For example:



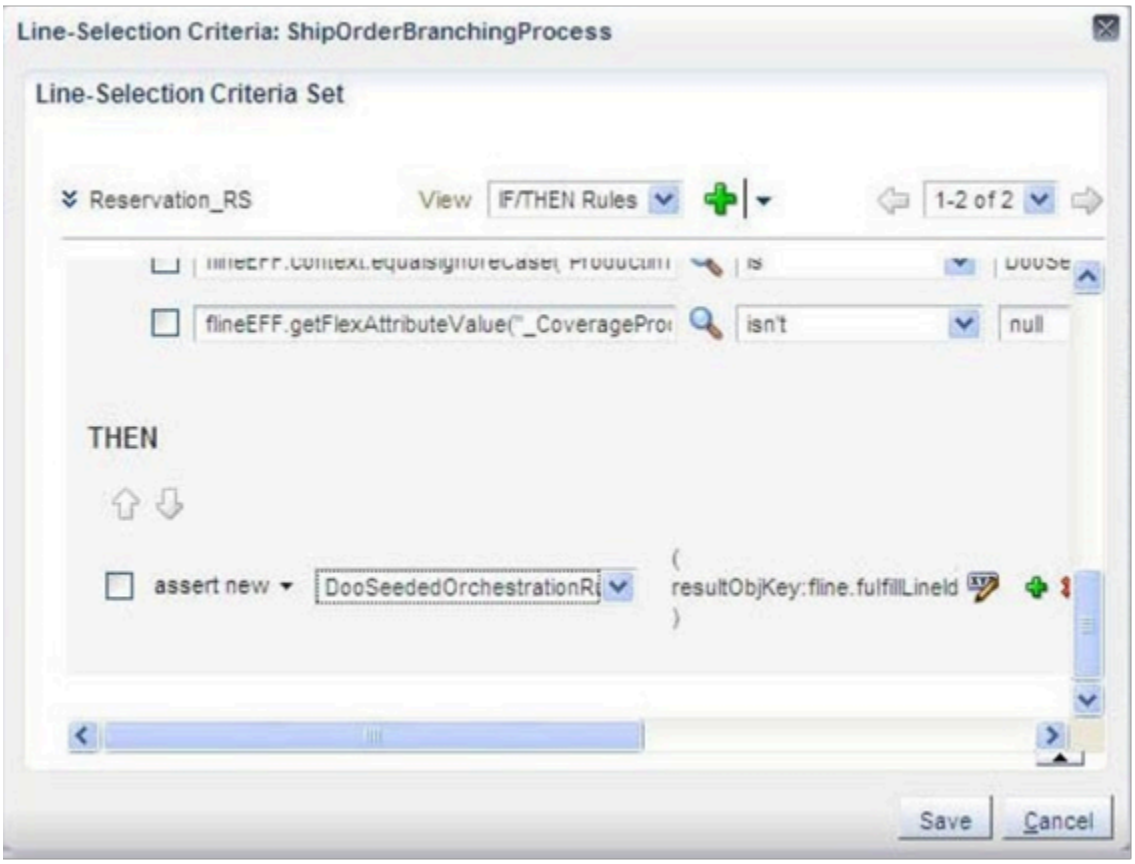
8. Create an action.

    - `Assert new DooSeededOrchestrationRules.Result(resultObjKey:fulfillLineId)`

      This action creates the result to use for the fulfillment line that satisfies the If statement.

      In the Condition Browser, select the attribute. This attribute sets resultObjKey to the run-time value of the fulfillLineId.

      For example:



Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Patterns That You Can Use with Extensible Flexfields in Business Rules
- Overview of Using Extensible Flexfields in Order Management
- Overview of Using Business Rules With Order Management

Use Extensible Flexfields in Change Management Rules

Create a cost of change rule to measure how much a change impacts an orchestration process. Use an extensible flexfield as input to the cost of change calculation.

You can also create a compensation pattern that specifies the adjustments that Order Management makes when it processes a fulfillment task. It does this work to process the request it receives to change a sales order in a way that makes sure it can fulfill the sales order so that it meets your fulfillment requirements.

For example:

- Assume a customer with a Platinum loyalty status receives a follow up phone call about their order, and that you defined an extensible flexfield named Loyalty that allows the Order Entry Specialist to store the loyalty status.
- On the change order, assume you define another extensible flexfield named Customer Satisfaction that allows the Order Entry Specialist to capture the level of customer satisfaction.
- If Order Management already sent a request to the fulfillment system that schedules a follow up phone call, but the Order Entry Specialist hasn't made the phone call, then the compensation pattern for customer satisfaction might be Cancel, meaning don't call the customer.
- If the compensation pattern for customer satisfaction isn't Cancel, then the rule can continue with the scheduled phone call.

You use an extensible flexfield in a change management rule the same way you use it in a line-selection rule. For details, see *Use Extensible Flexfields in Line-Selection Rules*.

Get details about change management.

- *Measure the Cost of Change*
- *Overview of Managing Change That Occurs During Order Fulfillment*
- *Compensate Sales Orders That Change*

Related Topics

- Use Extensible Flexfields in Line-Selection Rules
- Measure the Cost of Change
- Compensate Sales Orders That Change
- Overview of Using Extensible Flexfields in Order Management
- How Order Management Processes Change Orders

Identify Flexfield Contexts and Category Codes for Your Business Rules

Identify the flexfield contexts and category codes that you specify in an Oracle Business Rule.

In this example, assume you need to reference context code FND_ACFF_EFF_CONTEXT_CODE in your business rule.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Order Extensible Flexfields
2. On the Manage Order Extensible Flexfields page, search for Fulfillment Line Information.
3. In the search results, click the **row** that contains Fulfillment Line Information in the Name column, then click **Actions > Download Flexfield Archive**.
4. In the Confirmation dialog, click **Download**, then save the zip file to any folder.
   The file manager for your operating system opens. For example, if you're using Microsoft Windows, then Windows Explorer opens.

5. In your file manager, navigate to the folder in the zip file that contains the xml you must examine.
   `. . .\file_name.zip\oracle\apps\scm\doo\processOrder\flex\fulfillLineContextsB\view`

   where
   - . . . is the path where you save the zip file.
   - file_name. Name of the zip file.

   For example:
   `C:\Users\your_name\AppData\Local\Temp\10008_DOO_FULFILL_LINES_ADD_INFO.zip\oracle\apps\scm\doo\processOrder\flex\fulfillLineContextsB\view`
6. Use an XML editor to open the xml file that contains the context you must examine.
   The file list contains a variety of xml files, such as.
   - FulfillLineEffBFulfillLineContext1privateVO.xml
   - FulfillLineEffBPackShipInstructionprivateVO.xml

   You typically are interested in Context1private, so open FulfillLineEffBFulfillLineContext1privateVO.xml.
7. Search for EFF_CONTEXT_CODE, then notice that the full value of the name includes a prefix, such as FND_ACFF_EFF.
   For example:
   `<Property Name="FND_ACFF_EFF_CONTEXT_CODE" Value="FulfillLineContext1"/>`

   Most XML files have only one context code.
   Notice that the value is FulfillLineContext1. Use it when you set up your rule in Oracle Business Rules. For example, here's the code you use in your rule.
   `FLineEFF.context equals ignore case "FulfillLineContext1"`
8. Search for the ViewAttribute tag that contains the segment name you need.
   `ViewAttribute Name="_FL1AttributeChar1"`

   The file has only one FL1AttributeChar1 view attribute tag.
9. Notice that the value in the ViewAttribute's Name property is `_FL1AttributeChar1`. Use it when you set up your rule in Oracle Business Rules.
   For example, use this code in your rule:
   `HeaderEFF.getFlexAttributeDateValue("_CompleteCompliancedate")`

   or
   `FLineEFF. getFlexAttributeValue("_FL1AttributeChar1")`

   Get details about this example, including extensible flexfields. For details, see *Overview of Using Business Rules With Order Management*.

Patterns That You Can Use with Extensible Flexfields in Business Rules

Use patterns and hierarchies when you include an extensible flexfield in a business rule.

Rule Dictionaries That You Can Use with Extensible Flexfields

Rule Type	Dictionary
Pretransformation	PreTransformationRules
Posttransformation	PostTransformationRules
Transformation	OrderTransformationRules
Routing	DOOExternalInterfaceLayer
Assignment	AssignLaunchRules
Line Selection Criteria	DooSeededOrchestrationRules
Branching Condition	DooSeededOrchestrationRules
Lead Time Expression	DooSeededOrchestrationRules
Compensation Pattern	DooSeededOrchestrationRules
Cost of Change	DooSeededOrchestrationRules

Rule Patterns and Hierarchies That You Can Use With Extensible Flexfields

Patterns and Hierarchies That You Can Use to Display Extensible Flexfields on Order Headers

The **bold** text indicates a variable.

Pattern	Description
my_header	Here's the format. header is a **dictionary.fact** For example: `header is a OrderTransformationRules.HeaderVO` where • **header**. A variable you define when you create the rule.

Pattern	Description
	• **OrderTransformationRules**. The name of a predefined dictionary. • **HeaderVO**. A fact that resides in OrderTransformationRules.
my_header_extensible_flexfield_ category	Here's the format. header is a **dictionary**.j_HeaderEffDooHeadersAddInfoprivateVO
my_header extensible_flexfield_context	Here's the format. header is a **dictionary**.`HeaderEffContextNameprivateVO`

Note

- **dictionary** identifies the name of the dictionary that your rule references.
- DOO_HEADERS_ADD_INFO is the category for an order header.
- **ContextName** identifies the name of the context.

Your rule must establish a hierarchy.

Parent Pattern	Property	Child Pattern
my_header	.`HeaderEffCategories`	header is **my_header _extensible_flexfied_ category**
my_header _extensible_flexfied_category	.`HeaderEffB` **ContextNameprivateVO**	header contains **my_header _extensible_ flexfied_context**

Rule Patterns and Hierarchies That You Can Use to Display Extensible Flexfields on Order Lines

Use rule patterns when your extensible flexfield displays on the order line or the fulfillment line.

Pattern	Description
my_fline	Here's the format. **fline** is a **dictionary**.`FulfillLineVO` For example: `fline is a AssignLaunchRules.FulfillLineVO` where • **fline**. A variable you define when you create the rule. • **AssignLaunchRules**. The name of a predefined dictionary. • **FulfillLineVO**. A fact that resides in AssignLaunchRules.
my_fline_extensible_flexfield_category	Here's the format.

Pattern	Description
	my_category is a **dictionary**. j_FulfillLineEffDooFulfillLinesAddInfoprivateVO For example: `FlineEFFdfaip is a AssignLaunchRules.j_ FulfillLineEffDooFulfilllinesAddInfoprivateVO` where. • **FlineEFFdfaip**. A variable you define when you create the rule. • **AssignLaunchRules**. The name of a predefined dictionary.
my_fline_extensible_flexfield_context	Here's the format. **my_context** is a **dictionary**. FulfillLineEffBContextNameprivateVO For example: • FlineEFFpsi is a AssignLaunchRules.FulfillLineEffBPackShipInstructionprivateVO where • **FlineEFFpsi**. A variable you define when you create the rule. • **AssignLaunchRules**. The name of a predefined dictionary. • **PackShipInstruction**. The name of a context.

where

- **dictionary** identifies the name of the dictionary that your rule references.
- DOO_FULFILL_LINES_ADD_INFO is the category for each order line or fulfillment line.
- **ContextName** identifies the name of the context.

Your rule must establish a hierarchy.

Parent Pattern	Property	Child Pattern
my_fline	`.FulfillLineEffCategories`	**my_fline** is **my_ fline_extensible_flexfied_ category**
my_ fline_extensible_flexfied_category	`.FulfillLineEffB` **ContextName**privateVO	fline contains **my_fline _extensible_flexfied_ context**

Rule Patterns and Hierarchies That You Can Use to Display Extensible Flexfields on the General Tab of a Fulfillment Line

Use patterns when your extensible flexfield displays on the General tab of the fulfillment line.

Pattern	Description
my_fline_details	Here's the format.

Pattern	Description
	my_fline_details is a **dictionary**.`FulfillLineDetailVO`
my_fline_details_extensible_flexfield_category	Here's the format. **my_fline_details_extensible_flexfield_category** is a **dictionary**. j_FulfillLineDetailEffDooFulfillLineDetailsAddInfoprivateVO
my_fline_details_extensible_flexfield_context	Here's the format. **my_fline_details_extensible_flexfield_context** is a **dictionary**. FulfillLineDetailEffB**ContextName**privateVO

where

- **dictionary** identifies the name of the dictionary that your rule references.
- DOO_FULFILL_LINE_DTLS_ADD_INFO is the category for each fulfillment line detail.
- **ContextName** identifies the name of the context.

Your rule must establish a hierarchy.

Parent Pattern	Property	Child Pattern
my_fline_details	.`FulfillLineDetailEffCategories`	**my_fline_details** is **my_ fline_details_extensible_flexfied_category**
my_ fline__details_extensible_flexfied_category	.`FulfillLineDetailEffB` **ContextName**privateVO	**my_fline__details_extensible_flexfied_category** contains **my_fline_details_extensible_flexfied_context**

Related Topics

- Overview of Setting Up Extensible Flexfields in Order Management
- Overview of Using Extensible Flexfields in Order Management

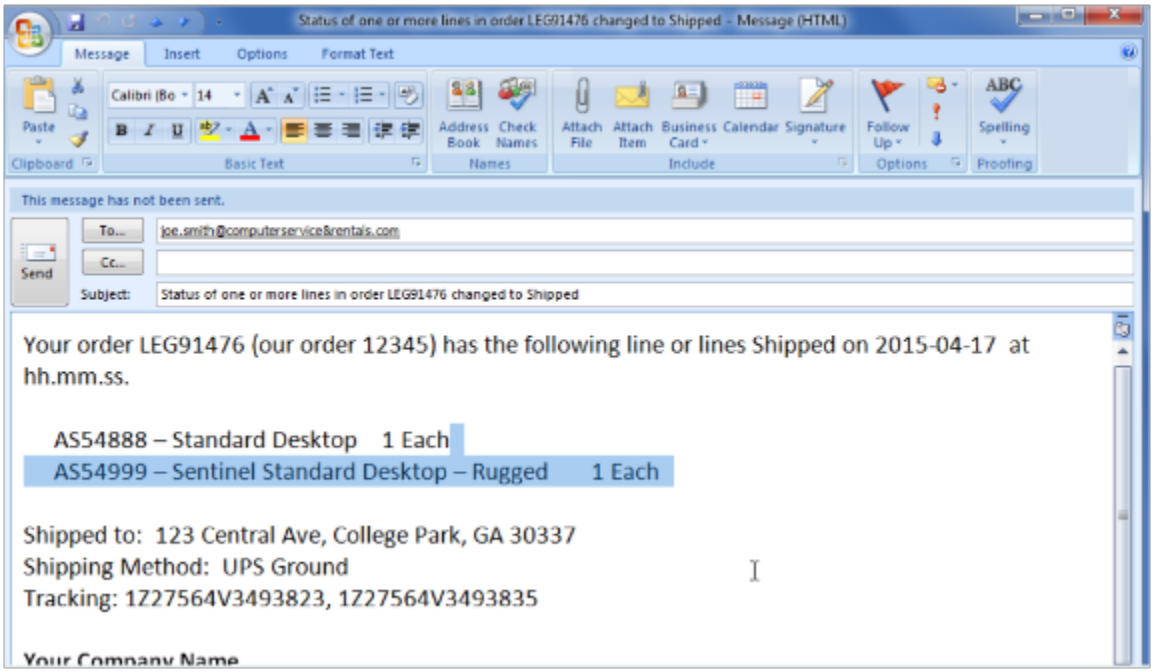
11 Email, Reports, and Attachments

Email

Administer Email Messaging in Order Management

Set up Order Management to send an email message when a business event happens, such as when the sales order status changes to Shipped.

Assume you must format the email that Order Management sends when the sales order status changes to Shipped.



Do it.

1. Go to the Setup and Maintenance work area, then go to the task.
    - Offering: Order Management
    - Functional Area: Orders
    - Task: Manage Business Event Trigger Points
2. Optional. Send an email notification when the sales order goes into hold.
    - On the Manage Business Event Trigger Points page, click the **Hold** row, then make sure the **Active** option in this row contains a check mark.
    - In the Details area, add a check mark to the **Send Email Notification** option.

3. Optional. Send an email notification when the order header status updates.
    - On the Manage Business Event Trigger Points page, click the **Order Header Status Update** row, then make sure the **Active** option in this row contains a check mark.
    - In the Details area, add a check mark to the **Raise Event** option and the **Send Email Notification** option for each Status Value, as necessary.
      For example, to send an email notification when the order status changes to Closed, add a check mark to the options in the Closed row.
      The Send Email Notification option depends on the event, so you must make sure you add a check mark to each option.
    - Repeat this step for other status values, as necessary.

Administer Email Messaging for Status Updates on Fulfillment Lines

Administering Order Management to send an email notification when the status updates on a fulfillment line requires that you modify the orchestration process definition.

1. On the Manage Business Event Trigger Points page, click the **Fulfillment Line Status Update** row, then make sure the **Active** option in this row contains a check mark.
2. Click **Save and Close**.
3. On the Setup page, search for, then open Manage Orchestration Process Definitions.
4. On the Manage Orchestration Process Definitions page, search for ShipOrderGenericProcess.
   Each orchestration process controls the status value for each fulfillment line, so you must modify the orchestration process that controls the status value. In this example, you modify the orchestration process that controls the shipping status value.
5. In the Search Results, click the **row** that contains ShipOrderGenericProcess, then click **Actions > Edit**.
6. In the Process Details area, click **Status Conditions > Fulfillment Line Status Values > Edit Status Rule Set**.
7. On the Edit Status Rule Set page, add a check mark to the **Notify External System** option and the **Send Email Notification** option for each Status Value where you must send a notification.
   For example, to send an email notification when the fulfillment line status changes to Shipped, add a check mark to the options in the Shipped row.
   The Send Email Notification option depends on the Notify External Systems event, so you must make sure you add a check mark to each option.
8. Repeat step 7 for other status value, as necessary.
   Order Management will send an email message when each fulfillment line that references this orchestration process definition reaches the status that you specify in steps 7 and 8. Order Management sends this email to the customer contact and to the ship-to contact that the sales order references.
9. Repeat steps 4 through 8 for each orchestration process definition in your deployment that updates status values.

Guidelines

If your message includes party data, such as the Ship-to Contact, then make sure you set up the contact on the customer account. For details, see *Overview of Displaying Customer Details on Sales Orders*.

Make sure the attribute that you're using as part of the message contains a value on the sales order. For example, if you're sending a message to the Ship-to Contact for order 768496, then make sure the Ship-to Contact on order 768496 contains a value.

Related Topics

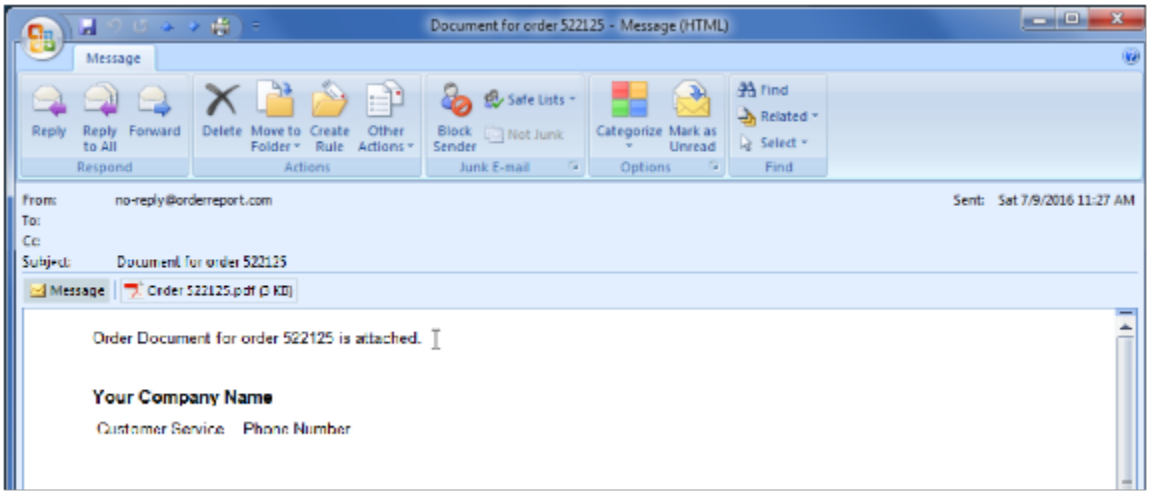
- Overview of Displaying Customer Details on Sales Orders

Administer Email Format in Order Management

Set up the format that Order Management uses when it sends an email message.

The Order Entry Specialist can click **Actions > Create Document > Send Email** from the sales order header to send order details through an email message. Order Management sends order details in a PDF (Portable Document Format) file that it attaches to the email.

Here's the predefined format that Order Management uses.



Note

- Use Oracle Analytics Publisher to modify format, such as add your company name and logo, or modify the template message text.
- Use the From Address for Email Messages parameter to specify the value that Order Management displays in the email From address. For details, see *Manage Order Management Parameters*.

In this example, modify the email template that Order Management uses for a status update.

1. Go to the Reports and Analytics work area.

   For details, see *Overview of Transactional Business Intelligence*.
2. On the Reports and Analytics page, click **All Folders > Shared Folders**.
3. Click **Supply Chain Management > Order Management > Fulfillment Alerts**.

   Notice that the work area displays several templates, such as Hold Notification, Sales Order Line Status Update, and Sales Order Status Update Notification.
4. Click **Sales Order Status Update Notification**.
5. On the Fulfillment Alerts page, click **Actions > Edit Report**.
6. Under SalesOrderStatusUpdateNotificationLayout, click **Edit**.
7. Modify the layout and save.

Related Topics

- Manage Order Management Parameters
- Overview of Creation and Administration of SCM Analytics and Reports

Reports

Use Reports and Analytics with Order Management

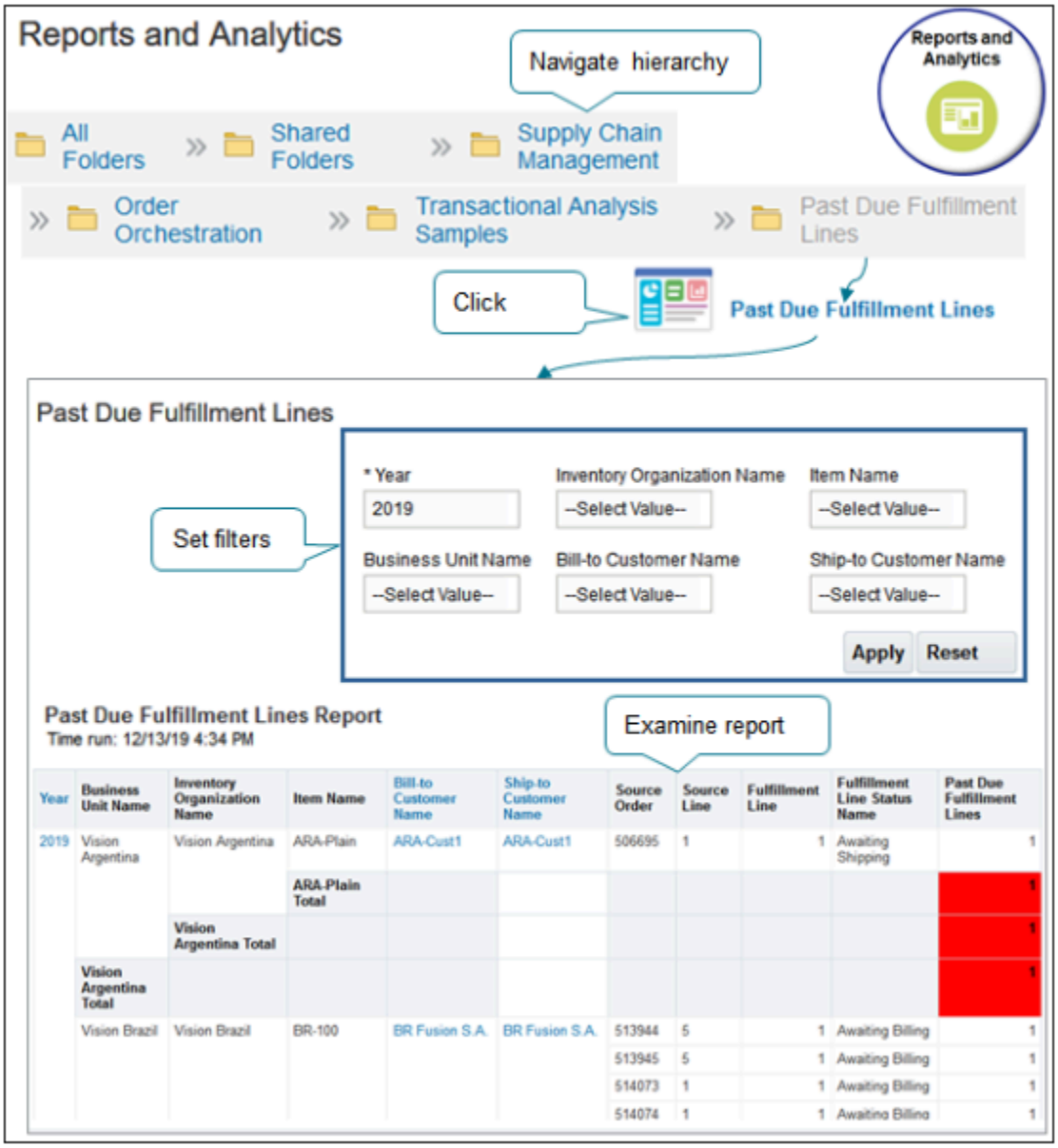
Use the Reports and Analytics work area to get detailed reports for some aspects of Order Management.

Create a report that includes various types of data.

- Draft sales orders
- Order header attributes
- Order line attributes
- Fulfillment line attributes
- Pricing details
- Sales credits
- Return orders
- Coverage and subscriptions

Examine a Predefined Report

Examine a report that includes fulfillment lines that are overdue.



Try it.

1. Go to the Reports and Analytics work area.
2. On the Reports and Analytics page, click **All Folders > Shared Folders**.
3. Click **Supply Chain Management > Order Orchestration > Transactional Analysis Samples > Past Due Fulfillment Lines**, then click the **Past Due Fulfillment Lines** link.
4. Wait for the report to build, then examine the report output.
5. Modify search parameters to filter the report, then notice the results.
   For example, set Bill-to Customer Name to a value and click **Apply**.

Create Your Own Report

Assume you need a report that displays order lines that have shipped.

1. Make sure you have the privileges that you need to create a report.

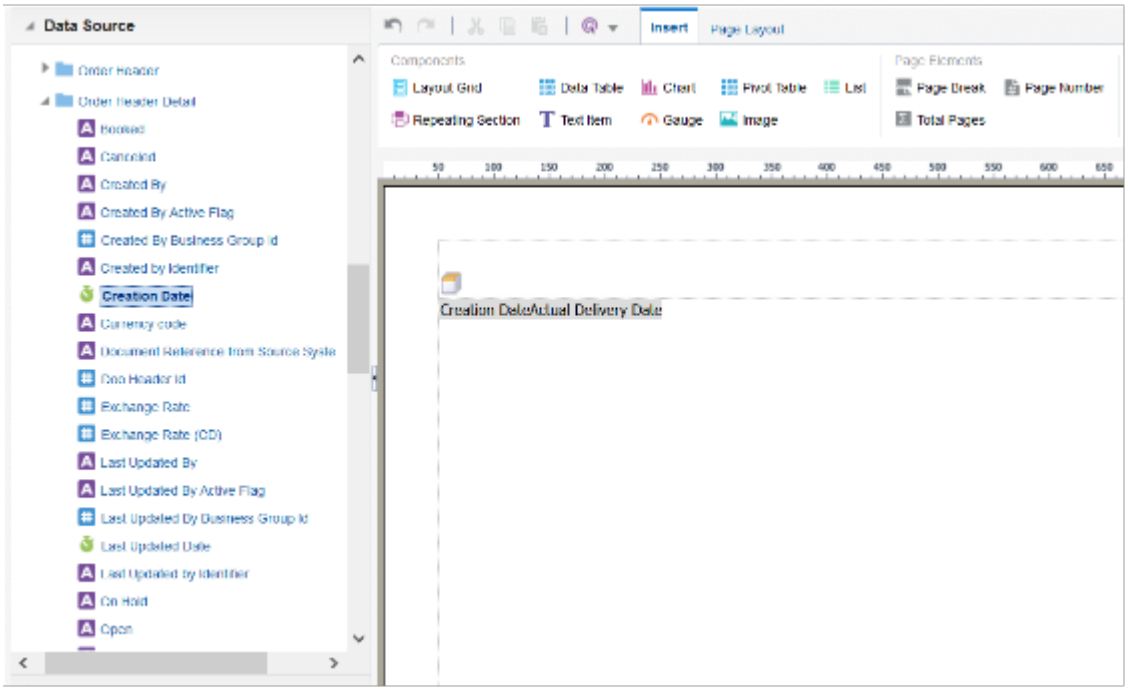
2. On the Reports and Analytics page, click **Create > Report**.

   The Oracle Transactional Business Intelligence application opens and displays.
3. In the Create Report dialog, click **Use Subject Area**.
4. Set the **Subject Area** to Order Management Order Lines Real Time.

   For details, see *Subject Areas for Transactional Business Intelligence in SCM*.
5. Click **Use Report Editor > Finish**, then save the file.

   You can use any name. For this example, save the file name as My Real Time Fulfillment Report.
6. On the My Real Time Fulfillment Report page, click **Header and Footer, Portrait**.
7. Drag and drop each element from the Data Source tree onto the editor.
   - In the Data Source tree, expand **Order Header Detail**, drag, and then drop **Creation Date** from the tree onto the editor.
   - In the Data Source tree, expand **Fulfillment Line Details**, drag, and then drop **Actual Delivery Date** from the tree onto the editor.

   Continue until the editor resembles this layout.



8. Click **Save Report**.
9. Navigate back to the Reports and Analytics page, then click **Browse Catalog**.
10. On the Catalog page, under My Real Time Fulfillment Report, click **Open**, then examine the report output.

Add the Primary Salesperson to a Report

Make sure you add the SalespersonName attribute to the report.

Assume you create your own sales order report and add the sales credit attribute in the report's header. You find that the report displays data for the salesperson when you run the report in the Reports and Analytics work area, but not when you use the Order Management work area, such as when you go to the Manage Orders page, then click Actions > Create Document > View.

This might happen if you add the sales credit attribute to the report but not the SalespersonName attribute. Make sure you add SalespersonName.

Limitations

If you use a web service to create a data model, then you can't burst the report. To burst means to separate data into sections, create a separate document for each section, then deliver each document to one or more destinations.

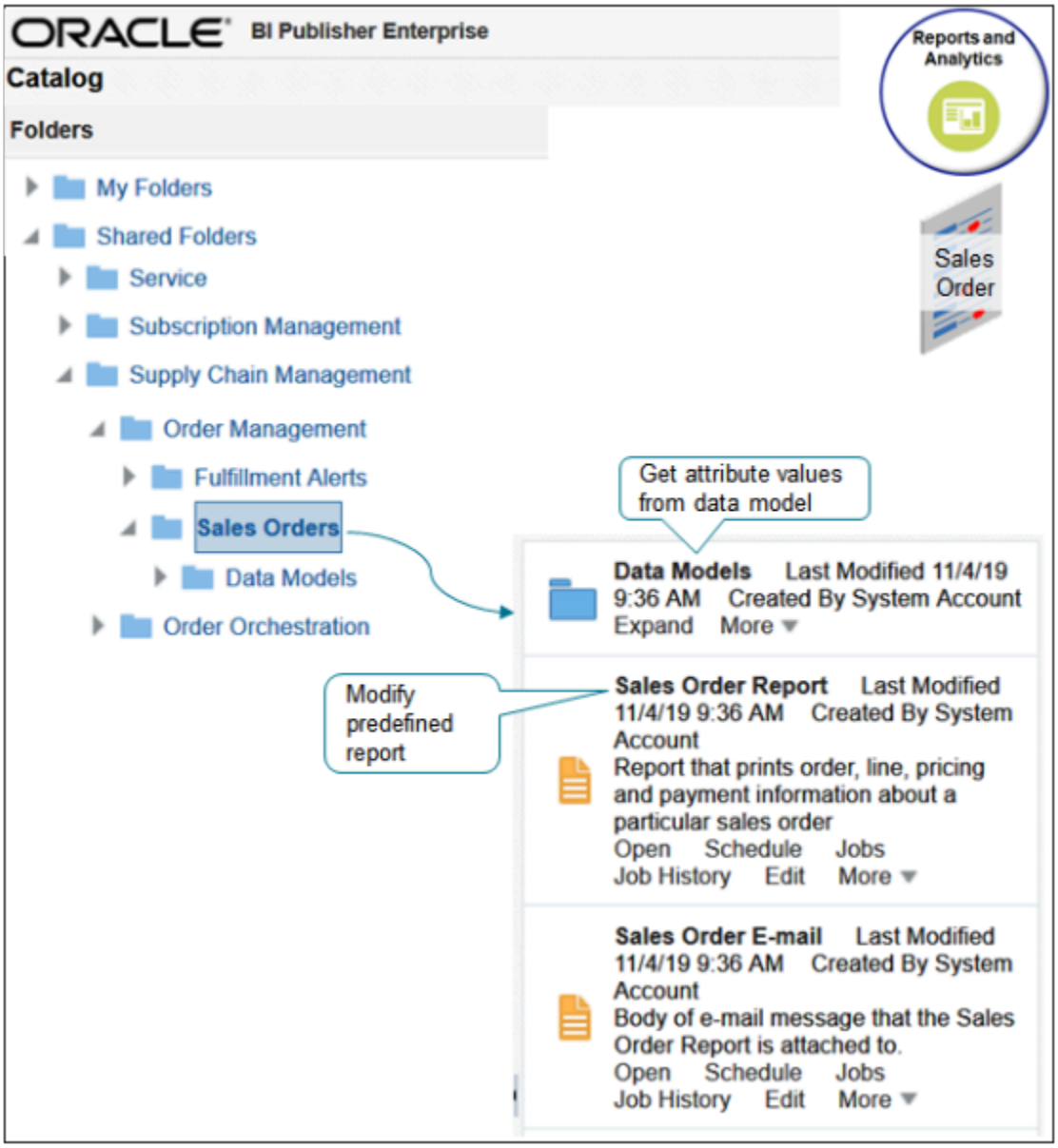
Learn more about what you can with reports. For details, go to *Creating and Administering Analytics and Reports for SCM*, then search for Overview of Business Intelligence Publisher.

Related Topics

- Administer Email Messaging in Order Management

Modify Report Templates

The Reports and Analytics work area comes predefined with several reports that you can use for Order Management. You can modify a template to meet your needs.



Note

- The reports come predefined as rich text file (RTF) templates.
- Use Oracle Analytics Publisher to edit them.
- Use a data model to get attribute values.
- Use the predefined Sales Order Report template as a starting point.
- For details about how to add a flexfield to a report, see *Guidelines for Setting Up Extensible Flexfields in Order Management*.

Try it.

1. Install Oracle Analytics Publisher Desktop.
    - Go to *Oracle Analytics Publisher Downloads*.
    - Click **Oracle Analytics Publisher Desktop 12.2.1.4.0 for 32 bit Office on Windows**, then wait for the file to finish downloading.
    - Follow the instructions to install Oracle Analytics Publisher Desktop.
2. Sign into Oracle Business Intelligence Enterprise Edition with the privileges that you need to edit predefined reports.

    For details, see:
    - *Overview of Transactional Business Intelligence*
    - *How You Access and Modify Report Components*
3. Go to the Reports and Analytics work area.

    Some implementations might take you directly to Oracle Analytics Publisher instead of to your Home page. If that happens, skip to the step where you click **Catalog** later in this procedure.
4. On the Reports and Analytics page, click **Browse Catalog**.

    Oracle Analytics Publisher opens in a new tab on your browser.

    On the new tab, make sure the page displays Oracle Analytics Publisher in the upper left corner. If it displays Oracle Transactional Business Intelligence or Oracle Business Intelligence, then sign out, and sign into the correct instance of Oracle Applications. Ask your reports administrator for details.
5. Click **Catalog**.
6. On the Catalog page, in the Folders area, expand **Shared Folders > Supply Chain Management > Order Management > Sales Orders**.
7. In the list area to the right of the Folders area, in the Sales Order Report row, click **More > Customize**.

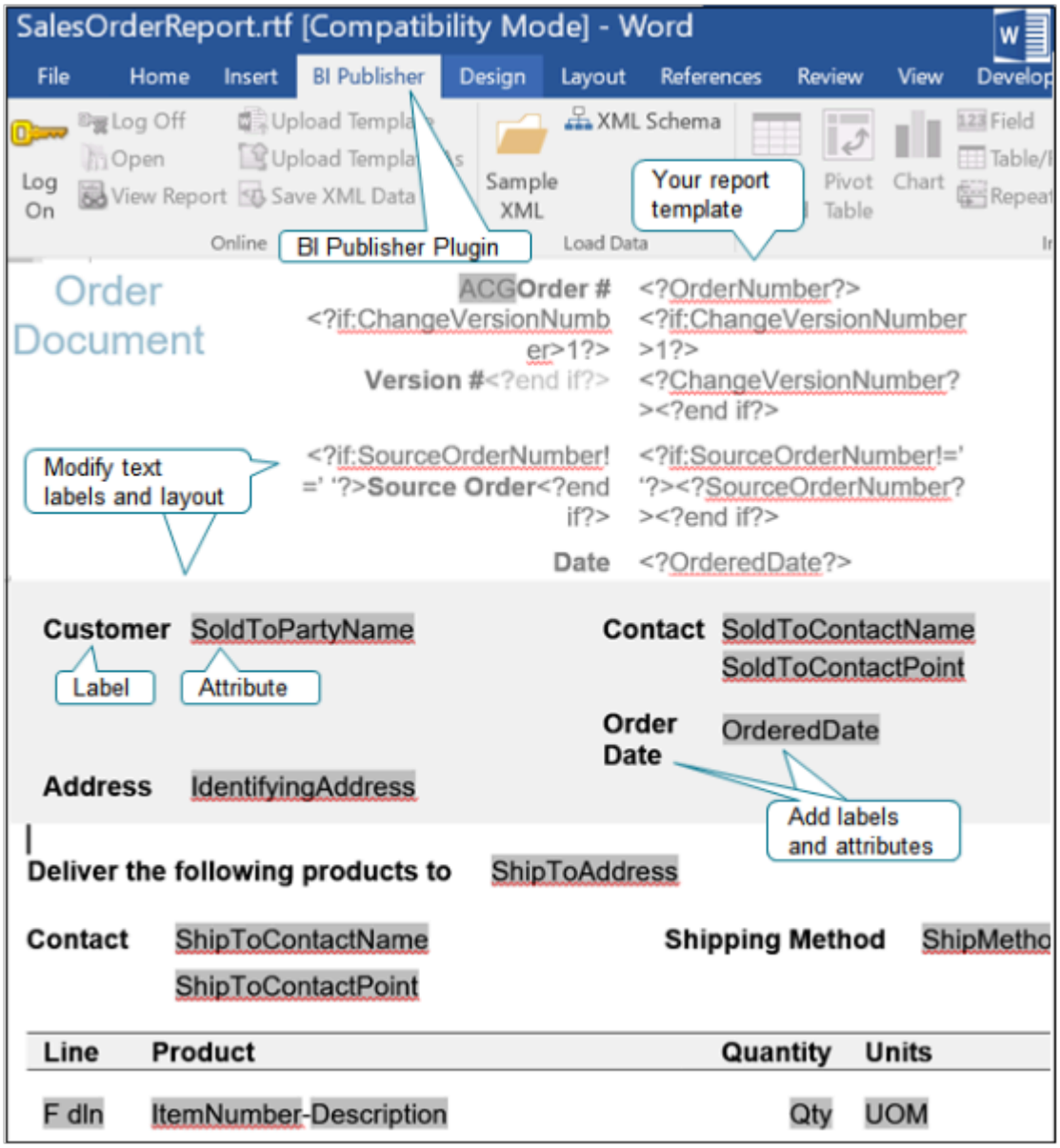
    The publisher opens the report for editing. For an example, see *How You Modify Copies of Predefined Reports in Manufacturing*.
8. On the Sales Order Report page, click **Save Report**.

    The publisher creates a copy of the report in the `Shared Folders/Custom/Supply Chain Management/Order Management/Sales Orders` folder. You will use this copy later.
9. Click **Edit**.
10. In the dialog that displays, select the **Open with Microsoft Word** option, then click **OK**.

    Microsoft Word opens.

11. In Microsoft Word, name the file with the rtf extension, such as SalesOrderReport.rtf, then save the file to a handy location. For this example, save it to `c:\`.

    Microsoft Word displays your report template.



Note

- The Oracle Analytics Publisher plugin displays the Oracle Analytics Publisher tab.
- The predefined report template includes a meaningful layout, labels, and attributes. You can use it to modify layout and text labels.
- Each label displays as bold text. For example, Customer is a label.
- Each attribute displays with a dark grey background. For example, SoldToPartyName is an attribute.

- You can add labels and attributes to the predefined template. For example, you can add the Order Date label and the OrderedDate attribute.

12. Modify your report.

    - Change the Contact label. Click the **Contact** label, and start typing.

Old Label	New Label
Contact	Sold-to Contact

    - To move the SoldToContactPoint attribute down one line, place your cursor at the beginning of SoldToContactPoint, then press the Enter key.
    - Click **Save**.

      Don't edit the attribute text. The attribute text reflects the data model text and it helps you to see the relationship between the model and your template.

13. Upload your changes.

    - Go back to Oracle Analytics Publisher in your browser.
    - Click **Catalog**, expand **Shared Folders > Custom > Supply Chain Management > Order Management**, then click **Sales Orders**.
    - In the list that contains Sales Order Report, click **Edit**.
    - On the Sales Order Report page, click **Add New Layout**.
    - In the Upload or Generate Layout area, click **Upload**.
    - In the dialog that displays, enter values, then click **Upload**.

Attribute	Value
Layout Name	My Sales Order Report
Template File	SalesOrderReport.rtf Browse to the rtf file that you saved on your computer. For this example, its `C:\SalesOrderReport.rtf`.
Type	RTF Template
Locale	English

    - Click **Save Report**.

Add Attributes to Your Template

What if the template doesn't include the attributes you need? You can add them from the data model.

Summary of the Set Up

1. Download the data model.
2. Modify your template.
3. Upload your template.

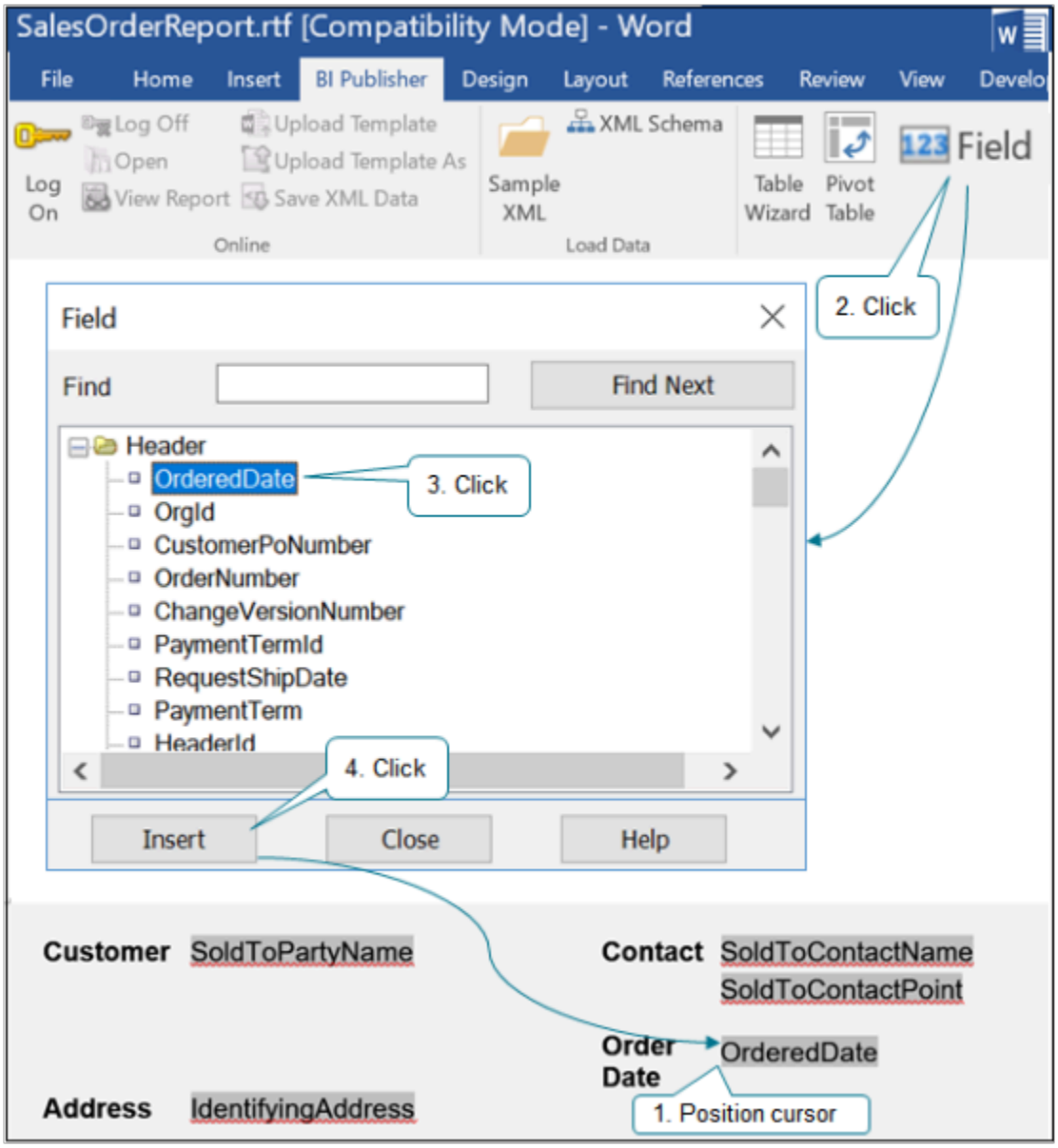
Assume you need to add the OrderedDate attribute.

Download the Data Model

1. Go to Oracle Analytics Publisher, then go to **Shared Folders > Supply Chain Management > Order Management > Sales Orders > Data Model**.
2. In the list area to the right of the Folders area, in the row that contains Sales Order Report Data Model, click **Edit**.
3. In the Data Model area, click **Properties**.
4. In the Properties area, click **SalesOrderReportDm.xml**, then save the file to the same location where you saved your report template.
    - For this example, save it to `c:\`.
    - Change the file name extension from xdm to xml name when you do the save.

Attribute	Value
File Name	_Supply Chain Management_Order Management_Sales Orders_Data Models_ SalesOrderReportDm.xml

Modify Your Template



Try it.

1. Go to your template in Microsoft Word.
2. Add the Ordered Date label to the template.

   Examine the screen print earlier in this topic to see where to add it. Place your cursor after the Contact label, enter a few carriage returns, then start typing in the label.
3. Position your cursor at the end of the SoldToContactPoint attribute, then enter a few carriage returns until the cursor aligns with the label.
4. In the Word ribbon, click the **Oracle Analytics Publisher** tab, then click **Sample XML**.
5. In the dialog that displays, select **_Supply Chain Management_Order Management_Sales Orders_Data Models_SalesOrderReportDm.xml**, then click **Open**.
6. Click the **Oracle Analytics Publisher** tab then, in the Insert area, click **Field**.

   The Field dialog displays the attributes from your data model.

7. In the Field dialog, click **OrderedDate**, then click **Insert**.
8. Click **File > Save**, then save the file as SalesOrderReport_1.rtf.
   You create a new file to avoid an overwrite conflict when you upload the template.

Upload Your Template

1. Go back to Oracle Analytics Publisher in your browser.
2. Click **Catalog**.
3. On the Catalog page, in the Folders area, expand **Shared Folders > Custom > Supply Chain Management > Order Management**, then click **Sales Orders**.
4. In the row that contains the sales order report you created earlier in this procedure, click **Edit**.
5. On the Sales Order Report page, click **Add New Layout**.
6. In the Upload or Generate Layout area, click **Upload**.
7. In the Upload Template File dialog, set the values, then click **Upload**.

Attribute	Value
Layout Name	Enter any value. For this example, enter `My Layout for Sales Order Report`.
Template File	SalesOrderReport_1.rtf Locate the SalesOrderReport.rtf file on your local computer. Its the one you edited in Microsoft Word.
Type	RTF Template
Locale	English

8. Select the data model.
   - On the Sales Order Report page, Next to Data Model, click **Select Data Model**.
   - In the Select Data Model dialog, expand **Shared Folders > Supply Chain Management > Order Management > Sales Orders > Data Model**.
   - Click **Sales Order Report Data Model**, then click **Open**.
9. On the Sales Order Report page, click **View Report**, then click the **My Layout for Sales Order Report** tab.
   The report displays with example data in the attributes.
10. Examine the report. Verify that it includes the changes you made in Microsoft Word, such as the Ordered Date.
11. Verify your changes.
    - Make sure you have the privileges that you need to manage sales orders. In particular, make sure you have the FOM_PRINT_ORDER_PRIV_OBI privilege. It allows you to use the Create Document action. For details, see *Privileges That You Need to Implement Order Management*.
    - Go to the Order Management work area, then create a sales order.
    - Add values to attributes on the order header, then add an item to an order line.

- On the Create Order page, click **Actions > Create Document > View**.
- In the Document dialog, set Template to one of the layouts you created in Oracle Analytics Publisher.
  - My Sales Order Report
  - My Layout for Sales Order Report
- Verify that the report includes the changes you made in Microsoft Word, such as the Ordered Date.

Related Topics

- How You Access and Modify Report Components
- Overview of Transactional Business Intelligence
- How You Modify Copies of Predefined Reports

Attachments

Overview of Integrating Attachments in Order Management

Integrate Order Management so it can receive an attachment as part of a source order from a source system, then send it to your order fulfillment system.

- Your users can use the Order Management work area add an attachment to a sales order, such as a document that includes requirements for manufacturing, a memo that includes guidelines for negotiating a price, or a URL to a page that includes item installation instructions.
- Order Management can attachments to your fulfillment system.
- Order Management can receive an attachment from a source system and include it as part of a sales order.
- Order Management can't receive an attachment from a fulfillment system.
- Order Management can't send an attachment to an order capture system.

You can map attachment attributes from a source order that resides on your source system, to the enterprise business message (EBM) in Order Management. Order Management then uses the connector to send the attachment to your order fulfillment system.

How it Works

Here's how Order Management processes an attachment it receives from your source system.

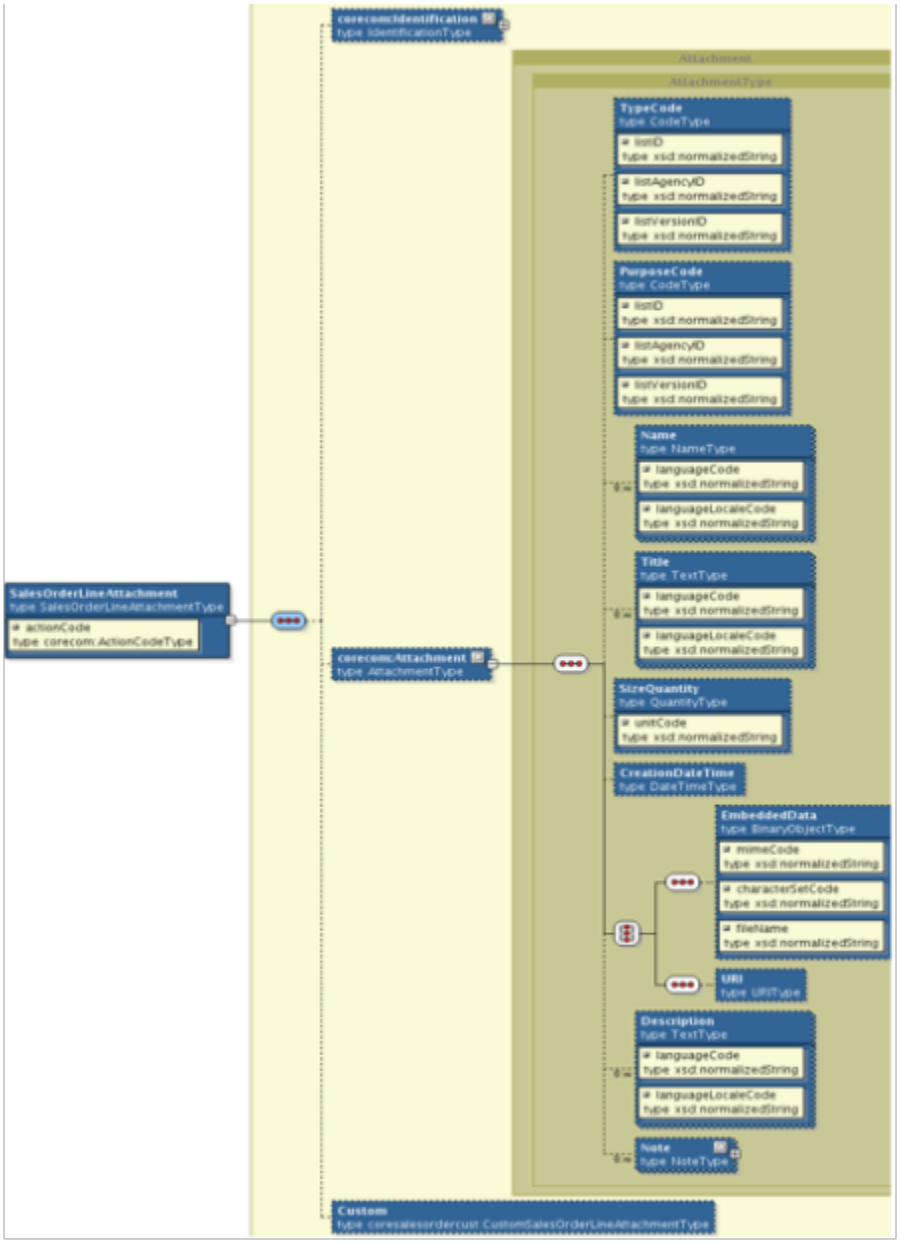
1. Uses an enterprise business object (EBO) to get each attachment from the source order.
2. Cross-references each value from the source system to a value that Oracle Applications use. This approach provides a single representation of the attachment in the source system and in Oracle Applications.
3. Converts the source order to a sales order, then.
   - Adds attachments from the source order header to the sales order header
   - Adds attachments from source order lines to sales order lines
     The sales order includes fulfillment details according to order attributes and product transformation rules. Learn how Order Management does the mapping. For details, see *Overview of Importing Orders Into Order Management*.

4. Sends a fulfillment request to your fulfillment system.

If the source system sends a change to an attachment or a source order, such as adding or deleting an attachment or modifying an existing attachment, and if Order Management already imported the source order, then Order Management updates the sales order so it reflects the change. Order Management can't send an attachment to the source system, so it isn't necessary to map attachment attributes to a format that the source system understands when you transform the response that the web service sends to the source system.

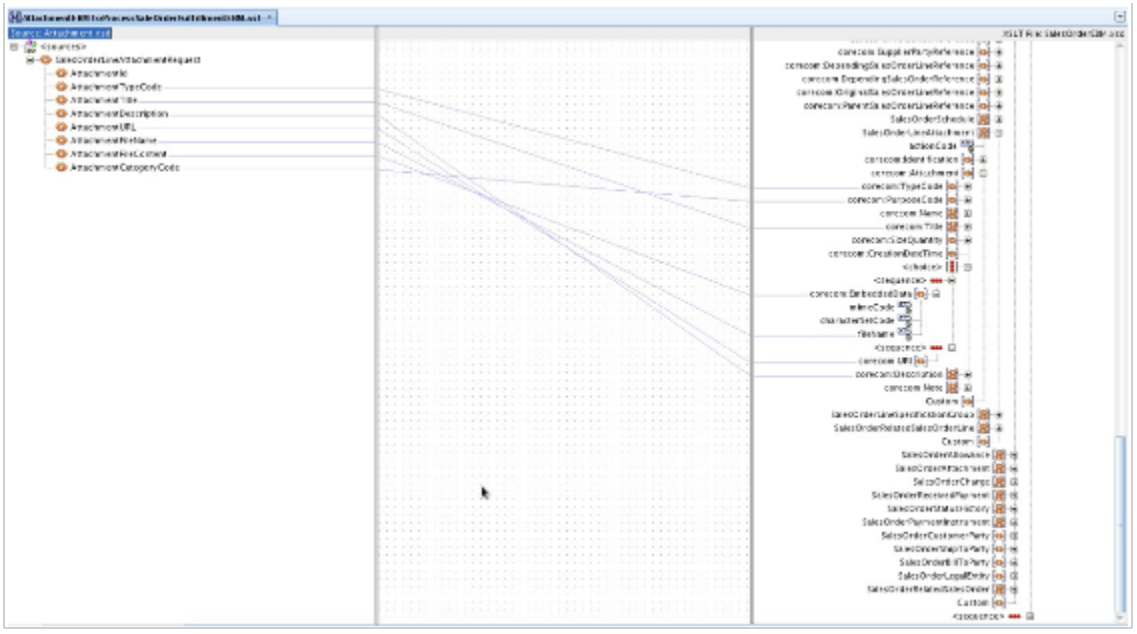
Map Attachments in Your Source System to Attachments in Order Management

Order Management uses the Sales Order enterprise business object to communicate with your source system. The attachment is part of the sales order. Here's the structure it uses.



Map Attachments in Order Management to Attachments in Your Fulfillment System

Order Management uses the ProcessSalesOrderFulfillmentEBM enterprise business message (`EBM`) to create, change, or cancel a sales order. Here's the structure that the message uses to map an attachment from the XSD (XML Schema Definition) for the sales order in Order Management to your fulfillment system.



Mapping Attachment Attributes

Here are the sections where the attachment attributes reside in processSalesOrderFulfilmentEBM.

Section	Description
SalesOrderAttachment	Attachments on the order header.
SalesOrderLineAttachment	Attachments on the order line.
SalesOrderLineScheduleAttachment	Attachments on the order line schedule.

ProcessSalesOrderFulfillmentEBM includes attachment attributes.

Attribute	Path	Description
`TypeCode`	`coreoom:Attachment/coreoom:TypeCode`	The type can be File, Text, or URL.
Title	`coreoom:Attachment/coreoom:Title`	Attachment name.
Description	`coreoom:Attachment/ coreoom:Description`	Details about the attachment.

Attribute	Path	Description
`URI`	`coreoom:Attachment/coreoom:URI`	If Type is URL, then `URI` contains the URL that locates the document that contains the attachment content.
FileName	`coreoom:Attachment/ coreoom:EmbeddedData/FileName`	If Type is File, then FileName contains the name of the file that contains the attachment content.
`EmbeddedData`	`coreoom:Attachment/ coreoom:EmbeddedData`	If Type is File, then `EmbeddedData` contains the file content. If Type is Text, then `EmbeddedData` contains the text that the user entered for the attachment. If the source order doesn't include the attachment content, then you must get this content before you transform the source order into ProcessSalesOrderFulfillmentEBM.

Order Management maps attributes.

Attribute	Description
Type	The type can be File, Text, or URL.
Title	The attachment name.
Description	Details about the attachment.
Document Content	Document Content depends on the value of the Type attribute. • **URL**. Document content is the URL. • **File**. Document content is the contents of the file. • **Text**. Document content is the text that the user enters.

Map Attachments in the Connector to Your Order Fulfillment System

Order Management sends a fulfillment request to your fulfillment system to do order fulfillment tasks.

- The order fulfillment system can access the attachment through the Orchestration Order Attachments Service web service.
- The GetAttachment method of this web service gets the attachments that the order header or fulfillment line references.
- The order fulfillment system uses GetAttachment to get attachments and map attachment attributes through the connector to the fulfillment system.

Use parameters in the input payload for the GetAttachment method.

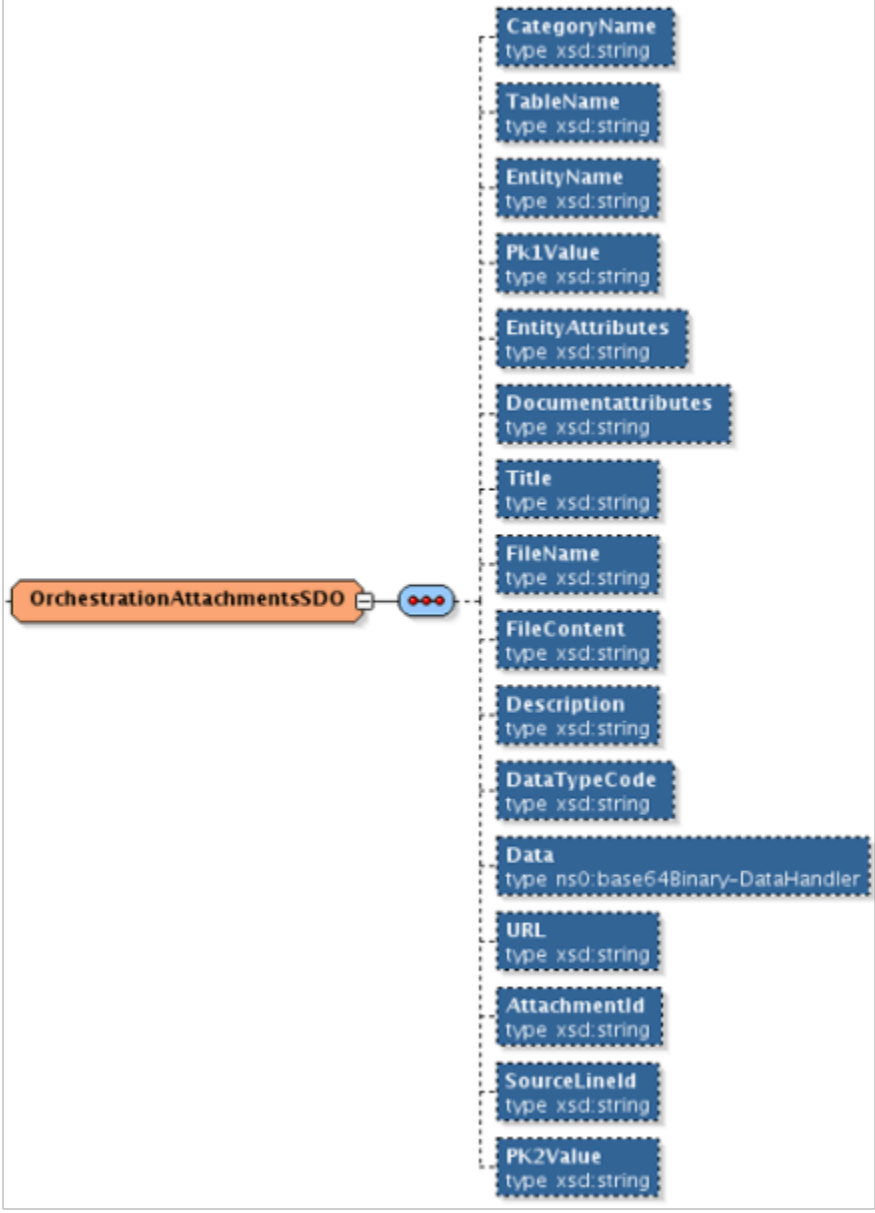
Parameter	Description
`SourceEntityName`	Name of the entity that the attachment references.
`SourceTableName`	Name of the database table for the entity that the attachment references.
`SourcePK1Value`	First value that identifies the entity that the attachment references.
`SourcePK2Value`	Second value that identifies the entity that the attachment references.

Here are the values you use for each parameter.

- Use DOO_HEADERS_ALL for the order header.
- Use DOO_FULFILL_LINES_ALL for the fulfillment lines.

Use these values for each parameter except for SourcePK2Value. You don't use either value for SourcePK2Value.

The GetAttachment method returns the OrchestrationAttachmentSDOs as output, which contains attachment details.



Get details about the service path, abstract WSDL URL, and so on. See the Service Data Objects section of *SOAP Web Services for Oracle SCM Cloud*.

Related Topics

- Integrate Attachments in Order Management
- Overview of Integrating Order Management with Other Oracle Applications
- Manage Attachments on Order Lines

Integrate Attachments in Order Management

Use transformation and web services to integrate attachments in Order Management.

1. Allow Order Management to receive sales order attachments from an order capture system. Collect the attachment category during orchestration data collection.
2. Transform the sales order, including attachment attributes, into ProcessSalesOrderFulfillmentEBM.

   This step transforms the Order Management message, including attachment attributes, into a format that the order fulfillment system can process. Note that Order Management doesn't accept attachments from the order fulfillment system, so it isn't necessary to map attachments from the order fulfillment system message into a format that Order Management can use in the response from the order fulfillment system.
3. Allow Order Management to send sales order attachments to your fulfillment system. Call the AttachmentsAM web service and use it to select and send attachments according to the type of fulfillment request and the attachment category.
4. Call the SalesOrderOrchestrationService web service.

   Specify the mapping in the request body to SalesOrderOrchestrationService. For example, here's a request that get the attachments that one fulfillment line references.

```
<soap:Body>
 <ns1:getAttachment xmlns:ns1="http://xmlns.oracle.com/apps/scm/doo/common/attachments/model/types/">
 <ns1:dooGetParameters xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/common/attachments/model/">
 <ns2:sourceEntityName>DOO_FULFILL_LINES_ALL</ns2:sourceEntityName>
 <ns2:sourcePK1Value>100000045762 </ns2:sourcePK1Value>
 <ns2:sourceTableName>DOO_FULFILL_LINES_ALL</ns2:sourceTableName>
 </ns1:dooGetParameters>
 </ns1:getAttachment>
</soap:Body>
```

5. Make a call in the to the order fulfillment system to get the attachments that each fulfillment line references and the corresponding order header.

Related Topics

- Overview of Integrating Attachments in Order Management

12 Maintain and Troubleshoot

Tools and Techniques

Get Help From My Oracle Support

Use the My Oracle Support website at support.oracle.com to troubleshoot a range of problems.

Here are some articles you might find useful. Each of the master notes organize and references a number of other support notes. Use the master notes as a starting point to troubleshoot problems.

Article	Description
Master Note, Order Promising and Collections (Doc ID 2649499.1)	Fix problems with promising, availability, collections, and so on.
Master Note, Transforming Data on Sales Orders Using Transformation Rules (Doc ID 2675921.1)	Fix problems with transformation that aren't already described in *Transformation Rules*.
Master Note, Importing Data using FBDI (Doc ID 2665940.1)	Fix problems with import that aren't already described in the chapter that starts with *Overview of Importing Orders Into Order Management*.
Master Note, Applying and Releasing Holds (Doc ID 2000460.1)	Fix problems with applying and releasing holds on sales orders.
Master Note, Shipping Set Up, Issues and Usage. (Doc ID 2731251.1) *Master Note, Index, Shipping. (Doc ID 2731268.1)*	Fix problems that happen with shipping in downstream fulfillment.
Order Management, Common Errors Raised (Doc ID 2260134.1)	Take action depending on the error you encounter.
Progress Fulfillment Lines Where Status is Awaiting Billing (Doc ID 2322271.1)	Learn how to fix lines that are stuck in the Awaiting Billing status.
Progress Fulfillment Lines Where the Line is Locked (Doc ID 2317580.1)	Learn how to unlock lines that are locked, depending on whether the status is Not Started, Scheduled, Reserved, Shipped, or Awaiting Billing.

Use SQL to Query Order Management Data

Use SQL to get data from the Order Management database, then analyze it.

Get details from the Order Management database to do a variety of administration tasks.

- Verify the data format and values you must use during order import or with a web service.
- Troubleshoot problems that happen during upgrades or other administrative set up.
- Save data into a data management tool of your choice so you can analyze it.

You create a data model in Oracle Business Intelligence and use it to query the Order Management database.

Assume you must query the database to get a list of users so you can identify the users who are active and the ones who aren't.

Summary of the Steps

1. Create and run query.
2. Create report.

This topic uses example values. You might need different values, depending on your business requirements.

Create and Run Query

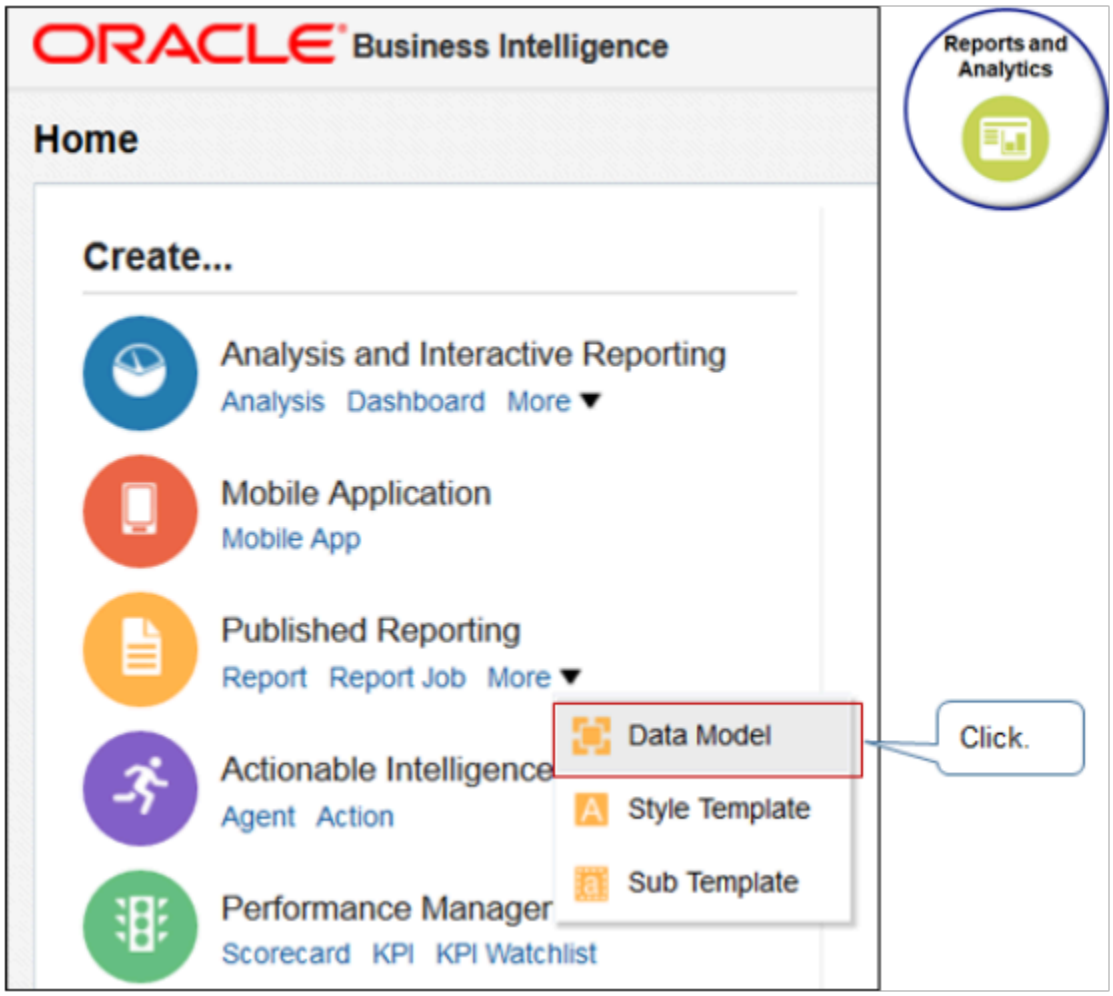
1. Make sure you have the privileges that you need to develop in Oracle Analytics Publisher.

   You need these privileges so you can create a data model in Oracle Business Intelligence.
2. Go to the Reports and Analytics work area.

   For details, see *Use Reports and Analytics with Order Management*.
3. On the Reports and Analytics page, click **Browse Catalog**.

   Your browser opens a new tab for Oracle Business Intelligence.
4. In Oracle Business Intelligence, click **Home**.

5. Create the data model.
    - In the Create area, under Published Reporting, click **More > Data Model**.



You use a data model to set up the SQL you use to query the Oracle database. You can reuse the data model for different SQL queries. It isn't necessary to create and save a separate data model for each SQL query.

- On the Diagram tab, click **New Data Set > SQL Query**.
- In the New Data Set SQL Query dialog, set the values.

Attribute	Value
Name	My SQL
Data Source	ApplicationDB_FSCM (Default) FSCM means Fusion Supply Chain Management.

Attribute	Value
Type of SQL	Standard SQL

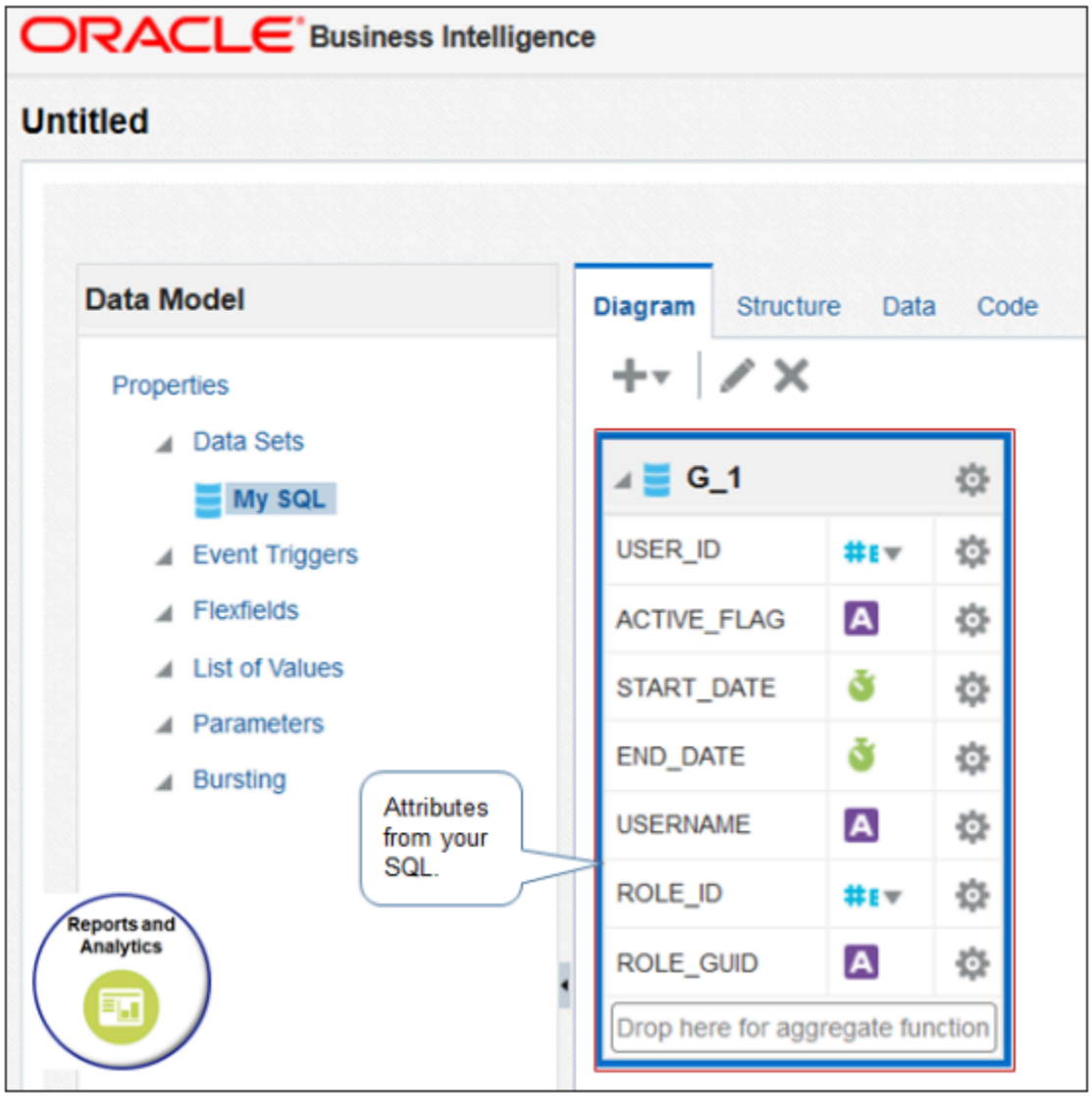
SQL Query	`Select pu.user_id, pu.active_flag, pu.start_date, pu.end_date, pu.username, pur.role_id, pur.role_guid`
	`Select pu.user_id, pu.active_flag, pu.start_date, pu.end_date, pu.username, pur.role_id, pur.role_guid`
	`from per_users pu, per_user_roles pur, per_user_history puh`
	You can enter some other query. For example, to get all order headers, enter this:
	`SELECT * FROM fusion.DOO_HEADERS_ALL`
	To get all order lines, enter this:

Attribute	Value
	`SELECT * FROM fusion.DOO_LINES_ALL`

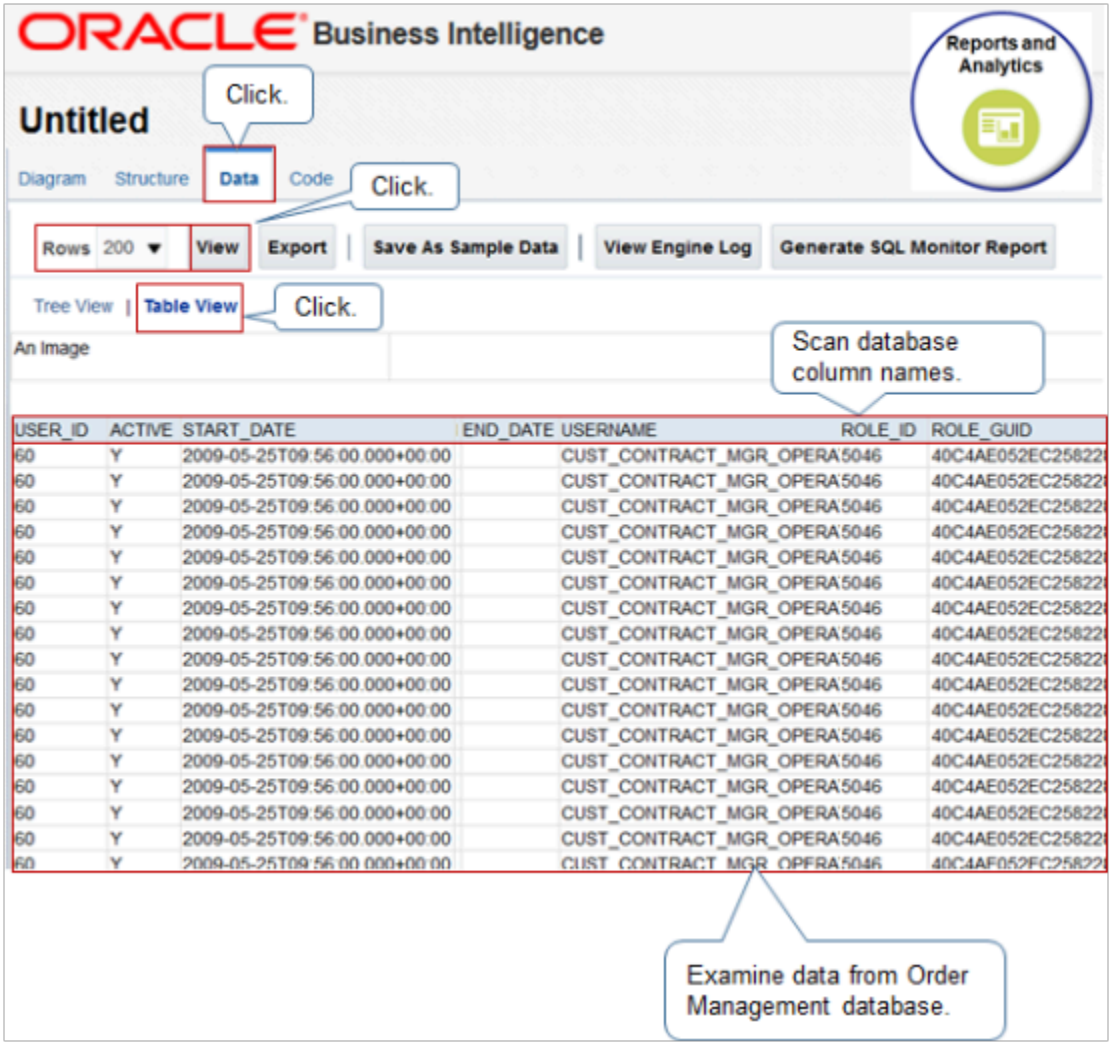
For example:

- Click **OK**.

  The Diagram tab displays the attributes that your SQL defined.



- Click **View Data**, set Rows to 200, click **View**, click **Table View**, then examine the output.



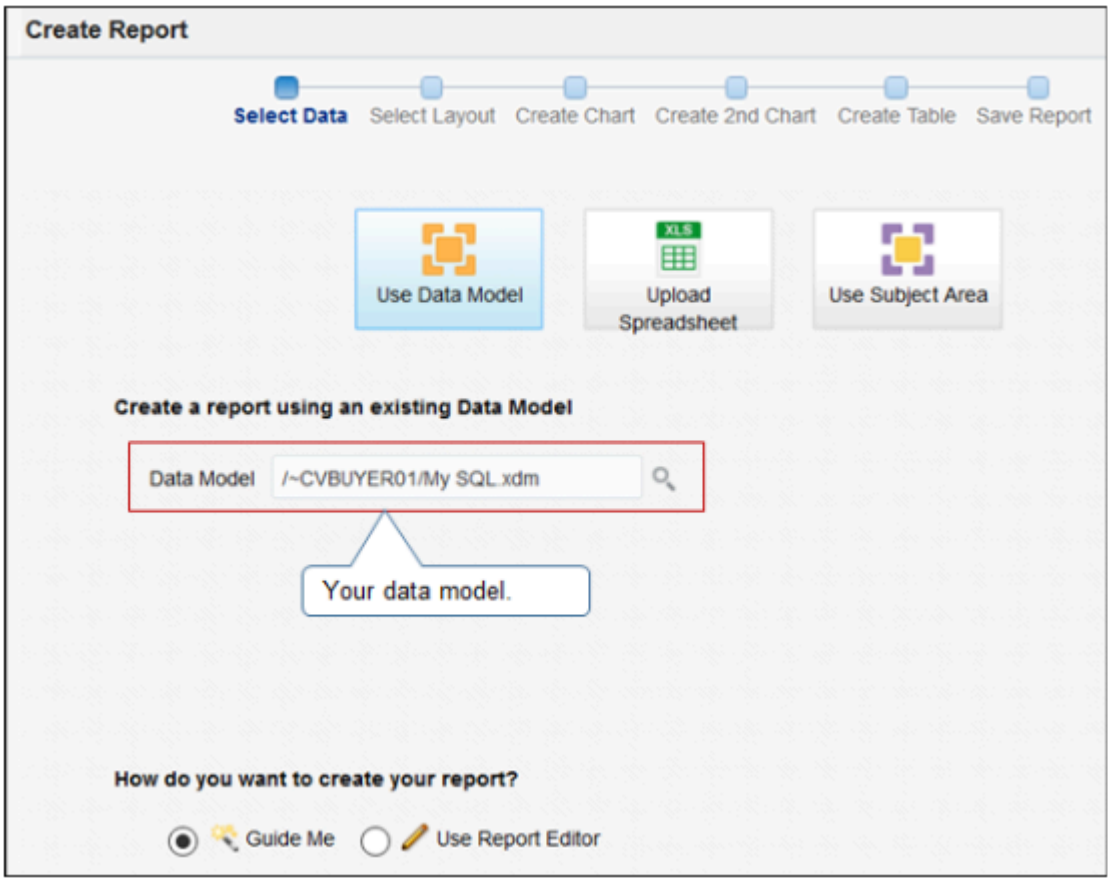
6. In the upper-right corner, click **Save**, then save the output in My Folders.

Create Report

1. On the Data tab, click **Save as Sample Data > OK**.

2. In the upper-right corner, click **Create Report**.

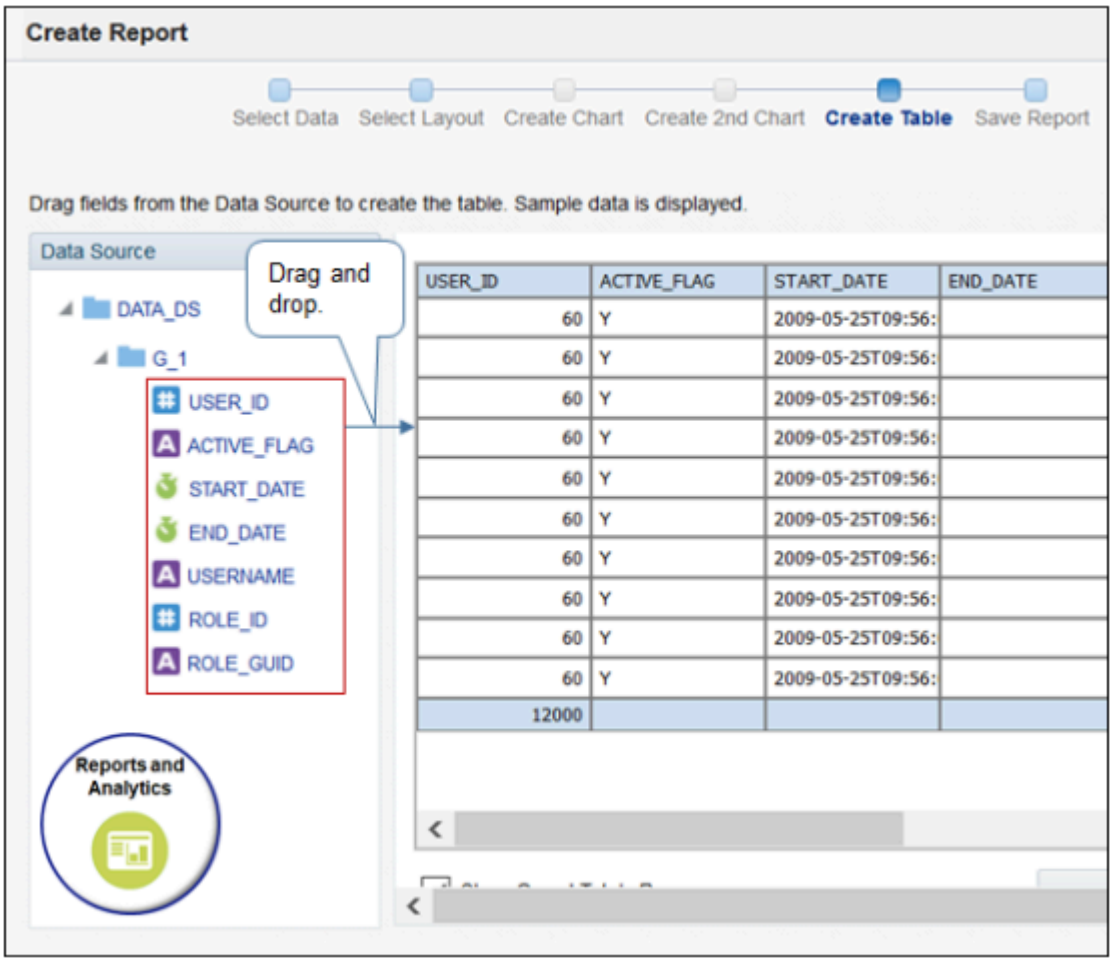
   The Create Report wizard displays for your data model.



3. Click **Next**.
4. On the Select Layout step, enable the **Landscape** option, enable the **Table** option, then click **Next**.

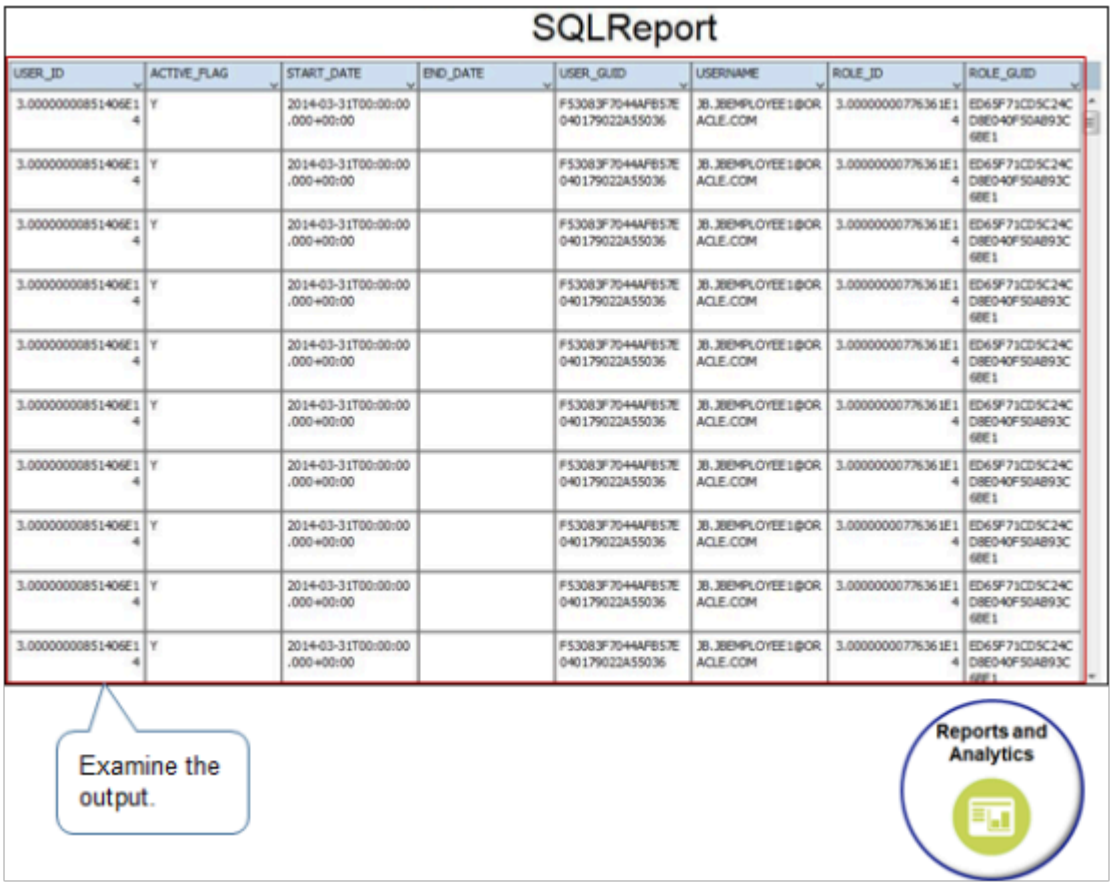
5. On the Create Table step, adjust the layout to fit your needs, then click **Next**.

   For example, drag and drop columns from the data source to the table.



6. On the Save Report step, enable the **View Report** option, then click **Finish**.

7. Save the report, name it, such as SQLReport, then examine the results on the report output that displays.

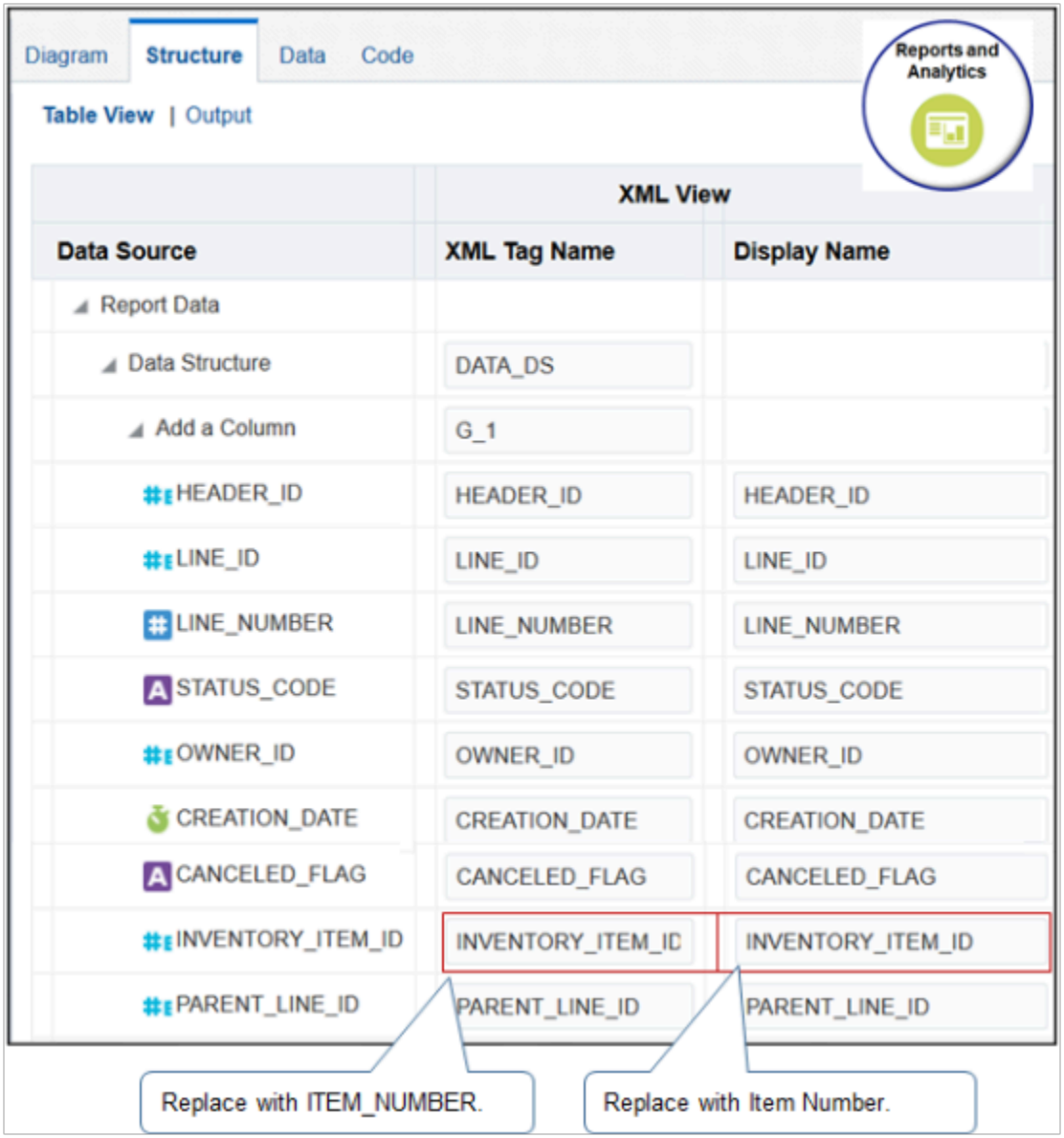


As an option, save the result to a file type of your choice.

Add a Field

You can't extend the predefined data model for Order Management. Instead, you can replace a field in the report output with the one you need.

Order Management uses the term **item** to describe the product your customer buys. The AS54888 Computer is an example of an item. The predefined data model uses the INVENTORY_ITEM_ID column to store the value that identifies the item. In this example, you replace INVENTORY_ITEM_ID with ITEM_NUMBER so your report output is more consistent with this usage.



1. Do steps 1 through 5 from earlier in this topic, except use these values in the New Data Set SQL Query dialog.

Attribute	Value
Name	Add a Column
Data Source	ApplicationDB_FSCM (Default)
Type of SQL	Standard SQL
SQL Query	`SELECT * FROM fusion.DOO_LINES_ALL`

Attribute	Value

2. Click **Structure > Table View**.
3. Locate the row that contains INVENTORY_ITEM_ID in the XML Tag Name column.
4. Modify values in the row you located.

Attribute	Old Value	New Value
XML Tag Name	INVENTORY_ITEM_ID	ITEM_NUMBER
Display Name	INVENTORY_ITEM_ID	Item Number

5. Click **View Data**, set Rows to 200, click **View**, click **Table View**, then examine the output.

Example SQL Queries

For a complete list of examples, see *SCM SQL Repository (Doc ID 2190295.1)*.

Sales Orders in Not Started Status or Processing Status and There Are No Exceptions

This problem typically happens when the SOA server (Service Oriented Architecture) becomes unstable or overloaded and the SOA transactions time out. Use a query to identify sales orders that remain in Not Started Status or Processing Status but that don't have any exceptions.

```
SELECT f.last_update_date,
h.source_order_number,
l.display_line_number,
f.fulfill_line_number,
f.status_code,
f.fulfill_line_id
FROM doo_headers_all h,
doo_fulfill_lines_all f,
doo_lines_all l
WHERE h.header_id =l.header_id
AND h.submitted_flag='Y'
AND l.line_id =f.line_id
AND f.open_flag ='Y'
AND f.creation_date>= sysdate -- Use the date range to meet your needs
AND f.status_code IN ('NOT_STARTED') -- Get orders only in NOT_STARTED status
AND NOT EXISTS
(SELECT OrchestrationGroupEO.TRANSACTION_ENTITY_ID
FROM DOO_WAIT_TASK_DETAILS WaitTaskDetailsEO,
DOO_ORCHESTRATION_GROUPS OrchestrationGroupEO
WHERE WaitTaskDetailsEO.GROUP_ID = OrchestrationGroupEO.GROUP_ID
AND WaitTaskDetailsEO.STATUS_CODE='Active'
AND OrchestrationGroupEO.STATUS ='ACTIVE'
AND TRANSACTION_ENTITY_NAME ='DOO_ORDER_FLINES_V'
AND transaction_entity_id =f.fulfill_line_id
)
ORDER BY f.source_order_number;
```

Use the Force Unlock action to recover these sales orders. For details, see *Recover Sales Orders That Are Locked or Not Started*.

Find the reservations that have finished for an item or list of items in your sales order.

```
SELECT iop.organization_code org ,
 ir.reservation_id ,
 dfa.header_id ,
 dfa.source_order_number ,
 ir.back_to_back_flag ,
 ir.staged_flag ,
 esi.item_number ,
 ir.reservation_quantity ,
 ir.creation_date
FROM inv_reservations ir ,
 inv_org_parameters iop ,
 egp_system_items_b esi ,
 doo_fulfill_lines_all dfa
WHERE iop.organization_id =ir.organization_id
 AND esi.inventory_item_id =ir.inventory_item_id
 AND esi. organization_id =ir.organization_id
 AND ir.source_fulfillment_line_id=dfa.fulfill_line_id
 AND dfa.inventory_item_id =ir.inventory_item_id
 AND ir.demand_source_type_id ='2' /*Data in Order Management comes predefined to use a
 demand_source_type_id of 2 for the sales order.*/
ORDER BY dfa.source_order_number;
```

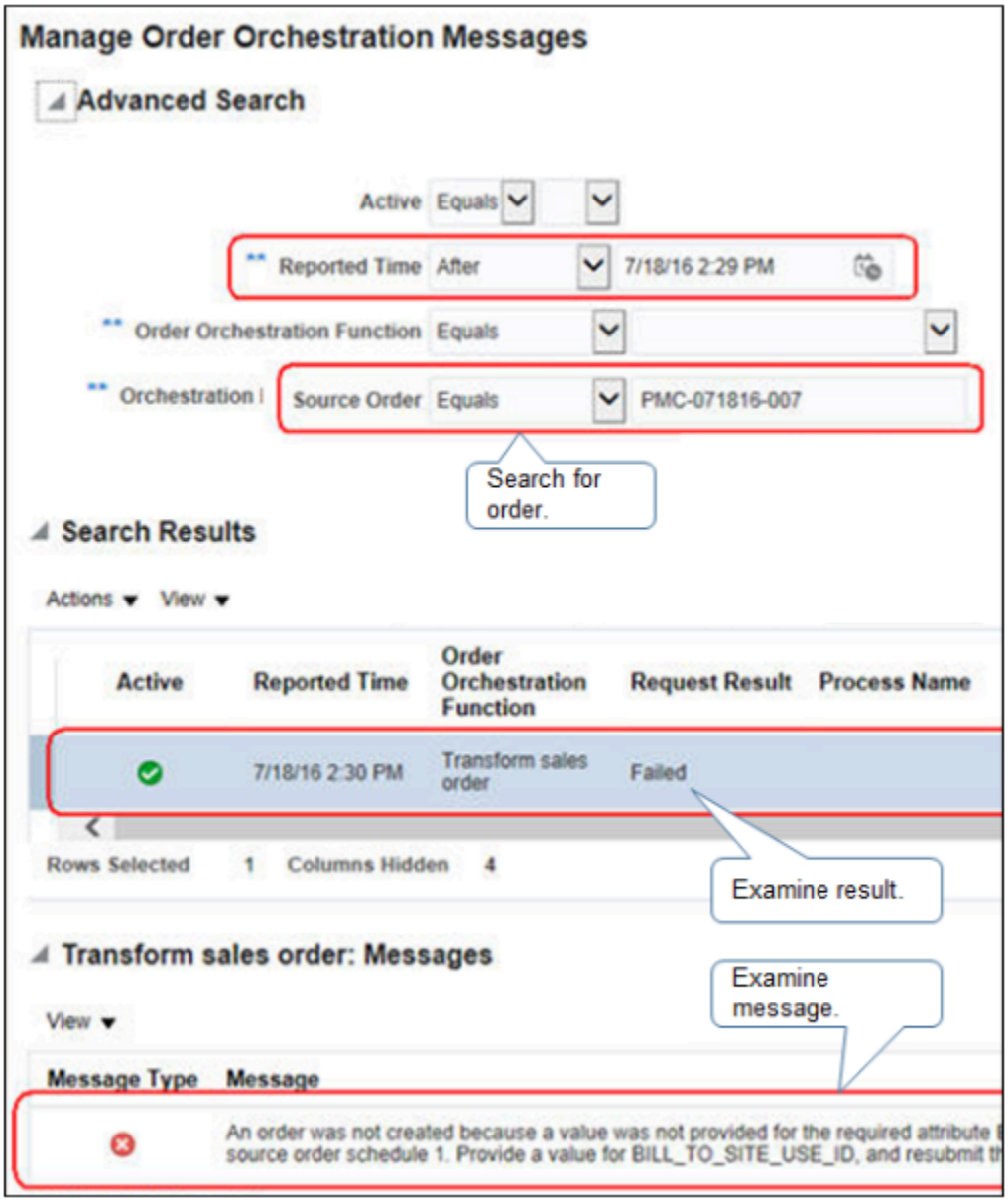
Related Topics

- Use Reports and Analytics with Order Management
- Recover Sales Orders That Are Locked or Not Started

Get Error Messages and Status Updates

Get error messages and status updates for each sales order you import or integrate.

Assume you imported source order PMC-071816-007 from your source system on 1/01/2019 at 2:30 PM, but can't find it when you search on the Overview page in the Order Management work area. You can search messages to get a status update or view error messages that happen during import.



1. Go to the Order Management work area, then click **Tasks > Manage Order Orchestration Messages**.
2. On the Manage Order Orchestration Messages page, set the values, then click **Search**.

Attribute	Value
Reported Time	After 1/01/2019 2:30 PM
Source Order	Equals PMC-071816-007

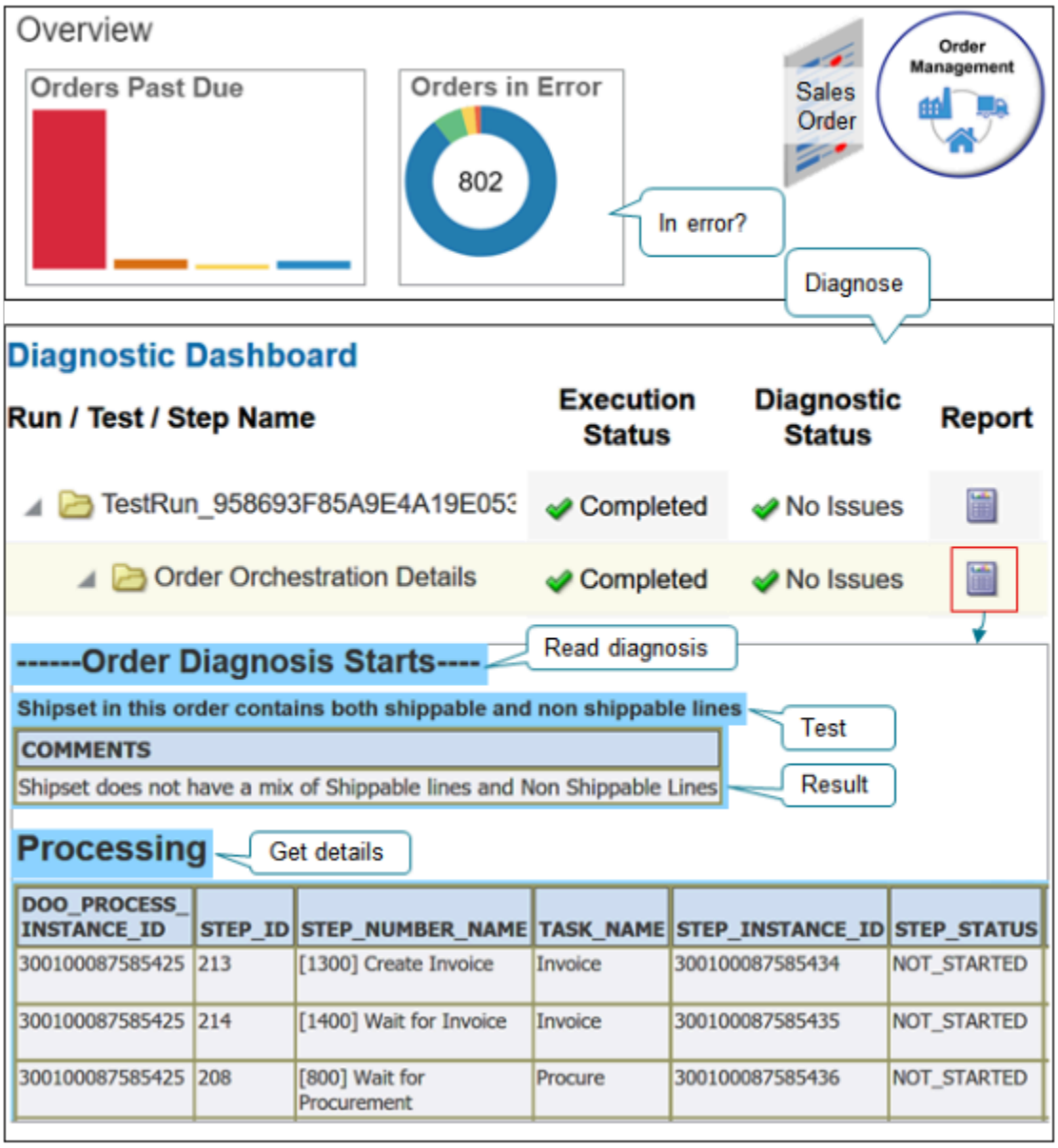
3. Examine the message in the search results.

Related Topics

- Overview of Importing Orders Into Order Management
- Roadmap for Setting Up Order-to-Cash
- How Order-to-Cash Works with Order Capture Systems

Use Diagnostics to Troubleshoot Sales Orders

Use a diagnostic tool to troubleshoot problems you have with a sales order.



Note

- Run the tool directly from your Home page in real time.
- Get the current status of the sales order.
- Get results from a range of diagnostic tests. For example, run the Order Orchestration test to determine whether.
  - A shipment set in the sales order includes a line that can't ship, such as a warranty.
  - Order fulfillment is locked because of a force cancel action.
  - A hold failure occurred.
- Get processing details about the sales order, such as which fulfillment tasks have run and their statuses.
-

Assume you must troubleshoot sales order 385081 that you created in the Order Management work area.

Try it.

1. Make sure you have the privileges that you need to administer Application Diagnostics.
   For details, go to *Security Reference for Common Features*, then see the Application Diagnostics Administrator chapter.
2. In the upper-right corner, click **down arrow > Run Diagnostics Tests**.
3. On the Diagnostic Dashboard page, in the Search for Tests area, enter the value, then click **Search**.

Attribute	Value
Test Name	Order Orchestration

4. In the search results, in the row that contains Order Orchestration Details in the Test Name attribute, add a check mark to the box that's left of the test name, then click **Add to Run**.
   In the Select Tests to Run and Supply Inputs area, notice that the page added a row.

Test Name	Prerequisites	Details	Input Status	Test Identifier
Order Orchestration Details	No	Icon you can click	Required Input Values Are Missing	Order Orchestration Details

5. In the Input Status column, click the **caution icon**.
6. In the Input Parameters dialog, in the New Value column, click the **magnifying glass**.
7. In the Search and Select dialog, enter the value.

Attribute	Value
Source Order Number	385081

8. In the search results, click that row that contains your sales order, then click **OK > OK**.

In the Select Tests to Run and Supply Inputs area, notice the value.

Attribute	Value
Input Status	Inputs Edited: Required Input Values Validated

9. In the Test Run Submitted dialog, notice the text, then click **OK**.

```
Tests Run "TestRun_958693F85A9E4A19E0535EBFF20A3CA7" has been submitted
```

Examine the results.

1. In the Diagnostic Test Run Status area, click **View > Refresh**, then examine the values.

Attribute	Value
Run / Test / Step Name	TestRun_958693F85A9E4A19E0535EBFF20A3CA7
Execution Status	Completed
Diagnostic Status	No Issues
Report	An icon you can click to get details.
Run By	Displays the user that you used to sign in, such as DIAG_ADMIN.

The refresh displays all tests that diagnostics has run in the last 24 hours.

2. In the Run / Test / Step Name column, expand the **folder**, expand the child **Order Orchestration Details folder**, then notice the hierarchy that displays.

```
TestRun_958693F85A9E4A19E0535EBFF20A3CA7
 Order Orchestration Details
 Diagnostics_Engine_Log
 Order Orchestration Data
```

3. Verify the status for each row of the hierarchy.

Attribute	Value
Execution Status	Completed

4. In the row that contains TestRun_958693F85A9E4A19E0535EBFF20A3CA7, in the Report column, click the **icon**.

The tool displays an HTML page that includes a report summary.

5. Close the report.

6. In the Diagnostic Test Run Status area, in the row that contains Order Orchestration Details, in the Report column, click the **icon**.

   The tool displays an HTML page that includes a report summary.
7. Use the report.

Use the Report

Analysis Area

Use the analysis area of the report to get details about the current state of the sales order.

Analysis	Details That the Report Displays
Recovery action is pending for the order.	Source Order Number Display Line Number Suggested Recovery
Changes to order not successfully processed.	Source Order Number Display Line Number Suggested Recovery
Order lines that haven't reached a stable state.	Task Step Name Source Order Number Display Line Number Actual Start Date Suggested Recovery
Order lines that haven't reached a stable state for tasks with an explicit wait.	Task Step Name Source Order Number Display Line Number Actual Start Date Suggested Recovery
Order lines that a fulfillment task has locked.	Source Order Number Display Line Number Suggested Recovery
Order lines that a shipping task has locked.	Source Order Number Display Line Number Suggested Recovery

Analysis	Details That the Report Displays
Order lines with problems that happen during error recovery.	Source Order Number Display Line Number Group Id Suggested Recovery

Diagnosis Area

Use the Order Diagnosis area of the report to identify problems that occurred.

Analysis	Example Results That the Report Displays
Locked draft headers.	Sales order doesn't have a locked draft.
Fulfillment lines with null delta types.	Sales order doesn't have a null delta type.
Shipment set in this order contains shippable and nonshippable lines.	Shipment set doesn't have a mix of shippable lines and nonshippable lines.
Order fulfillment is locked because of force cancel.	There are no problems related to a forced cancellation.
Double orchestration exists for fulfillment lines in the sales order.	There are no problems related to a double orchestration.
This order has pending action requests.	All user request are done and no more changes are allowed.
No active wait record exists for this order.	There are no problems related to waits.
Previous change is pending for this order.	Order lines are healthy.
This order has hold failures.	There are no apply hold failures.
Active hold on previous change.	There are no active holds.

Processing Details Area

Use the Processing Details area to get details about the processing that Order Management has done for the sales order, such as order line details, orchestration process details, orchestration process steps that have run and their statuses, fulfillment task details, pending actions, holds applied, error messages, and so on.

Here are a few of the categories that the area contains.

- Processing Order Header
- Order Header Extensible Flexfields
- Order Lines
- Order Lines Extensible Flexfields
- Processing Fulfillment Lines
- Fulfillment Lines Details
- Order Orchestration Groups
- Process BPEL Instances
- Process Instances
- Step Instances
- Step Instance Details
- Task Instances
- Order State
- Process Hold
- Action Requests
- Inventory Reservations
- Order Warehouse Details
- Order Interface To Invoice
- Order Invoice
- Order Charges
- Order Charge Components
- Order Totals
- Error Message Icon
- Order Header Error Messages
- Document References

Details are extensive. Here's some example data from part of the Step Instances section.

DOO_ PROCESS_ INSTANCE_ID	STEP_ID	STEP_ NUMBER_ NAME	TASK_NAME	STEP_ INSTANCE_ID	TASK_ INSTANCE_ID	GROUP_ID	STEP_ACTIVE	STEP_ STATUS
300100087585	213	[1300] Create Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_ STARTED
300100087585	214	[1400] Wait for Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_ STARTED
300100087585	208	[800] Wait for Procurement	Procure	300100087585	300100087585	300100087585	ACTIVE	NOT_ STARTED

DOO_PROCESS_INSTANCE_ID	STEP_ID	STEP_NUMBER_NAME	TASK_NAME	STEP_INSTANCE_ID	TASK_INSTANCE_ID	GROUP_ID	STEP_ACTIVE	STEP_STATUS
300100087585	209	[900] Create Reservation	Reserve	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	210	[1000] Create Shipment Request	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	211	[1100] Wait for Shipment Advice	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	204	[400] Pause	Pause	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	205	[500] Create Back to Back Shipment Request	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	206	[600] Wait for Back to Back Shipment Advice	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	201	[100] Schedule	Schedule	300100087585	300100087585	300100087585	ACTIVE	COMPLETED
300100087585	203	[300] Request Supply	Supply Orchestration	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	213	[1300] Create Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	214	[1400] Wait for Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	208	[800] Wait for Procurement	Procure	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	209	[900] Create Reservation	Reserve	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	210	[1000] Create Shipment Request	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED

DOO_PROCESS_INSTANCE_ID	STEP_ID	STEP_NUMBER_NAME	TASK_NAME	STEP_INSTANCE_ID	TASK_INSTANCE_ID	GROUP_ID	STEP_ACTIVE	STEP_STATUS
300100087585	211	[1100] Wait for Shipment Advice	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	204	[400] Pause	Pause	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	205	[500] Create Back to Back Shipment Request	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	206	[600] Wait for Back to Back Shipment Advice	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	201	[100] Schedule	Schedule	300100087585	300100087585	300100087585	ACTIVE	COMPLETED
300100087585	203	[300] Request Supply	Supply Orchestration	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	213	[1300] Create Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	214	[1400] Wait for Invoice	Invoice	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	208	[800] Wait for Procurement	Procure	300100087585	300100087585	300100087585	ACTIVE	NOT_STARTED
300100087585	209	[900] Create Reservation	Reserve	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	210	[1000] Create Shipment Request	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	211	[1100] Wait for Shipment Advice	Ship	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED
300100087585	204	[400] Pause	Pause	300100087585	300100087585	300100087585	INACTIVE	NOT_STARTED

DOO_ PROCESS_ INSTANCE_ID	STEP_ID	STEP_ NUMBER_ NAME	TASK_NAME	STEP_ INSTANCE_ID	TASK_ INSTANCE_ID	GROUP_ID	STEP_ACTIVE	STEP_ STATUS
300100087585	205	[500] Create Back to Back Shipment Request	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_ STARTED
300100087585	206	[600] Wait for Back to Back Shipment Advice	Ship Back-to-Back Goods	300100087585	300100087585	300100087585	INACTIVE	NOT_ STARTED
300100087585	201	[100] Schedule	Schedule	300100087585	300100087585	300100087585	ACTIVE	COMPLETED
300100087585	203	[300] Request Supply	Supply Orchestration	300100087585	300100087585	300100087585	INACTIVE	NOT_ STARTED

Get a Health Check

Run a test to scan through your database and identify problems across sales orders, then display a list of sales order that are in error.

1. On the Diagnostic Dashboard page, in the Search for Tests area, enter the value, then click **Search**.

Attribute	Value
Test Name	Order Management Health Check

2. In the Select column, add a check mark, then click **Add to Run**.
3. In the Select Tests to Run and Supply Inputs area, in the Input Status column, click the icon.
4. In the Input Parameters dialog, set the date range, then click **OK**.
5. Click **Run**.
6. In the Diagnostic Test Run Status area, click **View > Refresh**, then examine the values.
7. Refresh the view until Execution Status displays Completed.
8. In the Order Management Health Check row, click the icon in the Report column.

   Your browser displays the report in a separate tab.
9. Examine the results. For example:
   - Pending recovery for the order
   - Changes to sales order not successfully processed
   - Order lines locked by fulfillment tasks
   - Order lines locked by shipping tasks
   - Order lines that encountered problems while waiting for error recovery
   - Order lines that are at the Await Billing status even after being billed

- Order lines that are missing references
- Recover actions not available in Order Management work area but line has an error
- Unstable sales orders
- Performance of Order Management

Details

Troubleshoot Order Management

Fix problems that happen in your Order Management set ups.

Some solutions use SQL to query the Order Management database. For details, see *Use SQL to Query Order Management Data*.

To troubleshoot extensible flexfields, see *Troubleshoot Extensible Flexfields in Order Management*.

Setup

Trouble	Shoot
I use the Manage Hold Codes page in the Setup and Maintenance work area. When attempting to add a service on the Services tab, the Select and Add: Services dialog doesn't display the service I need. For example, it displays the Create Shipping service but not the Update Shipping service.	Use the Manage Task Types page to specify the tasks and services that you can select for the hold. You must set the Hold Enabled option to make the service display on the Services tab. For details, see *Set Up Task Types for Holds*.
I go to the Manage Order Lookups page in the Setup and Maintenance work area, but I can't add a lookup code that's longer than 30 characters. I need more characters to accurately describe the code.	The purpose of the Lookup Code attribute is to provide a quick way to scan all the codes. Use the Meaning or Description attributes to provide more detail. The same situation applies for other objects, such as the validation rule set in a processing constraint. Most of these objects include a code or short name, but also a description that you can use to elaborate.
I use the ImportSalesOrder web service to import values for the Scheduled Ship Date and Scheduled Arrival Date attributes of each sales order. However, Order Management replaces the imported values with the value of the Requested Ship Date attribute. It replaces the values when it updates the sales order status.	If you use Global Order Promising to schedule your sales order, then create a business rule that sets the Override Schedule attribute to Yes. This attribute instructs Global Order Promising not to schedule the dates but instead to use whatever dates that you already have. Examine an example rule. For details, see *Set Default Values for Scheduled Ship Date*
I go to the Manage Administrator Profile Values page in the Setup and Maintenance	Go to the Plan Inputs work area and do the Collect Planning Data task.

Trouble	Shoot

work area, then search the Profile Display Name attribute for Display Currency.
 	
But the Profile Value attribute in the Profile Values area of the search results doesn't have any values that I can select.
 	
This problem usually happens when you haven't collected currencies.
	• Set Source System to OPS.
	• Set Collection type to Targeted
	• Move the Currencies reference entity to the Selected Entities list.
	For details, see *Collect Planning Data for Order Management*.
	
	Verify that you collected currency data.
	
	`select * from msc_xref_mapping where entity_name like 'CURRENC%'`
	
	`select * from msc_currencies_tl`
	
	For details, see *Use SQL to Query Order Management Data*.

I encounter an error. `Fail to load task definition. :: oracle.apps.fnd.applcore.visualIr publicCommonApprovals.task.TaskLo Exception thrown in getSAXSource ::java.lang.NullPoir null` You might encounter this error because your privilege doesn't allow access to a set up task, such as the Manage Order Approval Rules task in the Setup and Maintenance work area.	You need the Manage Order Approval Rules privilege and the **BPM Workflow System Admin Role** privilege. You have two choices. • Make sure you have the privileges that you need to administer Order Management. It includes the privileges that you need. • Create a new role, assign the privileges to it, then run the *Retrieve Latest LDAP Changes* scheduled process.

Order Entry

Trouble	Shoot
I create a sales order in the Order Management work area. I search for, then add an item on the catalog line, but the line displays an error message. `No data was retrieved`	• Make sure enough stock is available in inventory to reserve and finish the transaction. For details, see *Using Inventory Management*. • Make sure you correctly set up your sourcing rules and ATP rules. Here are some resources. o Configure Global Order Promising subtopic in *Quick Start for Setting Up Order-to-Cash*. o *Defining a Basic Item Sourcing Rule and ATP Rule (Doc ID 2207137.1)*. • Make sure you collect on-hand availability. See *Collecting On Hand Inventory Levels for Scheduling purposes (Doc ID 2207161.1)*.
One of my users encounters an error while creating a sales order.	Wait for the server to come back online. If the problem persists after the server is available, then deploy the orchestration process. For details, see *Deploy Orchestration Processes*.

Trouble	Shoot
`Orchestration process 300000001288069 for order was not started because it is not deployed or the server is unavailable.` This problem happens when you create or modify an orchestration process but don't deploy it or when the server is down.	
Order Management creates a new sales order revision every time we update an attribute. We use OrderImportService or the Sales Order for Order Hub REST API to import source orders into Order Management. We have a scheduling application in our fulfillment system that periodically calculates the scheduled ship date, and we use an extensible flexfield on the order line to store the date. Our integration creates a new sales order revision every time we change the date. We don't want it to create a revision.	Use the Order Fulfillment Response Service instead of OrderImportService or REST API. Order Fulfillment Response Service doesn't create a revision. Learn about Order Fulfillment Response Service. For details, see *Web Services That You Can Use to Integrate Order Management*. For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, expand Order Management, then **click Sales Orders for Order Hub**.
I add my business unit to the Coverage Start Date parameter. Next, I create a sales order, add a coverage item, set the Contract Start Date for the item, then click Submit, but Order Management changes the Contract Start Date in the fulfillment view.	If you add your business unit to the Coverage Start Date parameter, then Order Management uses it to calculate coverage dates. For details, see *Manage Order Management Parameters*. If you prefer to manually enter the date on the order line, then remove your business unit from the Coverage Start Date parameter.
I create and submit a sales order, but the order lines remain in the Not Started status. This happens on every order I create. I also encounter an error. Task 300000090474951 for orchestration process 300000090474941 failed. Use the appropriate Order Orchestration work area action to recover this task. (DOO-2685501) I try to recover the sales order or revise the order but the problem persists.	Its possible that you have a routing rule on a pause task, which Order Management doesn't support. 1. Go to the Setup and Maintenance work area and the search for the page that you use to manage routing rules. Its either Manage External Interface Routing Rules or Manage External Integration Routing Rules for Sales Orders. For details, see *Overview of Using Business Rules With Order Management*. 2. Look for a rule that depends on a pause task and that sets the connector. For example, if you use Visual Information Builder, look for a rule that's similar to: **`If Task Type is equal to pause, then set connector name to Fusion-Reservation.`** If you use Oracle Business Rules, look for a rule that's similar to: **`If header.Task Type is "Pause", then Set Connector Name ( "Fusion-Reservation" )`** 3. Deactivate the rule. Remove the check mark from the Active option. 4. Go to the Order Management work area and use the Recover action on your sales order.
I encounter an error: **`Task Invoice for orchestration process 300000600179143 failed.`**	Set the Primary Salesperson attribute and the Sales Credits on the order header or the order line, then resubmit the sales order.

Trouble	Shoot
`Use recover task, recover order, or recover process to recover the failed task or process. (DOO-2685501) You must set the Primary Salesperson attribute and the Sales Credits on the order header or the order line, then resubmit the sales order. The Create Billing Lines fulfillment task failed because data is missing or isnt correct. The Required Salesperson option is enabled in Accounts Receivable. Source Order Information: 2829-2-1`	If you don't want to require the primary salesperson, then you can disable the option. 1. Go to the Setup and Maintenance work area, then go to the task: o Offering: Financials o Functional Area: Receivables o Task: Manage Receivables System Options 2. On the Manage Receivables System Options page, search for your business unit, then click it in the search results. 3. On the Edit System Options page, disable the Require Salesperson option, then click **Save and Close**. For details, see *Implementation Settings for Revenue Recognition*.

Return Orders

Trouble	Shoot
I create a sales order in the Order Management work area. In the Order Lines area, I click Actions > Add Unreferenced Return Lines. I add an item, set the return line type and reason, then click Add. But the return line doesn't have a price.	Promote your pricing algorithms. For details, see *Promote Pricing Algorithms Into the Latest Update*.
My return line is stuck in the Awaiting Receiving status.	Run the *Send Receipt Confirmation* scheduled process. As an option, set the Source Document Number parameter to the number that identifies the source order that contains the return line. For example, if the source order number is 14052903, then set Source Document Number to 14052903.
I encounter an error. `Order management didn't create a sales order for source order AFH879, source order line 7, source order schedule 4, billing plan 3. If the price on the return order is different from the price on the original order, then you must set the cancel reason to Price Change and resubmit the order.` In some situations you might need to adjust the price for future billing periods after you start recurring billing, but you can do this adjustment only for a referenced return line.	If you adjust price on a referenced return line for an outbound line that has recurring billing, and if the unit price on the return line is different from the unit price on the original line, then you must set the CancelReasonCode attribute to ORA_PRICE_CHANGE in your import payload. If you use any other value, then you'll get this error message.

Return a Closed Order Line

I encounter an error when I attempt to create a return for a closed order line.

This error might happen because the setup in Product Information Management doesn't allow you to return the item. You must set it up. For details, see *Guidelines for Processing Return Orders*.

As an option, run SQL on your Oracle database to confirm the value of the Returnable attribute on the item and the fulfillment line.

```
SELECT dfla.source_order_number ,
 dfla.inventory_item_id ,
 dfla.status_code ,
 dfla.returnable_flag "Fline Return Flag",
 esib.Item_number ,
 esib.Returnable_flag "Item Return Flag"
FROM doo_fulfill_lines_all dfla,
 egp_system_items_b esib
WHERE source_order_number = '&SOURCE_ORDER_NUMBER'
 AND esib.inventory_organization_id = dfla.inventory_organization_id
 AND esib.inventory_item_id = dfla.inventory_item_idWord' to retain layout.]
```

Features

Transportation Management

Trouble	Shoot
Order Management sends a request to Transportation Management, but the web service between Integration Cloud Service and Transportation Management isn't up and running, so the Order Management work area displays an error icon on the fulfillment line.	Use error recovery to retry the task. Don't retry the call through Integration Cloud Service because Order Management can't accept an update from Transportation Management while the task is in error.
Transportation Management sends an update to Order Management, but Order Management rejects the update because the fulfillment line isn't on a wait step in the orchestration process, or the orchestration process is already processing another request.	Manually resend the request from Transportation Management.

Sales Agreements

Trouble	Shoot
I set the Customer attribute on the order header, but no sales agreements are available in the Sales Agreement attribute on the order header when you create a sales order.	Make sure you set up a sales agreement for the customer and that its active.

Trouble	Shoot
The sales agreements that are available on the order header are different from the ones that are available on the order line.	Examine your sales agreement set up. Order Management includes sales agreements on the. • Order header that you set up for the customer and are active. • Order line that you set up for the customer, are active, and that you specified for the item on the order line.

Error Messages

Message	Solution
`Pricing did not apply a sales agreement adjustment for charge Sale Price because it cannot find adjustment basis 100010.`	Make sure you correctly set up the adjustment basis on the pricing term for the sales agreement.
`Pricing did not apply a tier adjustment on the sales agreement for charge Sale Price because it cannot find adjustment basis 100010.`	Make sure you correctly set up the adjustment basis on the pricing term that adjusts price according to tiers for the sales agreement.
`Pricing did not apply a tier adjustment for the sales agreement to charge Sale Price because it cannot find tier basis 100010.`	Make sure you correctly set up the tier basis on the pricing term that adjusts price according to tiers for the sales agreement. Make sure you correctly set up the price element on the tier basis. For example, you can't use the QP_NET_PRICE price element as the tier basis for a pricing term that references a sales agreement.

Get Details

Here's some SQL that you can use to get details about sales agreements that you can set on the order header.

```
SELECT h.CONTRACT_ID, h.ID, h.MAJOR_VERSION, h.VERSION_TYPE,h.CONTRACT_NUMBER, h.CONTRACT_NUMBER_MODIFIER,
 h.COGNOMEN CONTRACT_NAME, h.SHORT_DESCRIPTION, h.DESCRIPTION,h.STS_CODE, h.ORG_ID, h.INV_ORGANIZATION_ID,
 h.CURRENCY_CODE,h.START_DATE, h.END_DATE,
p.OBJECT1_ID1 PRIMARY_PARTY_ID, p.RLE_CODE FROM OKC_K_HEADERS_VL h, OKC_CONTRACT_TYPES_VL t,
 OKC_K_PARTY_ROLES_VL p
where h.CONTRACT_TYPE_ID=t.CONTRACT_TYPE_ID and h.BUY_OR_SELL='S' and t.CONTRACT_CLASS='AGREEMENT'
 AND t.INTENT='S' and t.LINE_CLASS='SALES_AGREEMENT' AND h.ID=p.DNZ_CHR_ID(+) AND
 h.MAJOR_VERSION=p.MAJOR_VERSION(+) AND p.PRIMARY_YN(+)='Y'
and h.STS_CODE='ACTIVE' and h.VERSION_TYPE in ('C','A') and h.TEMPLATE_YN='N' and h.ORG_ID=<BuId> and
 h.CURRENCY_CODE=<CurrencyCode>
and p.OBJECT1_ID1=<PartyId> and <PricingDate> BETWEEN h.START_DATE AND Nvl(h.END_DATE,<PricingDate>)
```

Here's some SQL that you can use to gets details about sales agreements that you can set on the order line.

```
SELECT h.CONTRACT_ID, h.ID CHR_ID, l.MAJOR_VERSION, l.VERSION_TYPE, h.CONTRACT_NUMBER,
 h.CONTRACT_NUMBER_MODIFIER, h.COGNOMEN CONTRACT_NAME, h.SHORT_DESCRIPTION CONTRACT_SHORT_DESCRIPTION,
 h.DESCRIPTION CONTRACT_DESCRIPTION,l.STS_CODE, h.ORG_ID, h.INV_ORGANIZATION_ID,
 l.CURRENCY_CODE,l.START_DATE, l.END_DATE, l.UOM_CODE,l.ID LINE_ID, l.LINE_NUMBER, l.ITEM_NAME,
 l.ITEM_DESCRIPTION,
l.OBJECT1_ID1 ITEM_ID, l.OBJECT1_ID2 ITEM_INV_ORG_ID, p.OBJECT1_ID1 PRIMARY_PARTY_ID, p.RLE_CODE
FROM OKC_K_HEADERS_VL h, OKC_CONTRACT_TYPES_VL t, OKC_K_PARTY_ROLES_VL p, OKC_K_LINES_VL l
where h.CONTRACT_TYPE_ID=t.CONTRACT_TYPE_ID and h.BUY_OR_SELL='S' and t.CONTRACT_CLASS='AGREEMENT' AND
 t.INTENT='S' and t.LINE_CLASS='SALES_AGREEMENT' AND h.ID=l.DNZ_CHR_ID AND h.MAJOR_VERSION=l.MAJOR_VERSION
```

```
 and l.SOURCE_CODE_CLASS='SALES_AGREEMENT' and l.STS_CODE='ACTIVE' AND h.ID=p.DNZ_CHR_ID(+) AND
 h.MAJOR_VERSION=p.MAJOR_VERSION(+) AND p.PRIMARY_YN(+)='Y'
-- and h.STS_CODE='ACTIVE' and h.VERSION_TYPE in ('C','A') and h.TEMPLATE_YN='N' and h.ORG_ID=<BuId>
 and h.CURRENCY_CODE=<CurrencyCode> and l.OBJECT1_ID1=<ItemId> and l.OBJECT1_ID2=<InvOrgId> and
 l.UOM_CODE=<UomCode>
and <PricingDate> BETWEEN l.START_DATE AND Nvl(l.END_DATE,<PricingDate>)
```

Order Management Extensions

Trouble	Shoot

We deploy Order Management in a language that isn't English, such as Japanese. I set the SetName attribute to Common Set in an extension.
 	
`vcrow.setAttribute("SetName", "Common Set");`
 	
I get an error when I submit the sales order.
 	
`Error executing Extended Order Type-AR Trns Type in event On Start of Submission Request: oracle.jbo.JboException: JBO-29000: Unexpected exception caught: java.lang.NullPointerException, message = Cannot invoke method getAttribute () on null object. (DOO-2685874) '`
	This problem happens because Common Set might not be available for your language. Instead, you can set the SetCode attribute to COMMON.
	
	`vcrow.setAttribute("SetCode", "COMMON");`
	
	For an example of this usage, see the `//Function to get Billing Transaction Type` code in the Set the Billing Transaction Type According to Order Type subtopic. For details, see *Extend Order Headers*.

My order management extension doesn't run when use I the Order Management work area to cancel a sales order, but it does run if I use REST API to cancel the order.	This problem happens because you can run an order management extension when you cancel a sales order only through REST API. You can't run an order management extension when you cancel the sales order in the Order Management work area.

Class or Type Not Allowed

I receive a response payload in XML format when I use a web service.

```
<ORACLE_INTEGRATION_MESSAGE xmlns="http://xmlns.oracle.com/oih/oracle_integration_message">
<DOCUMENT_NAME>STORE_PAYMENT_TRANSACTION</DOCUMENT_NAME>
<DOCUMENT_NUMBER>99999</DOCUMENT_NUMBER>
<DOCUMENT_TYPE>NEW_PAYMENT</DOCUMENT_TYPE>
<FROM_SYSTEM>AR</FROM_SYSTEM>
<TO_SYSTEM>STORE</TO_SYSTEM>
<PAYLOAD>
<PARAMETERLIST>
<!-- core mandatory attributes -->
<PARAMETER>
<NAME>GEC_SET_ID</NAME>
<VALUE>46777</VALUE>
</PARAMETER>
<PARAMETER>
<NAME>TRANSACTION_ID</NAME>
<VALUE>36787503</VALUE>
</PARAMETER>
</PARAMETERLIST>
```

```
</PAYLOAD>
</ORACLE_INTEGRATION_MESSAGE>
```

I have to parse this response in an order management extension. I attempt to import the XmlSlurper library to do the parse but the order management extension displays an error.

```
The use of this class or type is not allowed: groovy.util.XmlSlurper
```

Learn how to fix this problem. For details, see the Extract Details subtopic in *Troubleshoot Problems with Web Services*.

Performance

Trouble	Shoot
Performance decreases each time I add an order line to a sales order that I create or edit in the Order Management work area. For example, I add 25 order lines. • The first 15 requires about 1 second to add each line. • The 21st line requires 10 seconds. • The 22nd line requires 12 seconds. • The 25th line requires 22 seconds. I save the order, requery it, then add the 26th line, which requires 2 seconds. To avoid the problem, I save and requery the order each time I add 15 more lines. However, I have over 100 lines to add. This problem typically happens because you defined a pretransformation rule but didn't test to make sure the attribute value was empty before doing the transformation. So the rule examines every row in the sales order, including rows that the rule already evaluated, instead of evaluating only the rows that don't contain a value.	Make sure the attribute value that you transform in a pretransformation rule is empty before you set the value. For details, see *Manage Pretransformation Rules*.
It takes a long time to open the Overview page or to query for a sales order from the Overview page in the Order Management work area. This problem might happen because it takes a long time for Order Management to create and display infolets on the Overview page.	Use a quick action link to get to the page that you use to create or manage sales orders instead of going through the Overview page. 1. Open a sandbox. 2. Click **Structure**. 3. Navigate to Order Management. 4. Enable the Add as a Quick Action Link on Home Page option for the Create Orders and for the Manage Orders task. 5. The next time you need to create or manage a sales order, use the link on your home page. For details, see *Overview of Sandboxes*. If you don't use jeopardy, then set the Enable Orchestration Process Planning and Calculate Jeopardy order management parameter to No. For details, see *Manage Order Management Parameters*.

Related Topics

- Overview of Importing Orders Into Order Management
- Set Up User Roles and Privileges in Order Management
- Use SQL to Query Order Management Data
- Troubleshoot Problems with Configure-to-Order

Troubleshoot Problems with Web Services

Fix problems that happen in your Order Management implementation that involve web services.

No Matching Row

No Matching Row

Assume you use a web service to import a sales order. The payload includes:

```
<ns2:ProductNumber>BIGSTORE1111</ns2:ProductNumber>
<ns2:InventoryOrganizationIdentifier>300000001384059</ns2:InventoryOrganizationIdentifier>
```

And you encounter this error:

```
An order was not created because no matching row for attribute ProductNumber with the value BIGSTORE1111
 was found for the source order with the following details: source order BIGSTORE0007, source order line 2,
 source order schedule 2. Check the attribute value, and resubmit the order.
```

The No Matching Row error typically happens because the Oracle database doesn't contain the same value that your import contains. In this example, your import inventory item isn't defined in the Oracle database or isn't associated with the Inventory Organization that the web service payload specifies.

Use SQL to query the Oracle database.

```
SELECT HOU.ORGANIZATION_ID,
 HOUTL.NAME,
 HOUCL.CLASSIFICATION_CODE
FROM HR_ALL_ORGANIZATION_UNITS_F HOU,
 HR_ORGANIZATION_UNITS_F_TL HOUTL,
 HR_ORG_UNIT_CLASSIFICATIONS_F HOUCL
WHERE
 HOU.ORGANIZATION_ID=HOUTL.ORGANIZATION_ID AND
 HOU.ORGANIZATION_ID =HOUCL.ORGANIZATION_ID AND
 HOUCL.CLASSIFICATION_CODE = 'INV'
ORDER BY
 HOUTL.NAME;
```

The query returns:

ORGANIZATION_ID	NAME	CLASSIFICATION_CODE
300000001286109	BIGSTORE Item Org	INV
300000001286592	BIGSTORE_CN_DC1	INV

ORGANIZATION_ID	NAME	CLASSIFICATION_CODE
300000001286615	BIGSTORE_SG_DC1	INV
300000001286262	BIGSTORE_US_MFG1	INV
300000001286412	BS Master Inventory Org	INV
300000001286439	BS Retail Inventory Org	INV

Run another query to get the inventory item that's associated with the inventory organization.

```
SELECT
 ItemPEO.INVENTORY_ITEM_ID,
 ItemPEO.ORGANIZATION_ID,
 HOUTL.NAME,
 ItemPEO.ITEM_NUMBER,
 ItemPEO.INVENTORY_ITEM_STATUS_CODE
FROM EGP_SYSTEM_ITEMS_B ItemPEO,
 HR_ORGANIZATION_UNITS_F_TL HOUTL
where
 HOUTL.ORGANIZATION_ID = ItemPEO.ORGANIZATION_ID and
 ItemPEO.ITEM_NUMBER = 'BIGSTORE1111';
```

The query returns:

INVENTORY_ITEM_ID	ORGANIZATION_ID	NAME	ITEM_NUMBER	INVENTORY_ITEM_STATUS_CODE
300000001292032	300000001286109	BIGSTORE Item Org	BIGSTORE1111	Active

The payload sent `InventoryOrganizationIdentifier>300000001384059<`, but the Oracle database contains 300000001286109 for ORGANIZATION_ID.

Here's the correct value to send.

```
<ns2:ProductNumber>BIGSTORE1111</ns2:ProductNumber>
<ns2:InventoryOrganizationIdentifier>300000001286109</ns2:InventoryOrganizationIdentifier>
```

You attempt to import a source order. The import payload includes:

```
<ns2:ProductNumber>BIGSTORE1111</ns2:ProductNumber> <ns2:InventoryOrganizationIdentifier>300000001384059</
ns2:InventoryOrganizationIdentifier>
```

But you encounter this error during import:

```
An order was not created because no matching row for attribute ProductNumber with the value BIGSTORE1111
 was found for the source order with the following details: source order BIGSTORE0007, source order line 2,
 source order schedule 2. Check the attribute value, and resubmit the order.
```

This problem happens because the inventory organization in your import payload isn't defined in the Oracle database.

To fix this problem, first get the list of inventory organizations that the Oracle database currently contains. Run this SQL.

```
SELECT HOU.ORGANIZATION_ID,
```

```
 HOUTL.NAME,
 HOUCL.CLASSIFICATION_CODE
FROM HR_ALL_ORGANIZATION_UNITS_F HOU,
 HR_ORGANIZATION_UNITS_F_TL HOUTL,
 HR_ORG_UNIT_CLASSIFICATIONS_F HOUCL
WHERE
 HOU.ORGANIZATION_ID=HOUTL.ORGANIZATION_ID AND
 HOU.ORGANIZATION_ID =HOUCL.ORGANIZATION_ID AND
 HOUCL.CLASSIFICATION_CODE = 'INV'
ORDER BY
 HOUTL.NAME;
```

Assume the query returns:

ORGANIZATION_ID	NAME	CLASSIFICATION_CODE
300000001286109	BIGSTORE Item Org	INV
300000001286592	BIGSTORE_CN_DC1	INV
300000001286615	BIGSTORE_SG_DC1	INV
300000001286262	BIGSTORE_US_MFG1	INV
300000001286412	BS Master Inventory Org	INV
300000001286439	BS Retail Inventory Org	INV

Next, get details for BIGSTORE1111, which is the inventory organization that your import payload specifies.

```
SELECT
 ItemPEO.INVENTORY_ITEM_ID,
 ItemPEO.ORGANIZATION_ID,
 HOUTL.NAME,
 ItemPEO.ITEM_NUMBER,
 ItemPEO.INVENTORY_ITEM_STATUS_CODE
FROM EGP_SYSTEM_ITEMS_B ItemPEO,
 HR_ORGANIZATION_UNITS_F_TL HOUTL
where
 HOUTL.ORGANIZATION_ID = ItemPEO.ORGANIZATION_ID and
 ItemPEO.ITEM_NUMBER = 'BIGSTORE1111';
```

Assume the query returns:

INVENTORY_ITEM_ID	ORGANIZATION_ID	NAME	ITEM_NUMBER	INVENTORY_ITEM_STATUS_CODE
300000001292032	300000001286109	BIGSTORE Item Org	BIGSTORE1111	Active

Next, fix your import payload. Change this current value:

```
<ns2:ProductNumber>BIGSTORE1111</ns2:ProductNumber> <ns2:InventoryOrganizationIdentifier>300000001384059</
ns2:InventoryOrganizationIdentifier>
```

To the value that the query returned in column ORGANIZATION_ID:

```
<ns2:ProductNumber>BIGSTORE1111</ns2:ProductNumber> <ns2:InventoryOrganizationIdentifier>300000001286109</
ns2:InventoryOrganizationIdentifier>
```

Attribute Values

Assume you use enterprise business message ProcessSalesOrderFulfillmentEBM to create a sales order and encounter this error.

```
Error Message: {http://xmlns.oracle.com/apps/scm/doo/decomposition/receiveTransform/transformSalesOrder/
DooDecompTransformSalesOrderComposite}ProcessSalesOrderFulfillmentResponseEBM
Fault ID default/DooDecompTransformSalesOrderComposite!11.1.5.0.0*soa_7289e448-8b78-408f-af5c-116da7e1d9d2/
DecompositionProcess/70017-BpThw8-BpSeq55.49-2
Fault Time Sep 16, 2013 11:24:34 AM
Non Recoverable Business Fault :1 env:Server Attribute ORG_ID has invalid value 300000000562209 for service
 ProcessOrderValidation.
```

This error happens because the payload specifies value 300000000562209 for attribute ORG_ID, but the Order Management database doesn't contain this value. The payload also doesn't correctly specify attribute BusinessUnitReference and attribute BusinessUnitIdentification.

To fix this problem, query the database.

```
SELECT
 FABUV.BU_ID,
 FABUV.BU_NAME,
 FABUV.DEFAULT_SET_ID,
 IOP.ORGANIZATION_ID,
 IOP.ORGANIZATION_CODE,
 HAOU.NAME
FROM
 INV_ORG_PARAMETERS IOP,
 HR_ALL_ORGANIZATION_UNITS HAOU,
 FUSION.FUN_ALL_BUSINESS_UNITS_V FABUV
WHERE
 HAOU.ORGANIZATION_ID = IOP.ORGANIZATION_ID AND
 IOP.BUSINESS_unit_id = FABUV.BU_ID
ORDER BY
 IOP.ORGANIZATION_CODE,
 FABUV.BU_ID;
```

Assume the query returns:

BU_ID	BU_NAME	DEFAULT_SET_ID	ORGANIZATION_ID	ORGANIZATION_CODE	NAME
300000001110916	USA1 Business Unit	300000001111782	300000001201056	001	Seattle Warehouse
300000001110916	USA1 Business Unit	300000001111782	300000003887955	999	External Item Organization
300000001130053	USA2 Business Unit	300000001111783	300000001201066	002	Atlanta Warehouse
300000001130053	**USA2 Business Unit**	300000001111783	300000001130184	003	**Chicago Distribution Center**

BU_ID	BU_NAME	DEFAULT_SET_ID	ORGANIZATION_ID	ORGANIZATION_CODE	NAME
300000001341196	China Business Unit	300000001341955	300000004396135	050	Shanghai Distribution Center
300000001341196	China Business Unit	300000001341955	300000004396139	051	Shenzhen Warehouse

Assume you're attempting to create a sales order for Chicago Distribution Center, which is in USA2 Business Unit, and attribute BU_ID for this business unit contains 300000001130053. To fix the problem, you must define these attributes. Add this code to you payload.

```
<corecom:BusinessUnitReference>
 <corecom:BusinessUnitIdentification>
 <!-- BUSINESS UNIT ID-->
 <corecom:AlternateObjectKey>
 <corecom:ID>300000001130053</corecom:ID>
 </corecom:AlternateObjectKey>
 </corecom:BusinessUnitIdentification>
 </corecom:BusinessUnitReference>
```

Performance Problems

I use OrderFulfillmentResponseService to set up an integration with Integration Cloud Service so we can update the status on the fulfillment line. It works fine until we have a lot of sales orders to process. When we have thousands of order lines to update we expect the update to finish in about 10 to 15 minutes, but it actually takes over two hours.

The integration processes requests in parallel whether you set the Invocation Mode attribute on the connector to Asynchronous Service or Synchronous. For example, if you have 1,000 requests, the integration will process them all at the same time. It doesn't put them in a queue. Also, serviceOrderFulfillmentResponseService doesn't limit volume. For example, it doesn't put limits on the number of web service calls that it can send in one minute.

This problem usually happens because the server that you use to host Integration Cloud Service isn't optimized to handle a high volume of transactions. To fix this problem, work with the person who administers the server that hosts Integration Cloud Service and see if you can optimize it to handle more transactions, such as adding the number of threads the server can process at the same time.

Other Weird Stuff

You send a payload to web service ReceiveOrderRequest to request to create a sales order.

```
<ns1:OrchestrationOrderRequest xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/model/">
 <ns2:SourceTransactionIdentifier>PMC-070716-013</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionNumber>PMC-070716-013</ns2:SourceTransactionNumber>
 <ns2:BuyingPartyName>JAH CUSTOMER SHARED</ns2:BuyingPartyName>
 <ns2:BuyingPartyContactName>Peter Pan</ns2:BuyingPartyContactName>
 <ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
 <ns2:TransactionOn>2016-07-06T02:12:01</ns2:TransactionOn>
 <ns2:RequestingBusinessUnitIdentifier>US_W_888_BU</ns2:RequestingBusinessUnitIdentifier>
 <ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
 <ns2:OrchestrationOrderRequestLine>
```

The request seems to run successfully but the Order Management work area doesn't display the sales order. You run a flow trace in Oracle Enterprise Manager, which displays:

```
<bpelFault><faultType>0</faultType><selectionFailure xmlns="http://docs.xyz.org/wsbpel/2.0/process/
executable"></selectionFailure></bpelFault>
```

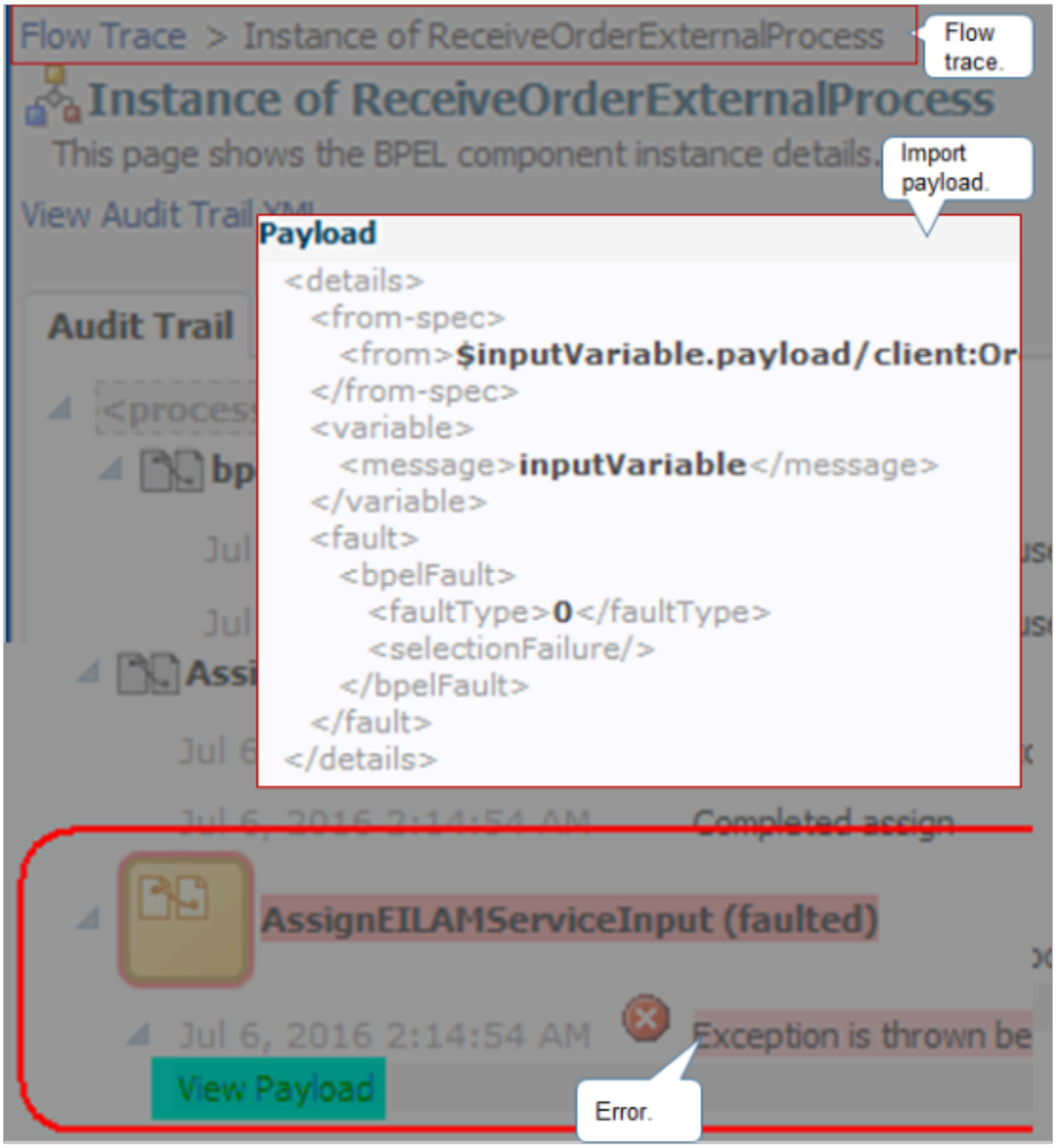
And also displays:

```
Exception is thrown because the from-spec at line 1,616 is evaluated to be empty
```

And also displays:

```
AssignEILAMServiceInput (faulted)

<details>
 <from-spec>
 <from>$inputVariable.payload/client:OrchestrationOrderRequest/ns4:SourceTransactionSystem</from>
 </from-spec>
 <variable>
 <message>inputVariable</message>
 </variable>
 <fault>
 <bpelFault>
 <faultType>0</faultType>
 <selectionFailure/>
 </bpelFault>
 </fault>
</details>
```

For example:



The problem happens because your request payload doesn't specify the source system. To fix it, add attribute SourceTransactionSystem to your payload.

```
<ns2:SourceTransactionSystem>ORA_BM_CPQ</ns2:SourceTransactionSystem>
```

For example:

```
<ns1:OrchestrationOrderRequest xmlns:ns2="http://xmlns.oracle.com/apps/scm/doo/decomposition/
receiveTransform/receiveSalesOrder/model/">
 <ns2:SourceTransactionIdentifier>PMC-070716-013</ns2:SourceTransactionIdentifier>
 <ns2:SourceTransactionNumber>PMC-070716-013</ns2:SourceTransactionNumber>

 <ns2:SourceTransactionSystem>ORA_BM_CPQ</ns2:SourceTransactionSystem>

 <ns2:BuyingPartyName>GED CUSTOMER SHARED</ns2:BuyingPartyName>
 <ns2:BuyingPartyContactName>Peter Pan</ns2:BuyingPartyContactName>
```

```
<ns2:TransactionalCurrencyCode>USD</ns2:TransactionalCurrencyCode>
<ns2:TransactionOn>2016-07-06T02:12:01</ns2:TransactionOn>
<ns2:RequestingBusinessUnitIdentifier>US_WT_DG09_BU</ns2:RequestingBusinessUnitIdentifier>
<ns2:PartialShipAllowedFlag>false</ns2:PartialShipAllowedFlag>
<ns2:OrchestrationOrderRequestLine>
```

View an example that uses flow trace. For details, see *Route Requests from Order Management to Fulfillment Systems*.

For details about Oracle Enterprise Manager, see https://www.oracle.com/technetwork/oem/enterprise-manager/overview/index.html.

We use the Order Import web service and Receive Order service to import source orders. I need the header details and order line details for holds that Order Management applies as a result of the import. I need to know whether Order Management applies the hold and, if not, why not.

The response from Order Import provides details in the callback for each asynchronous operation. Examine the elements in the response.

```
ord:ReturnStatus?</ord:ReturnStatus>
<!--Optional:-->
ord:MessageName?</ord:MessageName>
<!--Optional:-->
ord:MessageText?</ord:MessageText>
```

These elements indicate whether the service successfully processed the sales order or, if not, the errors it encountered and error text.

The RequestHold operation and Release Pause Tasks operation are asynchronous. You must implement the callback to receive a response from them.

Related Topics

- Overview of Importing Orders Into Order Management
- Roadmap for Setting Up Order-to-Cash
- Set Up User Roles and Privileges in Order Management
- Use SQL to Query Order Management Data

Filter Lines In Your Extensions, Rules, and Constraints

Make sure you filter out lines that you don't want to revise when you create an order management extension or business rule, and that you constrain the changes that you allow on the fulfillment line.

For example:

- If you create a business rule that modifies a value on a fulfillment line that's still in progress, then filter out lines that are canceled, closed, on backorder, or that Order Management has already sent to billing.
- Filter out order lines that you already fulfilled. For example, filter out lines that you already shipped for outbound lines or lines that you already received and delivered for return lines.
- Filter so you only process changes that you make to billing attributes, for example, on the order line's Billing tab, and only if you haven't already sent the order line to billing. For example, modify only the Accounting Rule, Payment Terms, Receivable Transaction Type, and so on. Don't modify any other attributes.
- Avoid the NullPointerException error. If your logic depends on using an attribute value as part of a calculation, then filter out lines that include an attribute that doesn't contain a value.

- Filter according to line category code. For example, to process only order lines, not return lines, filter the categoryCode attribute on the fulfillment line according to ORDER. To process only returns, filter it according to the value RETURN.
- If you use an order management extension, pretransformation rule, or posttransformation rule to set the default values for attributes, then filter out fulfillment lines that reference the original return when you populate the value for the Accounting Rule attribute and Invoicing Rule attribute.

Here are some more details that you might find useful.

- Filter out lines that aren't shippable. Don't attempt to ship an item that isn't shippable, such as a warranty. For an example that filters out lines that aren't shippable, see *Select Fulfillment Lines for Orchestration Process Steps*.
- Filter out lines that reference a coverage item, such as a service agreement. Reserve means you reserve an item in inventory. You don't store a service agreement in inventory because it isn't a physical item, so don't reserve it. For details, see *Another Example of Using Extensible Flexfields In Line-Selection Rules*.
- Filter out return lines that you don't want to ship on an outbound sales order. For details, see *Prevent Orchestration Process from Shipping Return Lines*.
- Filter out lines that already passed trade compliance. If the line passed, then don't send a request to screen the line for trade compliance. For details, see *Use Extensible Flexfields in Line-Selection Rules*.

Create Filters In an Order Management Extension

Assume you're revising a line that Order Management is currently fulfilling, so you don't want to revise lines that are closed, canceled, shipped, or that Order Management already sent to accounts receivable. Write an extension that filters the lines.

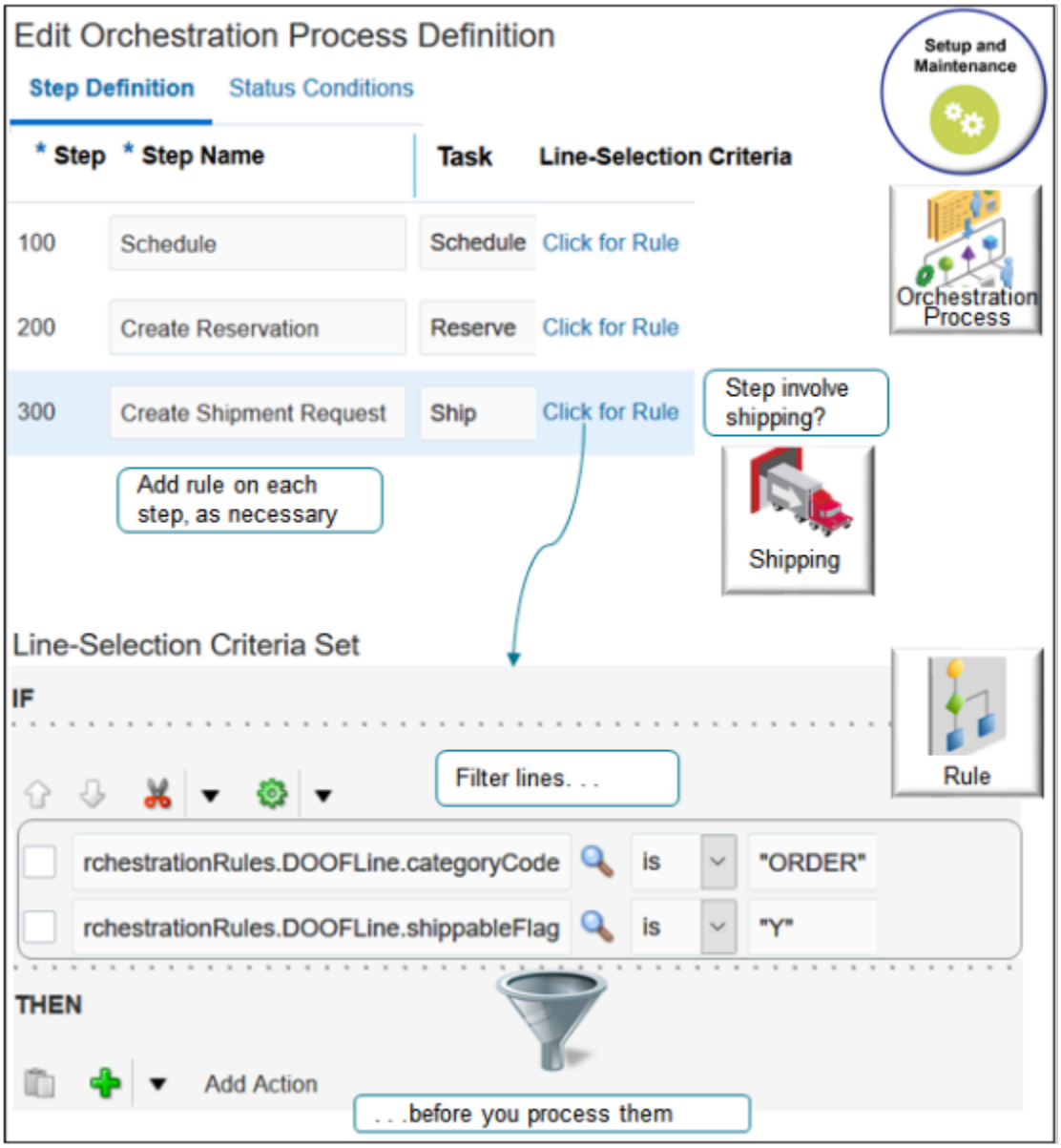
```
//
//============================================================
import oracle.apps.scm.doo.common.extensions.ValidationException;

def lines = header.getAttribute("Lines");

while( lines.hasNext() ) {
 def line = lines.next();
 Long referenceFlineId = line.getAttribute("ReferenceFulfillmentLineIdentifier");

 // If the reference line is null then this isn't a revision.
 if(referenceFlineId != null) {
 // Get running line if this is a revision.
 def runningLine = getLinesFromRunningOrder(referenceFlineId);

 if( runningLine == null ) {
 // We have an error condition. No fline found with referenceFlineId.
 throw new ValidationException("Something's not right. Couldn't find line using reference fline id.");
 }

 if (runningLine.getAttribute("FulfillLineStatusCode") == "CLOSED" ||
 runningLine.getAttribute("FulfillLineCanceledFlag") == "Y" || //Line is cancelled.
 runningLine.getAttribute("FulfillLineShippedQty") != null || //Line is shipped.
 runningLine.getAttribute("FulfillLineInvoiceInterfacedFlag") == "Y" ){ //Line is interfaced to invoicing.
 // This line isn't valid for setting default values.
 continue;
 }
 }
 else {
 // This sales order doesn't have a revision.
 //Its ok to set the default value for attributes.
 }
 //Put your defaulting logic here.
 //line.setAttribute(<attribute name>, <value>);
```

```
}

Object getLinesFromRunningOrder(Long runningLineId) {

 // Create an instance of the FulfillLinePVO public view object (PVO).
 def flinePVO = context.getViewObject("oracle.apps.scm.doo.publicView.analytics.FulfillLinePVO");

 // Create a view criteria object.
 def vc = flinePVO.createViewCriteria();

 // Create a view criteria row.
 def vcrow = vc.createViewCriteriaRow();

 // Set query conditions on the view criteria row.
 vcrow.setAttribute("FulfillLineId", runningLineId);
 vc.add(vcrow);

 def rowset = flinePVO.findByViewCriteriaWithBindVars(vc, -1, new String [0], new String [0]);

 if (rowset.hasNext()) {
 def fline = rowset.first();
 return fline;
 }
}
```

Create Filters in Business Rules

Write a business rule that filters fulfillment lines, such as in a pretransformation rule, posttransformation rule, line selection criteria, or assignment rule.

Here's an example line selection criteria.



The line selection criteria filters out lines that can't ship so Order Management only sends shippable lines to the fulfillment system that processes shippable lines, such as your shipping system or Oracle Global Order Promising.

Note

- Use the Manage Orchestration Process Definitions task in the Setup and Maintenance work area.
- Use the line selection criteria to add the rule.
- Add the rule to each orchestration process step that shippable affects, as necessary. For example, if the item is a warranty, then it isn't shippable. To filter out the line that isn't shippable, you probably want to add the rule to each step that references a schedule, reserve, or ship task, such as the Schedule step, the Create Reservation step, Create Shipment Request step, and so on.

- The entire If statement isn't visible in the screen capture. Here are the entire statements:
    - `DooSeededOrchestrationRules.DOOFLine.categoryCode is "ORDER"`
    - `DooSeededOrchestrationRules.DOOFLine.shippableFlag is "Y"`
- You can't use Visual Information Builder to select fulfillment lines. You must edit the orchestration process and use Oracle Business Rules.
- If the nonKitModelFlag attribute contains Y on a line selection criteria, and if the order line contains a configuration model, and if the model contains a kit, then the rule will select the line and process it.
    - Here's the If statement for the rule.

      ```
      If DooSeededOrchestrationRules.DOOFLine.nonKitModelFlag is Y
      ```
    - nonKitModelFlag is a temporary attribute. Its data exists only in active, working memory, so you can't access it later in downstream fulfillment.

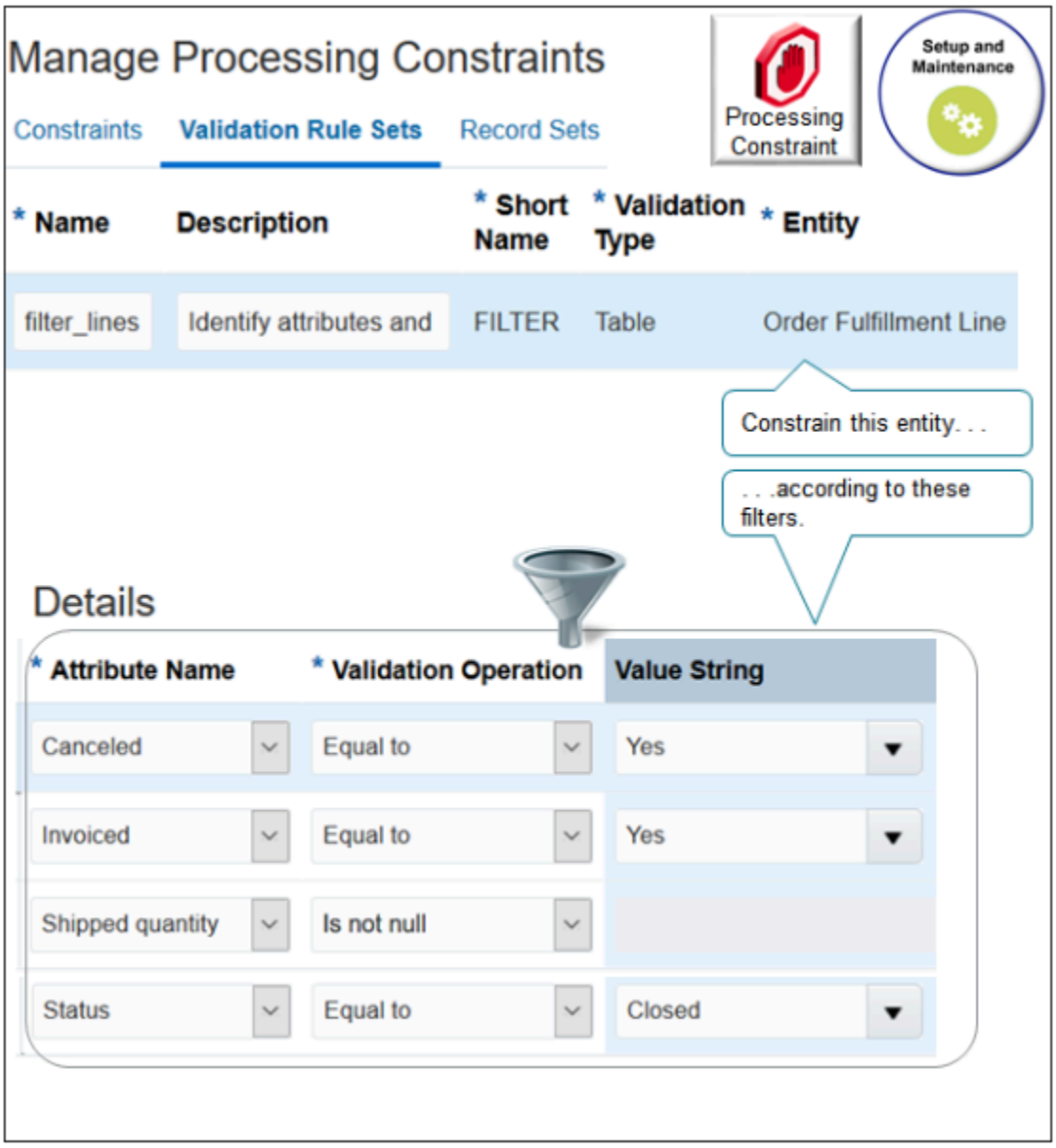
Example of a Pretransformation Rule

Write a pretransformation rule where you set the value for an attribute, but only after you filter lines. For example:

- If the order line isn't closed, canceled, shipped, or already sent to billing, then set the default value for the requested ship date attribute.

For more about posttransformation rules, see *Overview of Transformation Rules*.

Constrain Changes That You Allow on Fulfillment Lines



Note

- Use the Manage Processing Constraints task in the Setup and Maintenance work area.
- Set the Entity attribute to Order Fulfillment Line.

- Use the Details area to add your filters.

  Assume that if the fulfillment line is closed or canceled, or if Order Management already shipped it or invoiced it, then you don't want your users to modify a value on the line, such as Ship-to Site.

Attribute Name	Validation Operation	Value String
Canceled	Equal To	Yes
Invoiced	Equal To	Yes
Shipped Quantity	Is Not Null	-
Status	Equal To	Closed

You can also create a constraint that prevents the user from submitting a sales order that doesn't include payment terms for lines that meet a specific criteria. For example, the line isn't a return line, or the line is for a transfer. For details, see *Display Payment Terms on Sales Orders*.

Related Topics

- Overview of Creating Order Management Extensions
- Processing Constraints
- Overview of Orchestration Processes
- Select Fulfillment Lines for Orchestration Process Steps
- Another Example of Using Extensible Flexfields In Line-Selection Rules

Upstream

Fix Connection Problems with Source Systems

Troubleshoot an error message that indicates Order Management can't connect to a source system, such as Oracle Configure, Price, and Quote.

1. Go to the Order Management work area, then click **Tasks > Manage Order Orchestration Messages**.
2. On the Manage Order Orchestration Messages page, set the Order Orchestration Function attribute to Send Event Notification, then click **Search**.

3. Examine the results.
    - Make sure the search results displays an entry that includes the same URL you specified in the Connector URL attribute of page Manage Connector Details. For details, see *Integrate Order Management with Source Systems*.
    - Make sure the URL correctly identifies the connector that resides on the source system. To do this, sign into your source system, then examine the connector services that are running.
    - Wait a minute for the log to refresh, requery the Order Orchestration Messages page, then examine the search results again to determine whether the connection successfully restarted. A network error or some other problem might cause the connection to momentarily fail. If the connection restarted successfully, then the list will include details about the events that are associated with the connector URL.

Related Topics

- Integrate Order Management with Source Systems

Downstream

Troubleshoot Problems in Downstream Fulfillment

Fix problems in your Order Management implementation that happen in downstream fulfillment.

Common Errors

Trouble	Shoot
I encounter an error message. `Order promising can't schedule the item because the requested date happens after the order promising horizon.`	Set the Requested Ship Date on the sales order so there's enough time to schedule, ship, and deliver the item. As an alternative: 1. Go to the Setup and Maintenance work area, then open the Manage Administrator Profile Values task. 2. Search the Profile Option Code attribute for MSP_GOP_HORIZON_DAYS. 3. Change the Profile Value attribute so the end of the horizon happens after the requested ship date. 4. Collect data and refresh the Order Promising server.
The price on my order line in Order Management is different than the price on the invoice in Accounts Receivable. It's frequently off by a dollar or less.	Create a rounding rule so that Pricing rounds the value before Order Management sends it to Accounts Receivable. For details, see *Manage Rounding Rules*. If you create a rounding rule for this purpose, and if you import your sales order, then you must set the Freeze Price attribute to N in your import payload. If its Y, then the prices are frozen and Pricing won't apply the rounding rule. For details, see *Freeze Price on Sales Orders*.
I can't release a hold on a fulfillment line even after I release the credit check hold and approve the order. I get an error that's similar to: `No reservation is defined for the orchestration process`	This problem might happen when your orchestration process has a reservation task but you don't actually use that task. Examine your orchestration process definition and remove any reservation tasks that you aren't using.

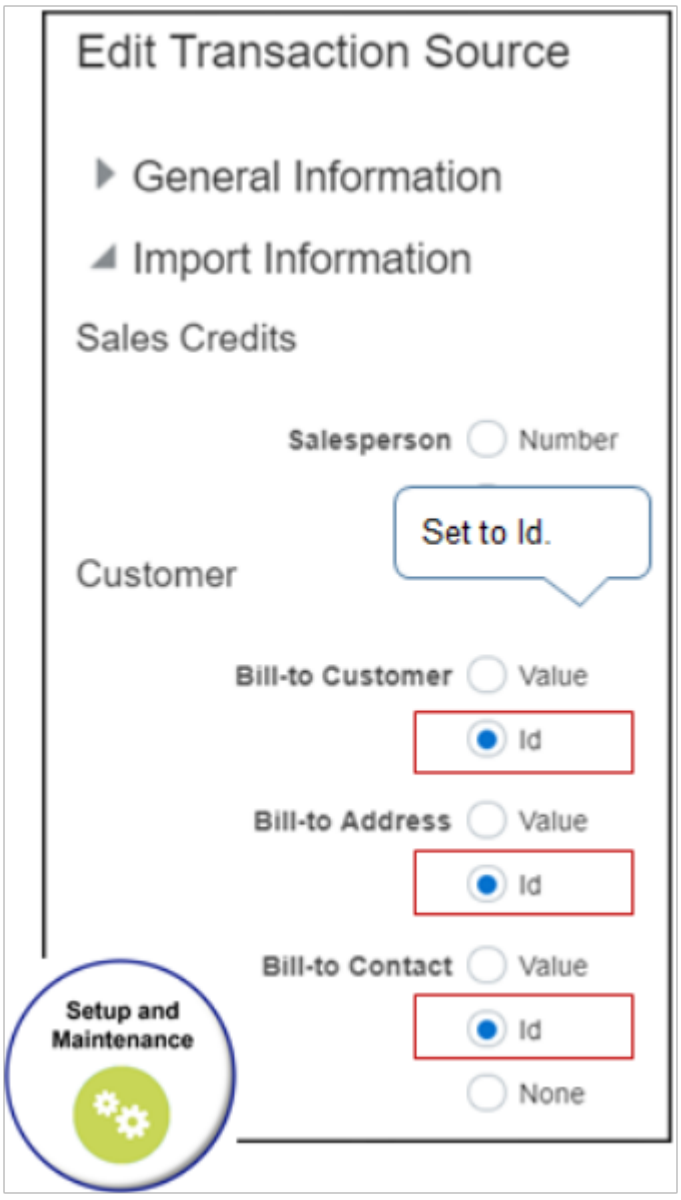
Trouble	Shoot
I use a web service to create a sales order and then update the price, but I encounter an error that's similar to one of these errors: `An order was not created because a value was not provided for the required attribute SOURCE_CHARGE_COMPONENT_ID in the source order with the following details: source order 132947, source order line 1, source order schedule 1, source order charge 300001671561802, source order charge component. Provide the required value, and resubmit the order.`  `The attribute SourceChargeComponentId is missing for service TaskLayerResponseAM.`  `The attribute HeaderCurrencyUnitPrice is missing for service TaskLayerResponseAM.` This problem might happen when you don't include a value for the SourceChargeComponentIdentifier attribute and the HeaderCurrencyUnitPrice attribute in your response payload.	You must include these values. If you're creating a return, then you must include these values from the original sales order. For details, see *Create Your Own Task Type*.

I Have a Billing Problem

Trouble	Shoot
I have an item that I normally ship, such as the AS54888 Desktop Computer. I have a unique need to invoice the item but not actually ship it. I enable attributes for my item. • I enable the Stock attribute and the Shippable attribute for the AS54888 item in the master organization. • I enable the Nonstock attribute and Nonshippable attribute for the AS54888 item in the inventory organization. I create a sales order, add the AS54888 item, then click Submit. The orchestration process skips the Shipping step because the inventory organization says the AS54888 isn't stockable or shippable, but	This problem happens because the fulfillment line in Order Management gets attribute values for the item from the master organization in the Product Information Management work area. It doesn't get attribute values from the inventory organization in the Inventory Management work area. To fix the problem, create an assignment rule that assigns the item to an orchestration process that doesn't ship but that does do billing. For example: `If Item = AS54888, then Do Process Name is set to DOO_BillonlyGenericProcess`

Trouble	Shoot
then the line gets stuck in the Awaiting Shipping status. Sometimes I encounter an error. **`The value provided for the item ID attribute is invalid. Value provided: 100000046397982.`** This error happens on the shipping step of my orchestration process.	

I encounter an error when I run the Import AutoInvoice scheduled process.
 	
`You must enter a value in either the Original System Bill-to Customer Reference column or the Bill-to Customer Account Number column.`
	This error happens when you set the value instead of the Id during set up. Try this:
	
	1. Make sure you have the privileges that you need to administer the Project Financial Management offering.
	2. Go to the Setup and Maintenance work area, then go to the task.
	   - Offering: Project Financial Management
	   - Functional Area: Project Billing Base
	   - Task: Manage Transaction Sources
	3. On the Manage Transaction Sources page, in the Name attribute, search for, then open Distributed Order Orchestration.

Trouble	Shoot
	4. On the Edit Transaction Source page, in the Customer area, set the option to Id for these attributes. o Bill-to Customer o Bill-to Address o Bill-to Contact  5. Run the Import AutoInvoice scheduled process again. For details about the scheduled process, see *Guidelines for Setting Up Order-to-Cash* and *Update Intercompany Receivables Invoice Import Details*.
I encounter an error.	This error happens when the user doesn't set the payment term on the order header or order line, and the customer account setup doesn't set the payment term.

Trouble	Shoot
`Data validation failed for Create Billing Lines fulfillment task because required data values were not provided. The payment term in the fulfillment line is null. Source Order Information: xxxxxx`	Set the payment term when you set up the account. Order Management typically sets the payment term from the customer account setup when the user creates a sales order, by default. Learn how to specify the payment term. For details, see *Overview of Displaying Customer Details on Sales Orders*. As an alternative, instruct your users to create an order revision, then set the payment term on the order header or order line.

Deactivating Customer Data After You Submit a Sales Order

If you deactivate customer data on a sales order after you submit it, and if you need to change something on the order, then you might need to wait for Order Management to fulfill and close all lines that reference the deactivated customer data.

For example, assume you:

1. Create a sales order:
    - You set the Bill-to Customer to Computer Supply and Rentals.
    - You add two order lines.
2. You submit the order.
    - Order Management sends order line 1 to Accounts Receivable and the line's status is Awaiting Billing.
    - Order line 2 is in the Manual Shipping status.
3. You deactivate the Computer Supply and Rentals bill-to customer in Oracle Customer Data Management.
4. You need to modify the quantity on line 2, so you create an order revision but encounter an error when you submit the revision.

`THE VALUE XXX GIVEN FOR ATTRIBUTE BILL-TO CUSTOMER WHICH IS DEPENDENT ON ATTRIBUTE BILL-TO CONTACT WITH VALUE YYY IS INVALID`

This error happens because Order Management attempts to validate the Computer Supply and Rentals bill-to customer when you submit the revision, but finds that Computer Supply and Rentals is no longer active. A similar error might happen if you deactivate the ship-to customer.

To avoid this problem, wait until all of these lines on the order are in Closed status, then create a revision.

I Can't Convert Currency During Billing

I encounter an error.

`The conversion type is invalid. If the currency of the transaction is the same as the ledger currency, enter User and set CONVERSION_RATE to 1.`

This problem happens when you don't set up a value that Order Management can use as the default for the conversion rate type. Order Management uses the conversion rate type when it converts a foreign currency transaction to the ledger currency.

1. Go to the Setup and Maintenance work area, click **Tasks > Search**, then search for and open the Manage Administrator Profile Values task.
2. On the Manage Administrator Profile Values page, search the Profile Option Code attribute for AR_DEFAULT_EXCHANGE_RATE_TYPE.
3. In the Profile Values area, add a value. Make sure you set the Profile Level attribute to Site.

If you're using an order management extension to do the conversion, then use this extension code instead of setting the profile value.

```
import oracle.apps.scm.doo.common.extensions.ValidationException;

import oracle.apps.scm.doo.common.extensions.Message;

def orgId = header.getAttribute("BusinessUnitIdentifier");
def buPrimaryLedgerId=getBUPrimaryLedgerId(orgId);
def ledgerCurrCode=getLedgerCurrency(buPrimaryLedgerId);

def headerCurrCode=header.getAttribute("TransactionalCurrencyCode");

if(!headerCurrCode.equalsIgnoreCase(ledgerCurrCode) &&
 header.getAttribute("CurrencyConversionTypeCode")==null){
String messageText = "Currency Conversion Type must be specified on the Order Header when the Sales Order
 Currency and the Currency associated to the Selling Business Unit is not the same";
List messages = new ArrayList();
messages.add(new Message(Message.MessageType.ERROR, messageText));
ValidationException ex = new ValidationException(messages);
throw ex;
}

Object getBUPrimaryLedgerId(Long orgId) {
def buPrimaryLedgerId;
def
 bussUnitPVO=context.getViewObject("oracle.apps.financials.commonModules.businessUnits.publicView.BusinessUnitPVO");
def buvc = bussUnitPVO.createViewCriteria();
def buvcRow = buvc.createViewCriteriaRow();
buvcRow.setAttribute("BusinessUnitId",orgId);
def buRowset = bussUnitPVO.findByViewCriteria(buvc, -1);
if(buRowset.hasNext()){
def buRow = buRowset.next();
buPrimaryLedgerId=buRow.getAttribute("PrimaryLedgerId");
}
return buPrimaryLedgerId;
}

Object getLedgerCurrency(String ledgerId) {
def ledgerCurrCode;
def
 ledgerPVO=context.getViewObject("oracle.apps.financials.generalLedger.ledgers.ledgerDefinitions.publicView.LedgerPV
def lvc = ledgerPVO.createViewCriteria();
def lvcRow = lvc.createViewCriteriaRow();
lvcRow.setAttribute("LedgerId",ledgerId);
def lRowset = ledgerPVO.findByViewCriteria(lvc, -1);
if(lRowset.hasNext()){
def ledgerRow = lRowset.next();
ledgerCurrCode=ledgerRow.getAttribute("CurrencyCode");
}
return ledgerCurrCode;
}
```

I Have a Problem with Sales Order Status

Trouble	Shoot
I create a sales order and submit it. Supply Chain Orchestration creates the supply order and reserves the item in inventory. Assume today is January 1, 2021. Global Order Promising examines your supply	If the Supply Type attribute on the line for the supply order is: • **ATP**. Promising examines the on-hand inventory and uses it to schedule the order line. You can use the Supply Orchestration work area to reserve inventory and pick the item.

Trouble	Shoot
chain, determines that it can't fulfill the item until January 7, and communicates these details to Order Management. Order Management sets the Scheduled Ship Date attribute on the fulfillment line to January 7, and changes the Status attribute on the order line to Waiting. I expected the status on the order line to be Awaiting Shipping, not Waiting. I go to the Inventory Management work area, select Shipments > Manage Shipment Lines, but can't find shipment lines for the sales order. Here's what happened. If the Scheduled Ship Date attribute on the fulfillment line happens 5 or more days after the current date in a back-to-back flow, then the orchestration process sets the Status attribute on the order line to Waiting.	• **Buy**. Wait until the purchase order is finished, then use Supply Chain Orchestration to reserve inventory and pick the item. For details, see *Manage Your Supply Process*.
We have a custom orchestration process that sends a request to some other system during compensation, such as sending a request for a credit check to a 3rd party vendor, sending a screening request to trade compliance, and so on. At run time, I revise a sales order, the orchestration process runs, but the order gets stuck in Change Pending status. This problem might happen because Order Management never receives a reply from the request.	• There's most likely a problem in your downstream system. Fix the problem, then verify that it can send a reply. • Use the Recover Process action. For details, see *Fix Errors in Sales Orders*. • If you must modify the sales order, then cancel the line that's stuck, create a new line, and submit the order. For details, see *Cancel Order Lines That Remain in the Same Status*.

The Warehouse Hasn't Reserved Inventory for My Item

Assume you create a sales order, submit it, and have a fulfillment line with these values.

Attribute	Value
Item	AS54888
Quantity	100

You go to a fulfillment view in the Order Management work area, and see that there's an error on the fulfillment line.

```
A reservation was not created because the reservation quantity is greater than the available-to-reserve quantity.
```

Several problems might cause this error. Here are some corrections you can try.

Change the Warehouse

Go to the Order Management work area, open your sales order, then use the Warehouse attribute on the fulfillment line to change the warehouse.

Modify an Available-To-Promise Rule

Assume set up an infinite available-to-promise rule to schedule the order line. However, at run time there's no on-hand quantity in the sourcing warehouse.

Order Promising doesn't track actual supply when you use an infinite available-to-promise rule. It assumes that supply is always available. To fix this problem, you can remove the infinite available-to-promise rule.

Modify a Sourcing Rule

Assume you create a sourcing rule but didn't set the From Quantity or Less Than Quantity attributes. You might have set these values in the Order Promising work area or during order import.

Here's how you can fix this problem.

1. Go to the Global Order Promising work area.
2. On the Global Order Promising page, click **Tasks > Manage Sourcing Rules**.
3. On the Manage Sourcing Rules page, search for, then open your sourcing rule for editing.
4. On the Edit Sourcing Rules page, in the Effective Start Date, click **View > Columns**, then add a check mark to the From Quantity option and the Less Than Quantity option.
5. Set the values.

Attribute	Value
From Quantity	0
Less Than Quantity	9999999

These values will work for most implementations, but you can use different ones to meet your requirements. For details, see *Create Sourcing Rules That Source According to Quantity and Supplier*.

If you import the sourcing rule, then set the values in your import payload instead of using the work area.

Attribute	Value
FROM_QTY	0
TO_QTY	9999999

6. Go to the Scheduled Processes work area, then run the *Refresh and Start the Order Promising Server* scheduled process.

Modify the Orchestration Process

You can remove the reservation step from you orchestration process and avoid the error message. The sales order will go to the Awaiting Shipping status even though there isn't any on-hand quantity available. If you remove the step, then you must manually reserve inventory for each sales order.

I Can't Update My Sales Credits

I encounter an error:

```
You do not have the required access to update line attribute "Sales Credit". Source Order Information:
36919901-3-1
```

Assume you use SourceTransactionSalesCreditIdentifier in an extension to populate the flineId for all sales credits that are in the same flineId. If you have more than one sales credit on the same fulfillment line, then the extension will create more than one record for SourceTransactionSalesCreditIdentifier, and the extension can't tell which credit to apply to the line.

To avoid this problem, you can use SalesCreditIdentifier to explicitly populate the value in SourceTransactionSalesCreditIdentifier:

```
sc.setAttribute("SourceTransactionSalesCreditIdentifier", sc.getAttribute("SalesCreditIdentifier"));
```

For example:

Bad	Good
`def sc = salesCredits.createRow(); sc.setAttribute("SourceTransactic flineId); sc.setAttribute("SalespersonIdent 5507); sc.setAttribute("Percent", new BigDecimal(50)); sc.setAttribute("SalesCreditTypeC 1); salesCredits.insertRow(sc);`	`def sc = salesCredits.createRow(); sc.setAttribute("SalespersonIdentifier", 5507); sc.setAttribute("Percent", new BigDecimal(50)); sc.setAttribute("SalesCreditTypeCode", 1); salesCredits.insertRow(sc); sc.setAttribute("SourceTransactionSalesCreditIdentifier", sc.getAttribute("SalesCreditIdentifier"));`

Related Topics

- Overview of Importing Orders Into Order Management
- Set Up User Roles and Privileges in Order Management
- Use SQL to Query Order Management Data
- Troubleshoot Problems with Configure-to-Order

Fix an Order Status That Doesn't Update to Shipped

Fix a problem where your fulfillment system confirms shipment, but the order status remains at Awaiting Shipping instead of updating to Shipped.

You might also encounter an error message in the Inventory Management work area when you use the Manage Pending Transactions page. You click a link for an error transaction, click the Edit icon, make your corrections, Click Add All to Process Schedule, and encounter the error:

```
At least one of the selected transactions was not added to the process schedule. (INV-2415775) Details: The
transaction was not added because it was not ready for processing, was already in the process schedule, was
locked, was not in confirmed status, or was updated by another user. The On-hand quantity is not affected
for the quantity shipped.
```

Process Responses from Order Fulfillment

The first fix you can try is to run the Process Responses from Order Fulfillment scheduled process. This process sends shipment and invoice details to Order Management, and Order Management uses them to update the order status.

1. Run the scheduled process and wait for it to finish. For details, see the Process Responses from Order Fulfillment subtopic in *Import and Fulfill Large Volumes of Sales Orders*.
2. Examine the sales order and confirm the order status is Shipped.
3. If resubmitting the scheduled processes doesn't fix the problem, then modify your shipment setup.

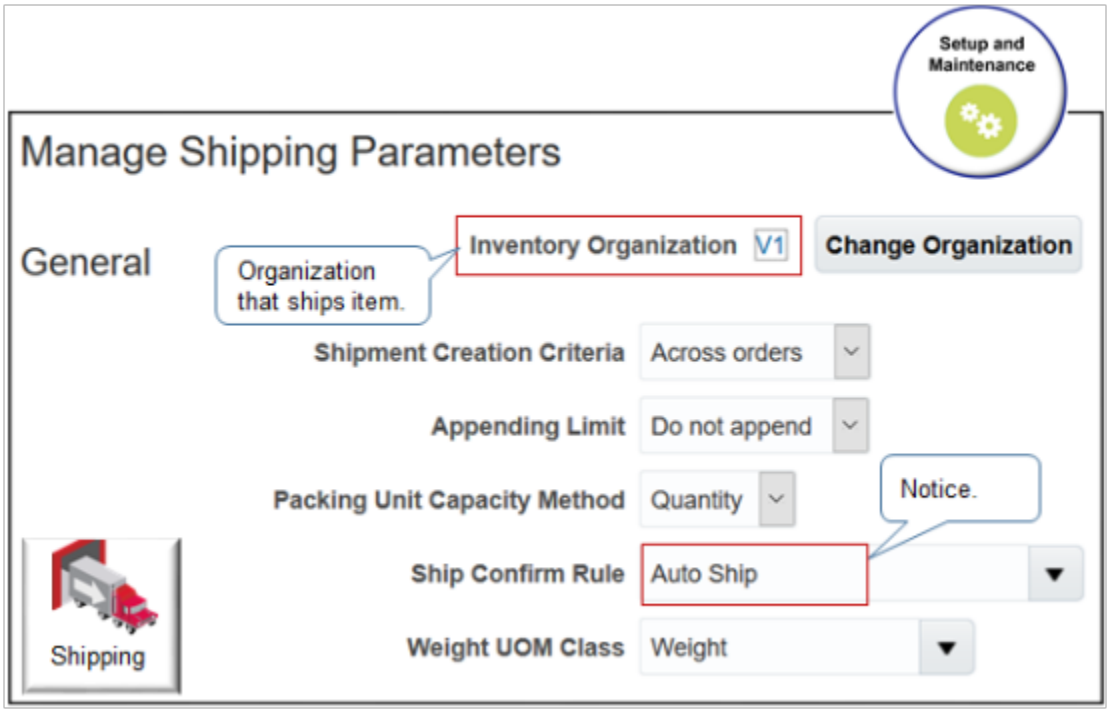
Modify Your Shipment Setup

To make sure Order Management updates the order status, you must make sure you interface the shipment to inventory, and that the shipment is closed.

For this example, assume Vision Operations (V1) is your inventory organization that fulfills the shipment.

1. Identify the ship confirm rule that your inventory organization uses.
    - Go to the Setup and Maintenance work area, then go to the task.
        - Offering: Manufacturing and Supply Chain Materials Management
        - Functional Area: Shipping
        - Task: Manage Shipping Parameters
    - Set organization to Vision Operations.
    - On the Manage Shipping Parameters page, notice the value in the Ship Confirm Rule attribute.

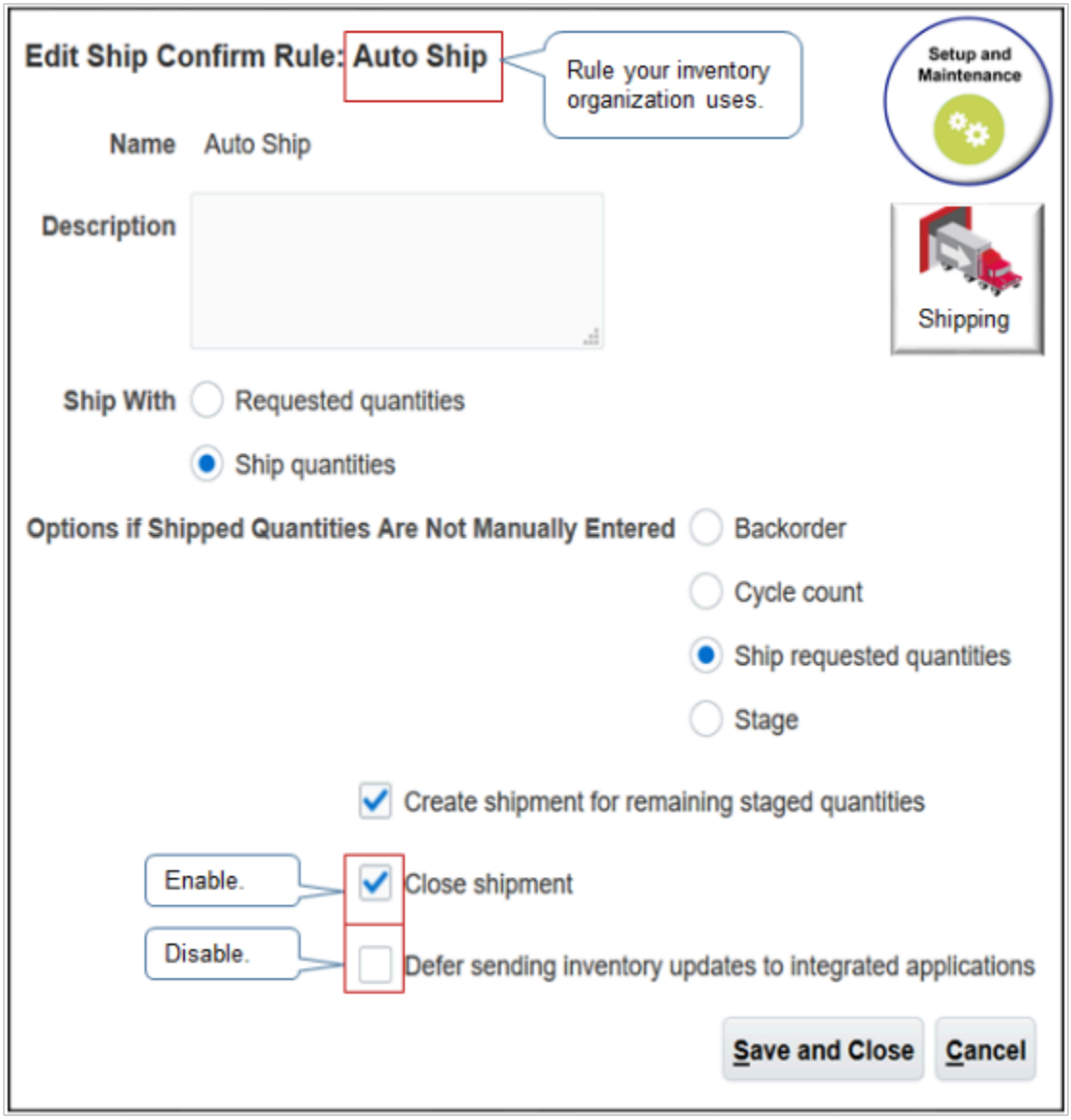
      For this example, assume the value is Auto Ship.



2. Modify the ship confirm rule.
    - In the Setup and Maintenance work area, go to the task.
        - Offering: Manufacturing and Supply Chain Materials Management
        - Functional Area: Shipping
        - Task: Manage Ship Confirm Rules
    - On the Manage Ship Confirm Rules page, search for Auto Ship, then open it for editing.
    - In the Edit Ship Confirm Rule dialog, set the values, then click Save and Close.

Option	Value
Close Shipment	Enabled
Defer Sending Inventory Updates to Integrated Applications	Disabled

Option	Value



3. Go to the Scheduled Processes work area, then run the *Manage Shipment Interface* scheduled process.

Parameter	Value
Mode	All
From Shipment	4028
To Shipment	4028

Parameter	Value
Ship-from Organization	M1 Manufacturing Select the organization that ships the item.

4. Run the Send Shipment Advice scheduled process. See the section earlier in this topic for details.

Modify Options on the Shipment

As an alternative, you can modify the options on the shipment.

1. Go to the Inventory Management work area.
2. On the Inventory Management page, click **Tasks > Show Tasks Shipments > Manage Shipments**.
3. On the Manage Shipments page, search for your shipment, such as 4028.
4. In the search results, click the row that contains your shipment, then click **Actions > Edit**.
5. On the Edit Shipment page, near the upper right corner, click **Actions > Change Ship Confirm Options**.
6. In the dialog that displays, enable and disable the same options that you set for the ship confirm rule.
7. Run the Manage Shipment Interface scheduled process and the Send Shipment Advice scheduled process.

Still Not Working?

1. Run an SQL on the sales order. Assume its order 4028.

```
select distinct dha.source_order_number,
DMT.message_text,
DMB.message_type
from doo_headers_all dha,
FUSION.DOO_MESSAGE_ENTITIES DME,
DOO_MESSAGES_B DMB,
DOO_MESSAGES_TL DMT
where dha.header_id = DME.entity_id
and DME.entity_name = 'ORDER'
and DMB.msg_request_id = DME.msg_request_id
and DMT.message_id = DMB.message_id
and dha.source_order_number = '4028'
```

   For details, see *Use SQL to Query Order Management Data*.
2. Do you receive an error like this?

```
A cross-referenced value was not found for attribute ShippingCarrierCode in source system OPS.
```

   If you do, continue this procedure.
3. Collect data.
   - Do the Collect Planning Data task.

Attribute	Value
Collection Type	Targeted

Attribute	Value
Selected Entities	Shipping Methods

- Run the *Refresh and Start the Order Promising Server* scheduled process.

For details, see *Collect Planning Data for Order Management*.

4. Run the Send Shipment Advice scheduled process.
5. If the status on the order line is Backordered, then run the *Send Intermediate Shipment Status Update* scheduled process. Set it to run periodically on a schedule so that it eventually updates the line after the line proceeds out of Backordered status.

What Does Partially Closed Mean, and How Do I Fix That?

A sales order status can be Partially Closed. It means that at least one order line in the order is closed, but not all of them are closed.

Assume your sales order 57693 has three lines.

- Line 1 is Closed.
- Line 2 is Awaiting Shipping.
- Line 3 is Closed.

The status for sales order 57693 will be Partially Closed.

You can't manually set the status to Partially Closed. The *Update or Close Sales Orders* scheduled process does that for you automatically when it finds a sales order that has at least one line that's closed but all lines aren't closed.

If you have a sales order that's in Partially Closed status and you want it to be closed, try running Update or Close Sales Orders manually. It could be that changes have occurred since the last time Update or Close Sales Orders ran according to its automatic schedule. For details, see *Update or Close Fulfillment Lines That Remain Open*.

Related Topics

- Set Up User Roles and Privileges in Order Management
- Use SQL to Query Order Management Data
- Collect Planning Data for Order Management

Update or Close Fulfillment Lines That Remain Open

Close a fulfillment line that remains open when it should be closed.

Order Management closes each fulfillment line when it finishes all orchestration process steps of the orchestration process. In some situations, a step might indicate that fulfillment is done, such as shipping is done for a shipping step. The indicator might mean your organization considers fulfillment done when the orchestration process reaches this step, but the fulfillment line is still open.

If you haven't run the *Update or Close Sales Orders* scheduled process since Order Management closed the fulfillment line, then the Order Management work area might display sales orders and order lines as open even if it closed all the fulfillment lines that these orders and lines reference. You can run the scheduled process to fix this problem.

- We recommend that you set up the scheduled process so it runs automatically according to a schedule, such as every 12 hours.
- The scheduled process also updates the status on the order header.
- The scheduled process updates the SHIPPED_QUANTITY, SCHEDULE_SHIP_DATE, and ACTUAL_SHIP_DATE columns in the DOO_LINES_ALL table. If you don't run the process, then the status on the order line might display as Closed in the Order Management work area but as Open in DOO_LINES_ALL, or the Scheduled Ship Date on the line might contain a value in the work area but not in DOO_LINES_ALL.
- If you don't run the scheduled process, then the data that you use in various integrations, such as reports and analytics, might not reflect the latest changes that you see in the Order Management work area.
- Make sure all instances of the Update or Close Sales Orders scheduled process are currently in the Succeeded status or the Error status. If you find a process that isn't in one of these statuses, such as Retrying, then you must wait for that instance to finish before you start a new instance.
- If you encounter a problem, try running the process without a value in the Interval Hours parameter.

The scheduled process updates the order status depending on the order line status.

Order Line Status	What the Process Does
All lines in the sales order are in the Canceled status.	Sets the order status to Canceled.
At least one line in order is in the Closed status and all other lines are in the Canceled or Closed status.	Sets the order status to Closed.

The scheduled process gets attribute values from the fulfillment line and uses them to update attributes on the order line.

Order Line Attribute	Fulfillment Line Attribute
Shipped Quantity	Shipped Quantity
Delivered Quantity	Delivered Quantity
Fulfilled Quantity	Fulfilled Quantity
Actual Ship Date	Actual Ship Date
Scheduled Ship Date	Scheduled Ship Date
Fulfillment Date	Fulfillment Date

These attribute names are slightly different in your business rules and web service payloads.

Order Line Attribute	Fulfillment Line Attribute
ShippedQty	SumShippedQty
RmaDeliveredQty	SumRmaDeliveredQty
FulfilledQty	SumFulfilledQty
ActualShipDate	MaxActualShipDate
ScheduleShipDate	MaxScheduleShipDate
FulfillmentDate	MaxFulfillmentDate

Try It

Assume you administer a call center that's open 9 AM to 5 PM, and you must process all order headers that Order Management creates during the open hours. You also must run the process every day for the 2021 fiscal year, and run the process after the call center closes.

1. Verify that the orchestration process successfully finished processing the fulfillment line.
    - Go to the Order Management work area, search for, and open your sales order.
    - On the Overview page, click **Actions > Switch to Fulfillment View**.
    - Click the **Fulfillment Lines** tab.
    - In the Fulfillment Line Details area, on the General tab, verify that the Status attribute contains Closed.
    - Click the value in the Orchestration Process Number attribute, such as 300100541247223.
    - On the Orchestration Process page, in the Orchestration Plan area, verify that the last row in the Task Progress column contains a green check mark.
2. Sign into Oracle Applications with the credentials that you need to run the scheduled process.

Credential	Value
Job role	Order Manager (ORA_DOO_ORDER_MANAGER_JOB
Privilege	Plan Orchestration Processes (DOO_PLAN_ORCHESTRATION_PROCESSES_PRIV)
Duty Role	Orchestration Order Management (ORA_DOO_ORCHESTRATION_ORDER_MANAGEMENT_DUTY)

   This topic uses predefined job roles. You must create your own job roles, depending on your security requirements. For details, see *Privileges That You Need to Implement Order Management*.
3. Go to the Scheduled Processes work area.

4. On the Scheduled Process page, click **Actions > Schedule New Process**.
5. In the Schedule New Process dialog, click the **down arrow**, click **Search**, then search for the value.

Parameter	Value
Name	Update or Close Sales Orders

6. Click **OK > OK**.
7. Make sure at least one sales order is open and that it meets the values that you set in the next step. If no sales order meets the criteria that you set, then the process might return an error.
8. In the Process Details dialog, set the values.

Parameter	Description
Entity Name	Enter the text **`Header`** or **`Line`**: - **Header**. For sales order. - **Line**. For order line. For this example, enter **`Header`**.
Entity ID	To process one: - Order header, enter the ID for the order header. - Order line, enter the ID for the order line. If you don't enter a From Creation Date, To Creation Date, or interval hours, then you must enter an entity ID. You can run a query to get the entity ID. For this example, leave Entity ID empty.
From Creation Date and To Creation Date	Use From Creation Date and To Creation Date to specify the time period to consider when updating or closing the entity. - **From Creation Date**. Specify the earliest time and date when Order Management created the entity. - **To Creation Date**. Specify the latest time and date when Order Management created the entity. For example, if you set Header as the entity, and if you set: - Noon yesterday as the From Creation Date, and 4 PM yesterday as the To Creation Date, then the scheduled process will update all the order headers that Order Management created yesterday on or after Noon and on or before 4 PM. - Noon yesterday as the From Creation Date but don't specify the To Creation Date, then the scheduled process will update all order headers that Order Management created on or after Noon yesterday up to the current time. If you enter interval hours, then don't enter a From Creation Date or To Creation Date. For this example, leave these attributes empty.

Parameter	Description
Interval Hours	Period of time in hours to count backward from the time when the process starts. The scheduled processes uses this interval to identify the entities that it processes. For example, if you enter 8 for interval hours, and if you enter Header, and if you leave all other attributes empty, then the scheduled process will process all the entities that Order Management created in the eight hours immediately before you click Submit. For this example, enter 10 in the Interval. The call center is open 9 to 5, which is 8 hours. You will set the scheduled process to start at 5:30 so the Order Entry Specialists can finish creating any sales orders that they're currently adding. Using 10 as the interval will pick up these sales orders, and also any sales orders that you create after 7:30 AM.

All parameters are optional except for Entity Name.
For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

9. Click **Advanced > Using a Schedule**, then set the values.

Attribute	Value
Frequency	Daily
Every	1 Day
From Creation Date	1/1/21 5:30 PM
To Creation Date	12/31/21 5:30 PM

10. Click **Submit**.

Guidelines to Set Your Dates and Intervals

You can use these combinations:

- From Creation Date and To Creation Date
- Only the From Creation Date
- Only the Interval

You can't use only the:

- From Creation Date and the Interval
- To Creation Date and the Interval
- To Creation Date

You can't leave From Creation Date, and To Creation Date, and the Interval empty.

If you don't follow these guidelines, you can still submit the scheduled process, but it will fail.

Query to Get the Entity ID

Here's some SQL that you can use to get the entity Id.

```
SELECT DISTINCT
lines.line_id,
lines.creation_date,
lines.header_id
FROM
fusion.doo_process_instances pi,
fusion.doo_orchestration_groups og,
fusion.doo_lines_all lines
WHERE
og.doo_process_instance_id = pi.doo_process_instance_id
AND og.status = 'ACTIVE'
AND og.line_id = lines.line_id
AND pi.process_active IN (
'COMPLETED',
'INACTIVE'
)
AND lines.open_flag = 'Y'
AND lines.header_id IN (
SELECT
nvl(header_id, 0)
FROM
doo_headers_all
WHERE
creation_date BETWEEN :currstartdate AND :currenddate
)
UNION
SELECT DISTINCT
lines.line_id,
lines.creation_date,
lines.header_id
FROM
fusion.doo_lines_all lines,
fusion.doo_fulfill_lines_all flines
WHERE
lines.line_id = flines.line_id
AND lines.open_flag = 'Y'
AND flines.canceled_flag = 'Y'
AND flines.open_flag = 'N'
AND lines.header_id IN (
SELECT
nvl(header_id, 0)
FROM
doo_headers_all
WHERE
creation_date BETWEEN :currstartdate AND :currenddate
)

=========
HeaderWithCompletedProcessInstancesVO
---------------------

SELECT DISTINCT
headers.header_id,
headers.creation_date
FROM
doo_process_instances pi,
doo_orchestration_groups og,
doo_headers_all headers
WHERE
```

```
og.doo_process_instance_id = pi.doo_process_instance_id
AND og.status = 'ACTIVE'
AND og.header_id = headers.header_id
AND headers.open_flag = 'Y'
AND pi.process_active IN (
'COMPLETED',
'INACTIVE'
)
UNION
SELECT DISTINCT
headers.header_id,
headers.creation_date
FROM
doo_lines_all lines,
doo_headers_all headers
WHERE
headers.open_flag = 'Y'
AND headers.header_id = lines.header_id
AND lines.canceled_flag = 'Y';
```

Resolve Problems with the Warehouse Attribute on Sales Orders

Resolve a problem where the Warehouse attribute on the order line is empty or doesn't contain the warehouse that you need to pick.

This topic includes a series of remedies you can try to fix the problem.

Troubleshoot Global Order Promising

This problem typically happens because there's a problem with Global Order Promising. For details, see *Troubleshoot Global Order Promising for Order Management*.

Collect Data for the Organization

Forgetting to collect data for the organization that contains the inventory that you use to fulfill the item is another frequent cause of the problem.

Assume you use Oracle Order Orchestration and Planning as your destination system. Assume Vision Manufacturing is the source system that contains inventory for the AS54888 item.

Make sure you collected the Vision Manufacturing organization.

1. Go to the Plan Inputs work area, then click **Tasks > Maintain Supply Network Model**.
2. On the Maintain Supply Network Model page, search for the value.

Attribute	Value
Organization	Vision Manufacturing

   If you can find Vision Manufacturing, then you already collected it. Skip the rest of these steps and go to the next remedy in this topic.
3. If you can't find Vision Manufacturing, then you must collect it.
   - In the Setup and Maintenance work area, go to the task.
     - Offering: Supply Chain Planning

- Functional Area: Supply Chain Planning Configuration
- Task: Manage Planning Source Systems

- On the Manage Planning Source Systems page, in the Source Systems area, click the **row** that contains the value.

Attribute	Value
Name	Oracle Fusion Order Orchestration And Planning

- Click **Manage Organization List**.
- In the Manage Organization List dialog, click **Refresh Organization List**.
- In the search results, in the row that contains Vision Manufacturing, set the value, then click **Save and Close**.

Attribute	Value
Enable for Collections	Contains a check mark.

- Go to the Plan Inputs work area.

  Don't use the Plan Inputs task that's available in the Setup and Maintenance work area. Use the Plan Inputs work area instead.
- Click **Tasks > Collect Planning Data**.
- In the Collect Planning Data dialog, set the value.

Attribute	Value
Source System	OPS Assume you set up your source system to use OPS as an abbreviation for Oracle Order Orchestration and Planning.

- Move the Calendars entity and the Organizations entity to the Selected Entities window, then click **Submit**.
- Refresh the server. For details, see *Refresh the Order Promising Server for Order Management*.

4. Create a sales order and see if you can pick the warehouse on the order line.

Enable the Inventory Organization as a Manufacturing Plant

If the problem persists, try this.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Manufacturing and Supply Chain Materials Management
   - Functional Area: Facilities

- Task: Manage Inventory Organizations

2. On the Manage Inventory Organizations page, search for the value.

Attribute	Value
Name	Vision Manufacturing Assume you store your item in Vision Manufacturing's inventory.

3. In the search results, click the **row** that has your organization, then click **Manage Organization Parameters**.
4. On the Manage Inventory Organization Parameters page, verify the value.

Attribute	Value
Organization is a Manufacturing Plant	Contains a check mark. If this option isn't enabled, then enable it.

5. Create a sales order and see if you can pick the warehouse on the order line.

Set Up the Workday Pattern

If the problem persists, try this.

1. In the Setup and Maintenance work area, go to the task.
   - Offering: Manufacturing and Supply Chain Materials Management
   - Functional Area: Facilities
   - Task: Manage Facility Workday Patterns

   Learn about workday patterns. For details see, *Examples of Managing Workday Patterns*.
2. On the Manage Workday Patterns page, in the search results, click the **name** of your work day. For this example, assume you use the 7 Day 7 Shift work day.
3. In the Edit Time Workday Pattern dialog, make sure the Length in Days attribute contains 7. If it isn't 7, collections will fail.
4. If it isn't 7, try this.
   - In the Edit Time Workday Pattern dialog, set the value, then click **Save and Close**.

Attribute	Value
Name	7 Day 7 Shift OBSOLETE The Length in Days attribute is ready only, so you will obsolete the existing work day and create a new one.

- On the Work Workday Patterns page, in the search results, click **7 Day 7 Shift OBSOLETE**, then click **Actions > Duplicate Workday Pattern**.
- In the Duplicate Workday Pattern dialog, set the values, then click **Save and Close**.

Attribute	Value
Name	7 Day 7 Shift
Length in Days	7

- On the Work Workday Patterns page, click **Done**.
- On the Search page, search for Manage Facility Schedules.
- On the Manage Schedules page, in the Name column, click your **schedule**, such as 1 Year Duration Monthly.
- In the Edit Duration Schedule dialog, in the Workday Patterns area, click **Actions > Add Row**, set the value, then click **OK**.

Attribute	Value
Name	7 Day 7 Shift This action adds your new work day pattern to the schedule that you use for the inventory organization.

- In the Edit Duration Schedule dialog, in the Workday Patterns area, click the **row** that has 7 Day 7 Shift OBSOLETE, then click **Actions > Delete**.
- Click **Save and Close > Done**.
- Go to the Plan Inputs work area, then use the Collect Planning Data task to collect the Calendars entity and the Organizations entity.

5. Create a sales order and see if you can pick the warehouse on the order line.

Manage Your Plant Parameters

If the problem persists, try this.

1. In the Setup and Maintenance work area, go to the task.
    - Offering: Manufacturing and Supply Chain Materials Management
    - Functional Area: Manufacturing Master Data.
    - Task: Manage Plant Parameters
2. On the Manage Plant Parameters page, select your calendar in the Manufacturing Calendar attribute, then make sure each required attribute contains a value, and verify that you set the values correctly for your implementation. For details, see *Critical Choices for Setting Up Plant Parameters*.
3. Go to the Plan Inputs work area, then use the Collect Planning Data task to collect the Calendars entity and the Organizations entity.

4. Create a sales order and see if you can pick the warehouse on the order line.

Related Topics

- Troubleshoot Global Order Promising for Order Management
- Refresh the Order Promising Server for Order Management
- Critical Choices for Setting Up Plant Parameters

Delete Records That Contain Data About Actions

Delete records that contain data about actions that Order Management users have done while managing sales orders. Deleting these records can improve performance.

Order Management creates a request and saves it in the DOO_ACTION_REQUESTS table each time one of your users does an action on a fulfillment line in the Order Management work area, such as override a scheduled ship date, unreserve, and so on. Requests accumulate over time, they can potentially result in tens of thousands of rows in the table, and that can have a negative impact on performance.

You can use the Purge Recent User Requests scheduled process to delete data for these actions.

- Apply Hold
- Release Hold
- Recover Order

You can also use it to delete data for these actions in a fulfillment view.

- Edit
- Split
- Schedule
- Unschedule
- Reserve
- Unreserve

This scheduled process:

- Deletes each request that's in the Complete status. If the request is in Processing status, and if the action code of the request is Recover Task or Recover Process, then the scheduled process also deletes these requests.

  Deletes data about these actions from the DOO_ACTION_REQUEST table and the DOO_ACTION_REQUEST_ATTRIBUTES table in the Oracle database.
- Doesn't delete sales order data.
- Only deletes data about actions that Order Management has finished. It doesn't delete data about actions that Order Management is still processing.

We strongly recommend that you periodically run the scheduled process.

Try it.

1. Go to the Scheduled Processes work area.
2. On the Scheduled Process page, click **Actions > Schedule New Process**.

3. In the Schedule New Process dialog, click the **down arrow**, click **Search**, then search for the value.

Attribute	Value
Name	*Purge Recent User Requests*

4. Click the line in the search results, then click **OK > OK**.
5. In the Process Details dialog, set the values.

Parameter	Value
Number of Days of User Request Data to Keep	Enter a numeric value. For example, if the current system date is January 6, 2021, and if you set this parameter to 5, then the scheduled process deletes data for user actions that finished before January 2, 2021. In general, five days of data is typically a sufficient amount of data to keep, but you can adjust the value depending on your business needs. Specify the number of days that the scheduled process should not consider, starting with the current date. For example, if today is March 15, and if you set this parameter to 5, then the scheduled process won't delete any requests that Order Management created from March 15 to March 11. You must specify a positive number. In general, we recommend that you set this parameter to 30. This provides you one full month's worth of historic data that might be useful for troubleshooting purposes.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

6. Click **Advanced > Using a Schedule**, then set the values.

Attribute	Value
Frequency	Daily
Every	1 Day
Start Date	Select today's date
End Date	Select a date several years in the future.

We recommend that you schedule this process to run one time each day.
7. Click **Notification**, then select someone who can monitor the results, such as an order administrator.
8. Click **Submit**.

Fix Errors in All Sales Orders

Fix errors that happen with a set of sales orders, orchestration processes, or fulfillment tasks, not only errors that a search in the Order Management work area returns.

You can use Recover Order, Recover Process, or Recover Task actions to fix errors that cause problems in your sales orders. However, these actions can only fix errors that happen for sales orders that a search in the Order Management work area returns.

Instead, you can use a scheduled process to:

- Fix errors that happen during order fulfillment. For example, an order fulfillment line that can't reserve inventory because there isn't enough inventory available to fulfill the item.
- Fix system errors on order fulfillment lines.
- Evaluate conditions in your orchestration process. For details, see *Guidelines for Pausing Orchestration Processes*.
- Have a problem with an advance shipment notice. For details, see *Recover an Advance Shipment Notice*.

Try It

Fix errors in all sales orders.

1. Fix the problem that causes the error.
   If you don't fix the problem, then the same error will continue to occur whether or not you run recovery.
2. Sign into Oracle Applications. Make sure you have the privileges that you need to run the Recover Errors scheduled process. For details, see *Recover Errors*.
3. Go to the Scheduled Processes work area.
4. On the Scheduled Process page, click **Schedule New Process**.
5. In the Schedule New Process dialog, set the value, then click **OK**.

Attribute	Value
Name	Recover Errors

6. In the Process Details dialog, set the parameters to filter the sales orders that you must fix.
   For example, set Customer Name to Computer Service and Rentals, and the scheduled process will attempt to fix errors in all sales orders that contain Computer Service and Rentals in the Customer Name attribute.

Parameter	Fix Errors That Occur in Sales Orders
Orchestration Order Number	Where the Order attribute contains the value that you enter. Assume you create a sales order and its Order attribute contains 56849. If you set this parameter to 56849, then the scheduled process will attempt to fix errors in sales order 56849.

Parameter	Fix Errors That Occur in Sales Orders
Source Order Number	Where the Source Order attribute contains the value that you enter.
From Ordered Date	When the Ordered Date attribute occurs on or after the date that you enter.
To Ordered Date	When the Ordered Date attribute occurs on or before the date that you enter.
Source System	When the Source Order System attribute contains the source system that you select.
Process Name	When the Orchestration Process Number attribute references the orchestration process that you select.
Customer Name	When the Customer attribute contains the value that you select.
Inventory Item	That have the inventory item that you enter.
Task Type	That are currently on the task type that you select. For example, if you select Schedule, then the scheduled process will attempt to fix errors for sales orders that are currently on the Schedule task. o We recommend that you specify a task type or a task name. If you don't, then the scheduled process will attempt to recover lines that are in error for all tasks, and that might affect performance.
Task Name	That are currently on the task that you select. For example, if you select Schedule Goods, then the scheduled process will attempt to fix errors for sales orders that are currently on the Schedule Goods task.
Order Orchestration Function	That involve a specific type of processing or feature. For example, select Approvals, and the scheduled process will attempt to fix errors in each sales order that has an approval. Here are some of the values that you can set. o **Fulfillment Task**. Recover order lines that are stuck at various fulfillment tasks in the orchestration process. o **Fulfillment Process**. Recover order lines that are stuck because there's an error with the orchestration process. For example, you didn't deploy the orchestration process. o **Start Orchestration**. Use this value to unlock draft sales orders, recover lines that aren't started, or actions that are stuck. For details, see *Recover Sales Orders That Are Locked or Not Started*.
Scope	See *Set the Scope*.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

7. Click **Submit**.
8. In the Confirmation dialog, note the value of attribute Process ID, then click **OK > Close**.
9. Click **Actions > Refresh**.
10. Use the Process ID that you noted earlier to locate your scheduled process, then make sure the Status attribute for this process displays Succeeded.
11. Take action, depending on the status.

Status	Take Action
Error	In the Log and Output section, click the **Attachment** link, then examine the results. An Error status usually indicates that you must fix the root cause of the error, then run the scheduled process again.
Succeeded	View the log to determine how many fulfillment lines Order Management picked, how many it processed, and how many it couldn't process.

Set the Scope

Set the Scope parameter to filter errors according to where they happen.

If You Set the Scope To	Then the Scheduled Process Will
Order Fulfillment Errors	Attempt to fix errors that happen only with sales order data in your downstream applications during a fulfillment task. The scheduled process will ignore the Number of Days to Look for System Errors parameter and the Date to Start Looking, Counting Backward parameter. It will instead use all the other parameters.
System Errors	Attempt to recover order lines that are stuck because of a system problem, such as the database times out, the server times out, there's an unstable server connection, a resource isn't available because there's a stuck thread, there's isn't enough memory, and so on. The scheduled process will use the Number of Days to Look for System Errors parameter and the Date to Start Looking Counting Backward parameter. It will ignore all other parameters. If you set the Scope to System Errors, then we recommend that you run the scheduled process no more than one time each day.
Order Fulfillment and System Errors	Attempt to fix errors that happen with sales order data and with the system. The scheduled process will use only the Number of Days to Look for System Errors parameter and the Date to Start Looking Counting Backward parameter when it fixes system errors. It will use only the other parameters when it fixes fulfillment errors.

Set the Scope for System Errors

You can specify these parameters when you set the Scope parameter to System Errors or to Order Fulfillment and System Errors.

Set This Parameter	To Fix System Errors That Occur in Sales Orders
Number of Days to Look for System Errors	During the number of days that you specify. For example, set Number of Days to Look for System Errors to 5, set Date to Start Looking Counting Backward to May 15, and the process will attempt to fix system errors that happen on May 11 through May 15.
Date to Start Looking, Counting Backward	Starting on the date that you specify, and then counting backward from that date. If you don't set a date, then the scheduled process starts looking for system errors today, and then starts counting backward from today. If you set up this scheduled process to run on a schedule, then leave Date to Start Looking Counting Backward empty.

These parameters apply only to system errors. They don't apply to fulfillment errors.

Assume:

- A fulfillment error happens on May 10:

  ```
  Orchestration couldn't create a reservation because the reservation quantity is greater than the available-to-reserve quantity.
  ```

- A system error happens on May 11:

  ```
  The request failed. Orchestration process 300100098837243 for sales order 482655 didn't start because it isn't deployed or the server isn't available. Deploy the orchestration process. Make sure the server is up and running.
  ```

If you set these values, then the scheduled process will attempt to fix only the system error:

Parameter	Value
Scope	System Errors
Number of Days to Look for System Errors	5
Date to Start Looking, Counting Backward	May 15

If you set these values, then the scheduled process won't attempt to fix either error:

Parameter	Value
Scope	System Errors
Number of Days to Look for System Errors	1
Date to Start Looking, Counting Backward	May 15

If you set these values, then the scheduled process will attempt to fix the system error and the fulfillment error:

Parameter	Value
Scope	Order Fulfillment and System Errors
Number of Days to Look for System Errors	5
Date to Start Looking, Counting Backward	May 15

If you set these values, then the scheduled process will attempt to fix the fulfillment error but not the system error:

Parameter	Value
Scope	Order Fulfillment and System Errors
Number of Days to Look for System Errors	1
Date to Start Looking, Counting Backward	May 15

Filter the Orchestration Processes That You Want to Look At

You can also set these parameters when you set the Scope parameter to System Errors or to Order Fulfillment and System Errors. These parameters apply only to system errors. They don't apply to fulfillment errors.

Parameter	Description
Number of Orchestration Process Versions to Fix	Specify the number of versions to fix, starting with the latest version. Assume you use orchestration process *x* and orchestration process *y*. *x* has versions 1, 2, 3, and *y* has versions 1, 2, 3, 4, and 5. • This parameter has a default value of 1, so if you don't set it, then we'll look at errors in version 3 of *x* and version 5 of *y*. • If you set this parameter to 3, then we'll look at errors in versions 1, 2, and 3 of *x* and versions 3, 4, and 5 of *y*. • If you leave this parameter empty, then we'll look at all versions, and that might degrade performance.
Days to Look for Active Orchestration Processes	Specify how many days to look for orchestration processes that are active. For example, set Days to Look for Active Orchestration Processes to 30, and we will look only at orchestration processes that were active today or during the prior 29 days from today. We won't look at any orchestration process that has an end date that happens more than 30 days ago.

Assume:

- Today is May 15.
- You use orchestration process *x*, and it has versions 1, 2, 3.
- You also use orchestration process *y*, and it has versions 1, 2, 3, 4, and 5.
- The end date for orchestration process *x* is empty.
- The end date for orchestration process *y* is May 1.

If you set these values, then the scheduled process will attempt to fix system errors that involve versions 1, 2, and 3 of process *x*, but it won't consider any versions of process *y*:

Parameter	Value
Scope	System Errors
Number of Orchestration Process Versions to Fix	3
Days to Look for Active Orchestration Processes	10

If you set these values, then the scheduled process will attempt to fix system errors that involve versions 1, 2, and 3 of process *x*, and it will attempt to fix system errors that involve versions 3, 4, and 5 of *y*:

Parameter	Value
Scope	System Errors
Number of Orchestration Process Versions to Fix	3
Days to Look for Active Orchestration Processes	30

Use All the Filters

You can narrow your search even further.

Continuing our scenario, if you set these values, then the scheduled process will attempt to fix only the system error, and it will only look at system errors that involve versions 1, 2, and 3 of process *x*, but it won't consider any versions of process *y*:

Parameter	Value
Scope	System Errors
Number of Days to Look for System Errors	5
Date to Start Looking, Counting Backward	May 15
Number of Orchestration Process Versions to Fix	3
Days to Look for Active Orchestration Processes	10

Use the Default Values

Here are the default values when you set the Scope parameter to System Errors or to Order Fulfillment and System Errors.

Parameter	Default Value	If You Remove the Value and Leave it Empty, the Scheduled Process Will
Number of Days to Look for System Errors	30	Look at errors that happen on all days.
Date to Start Looking, Counting Backward	Empty	Look at errors starting today to the beginning of time, or at least to the Precambrian era.
Number of Orchestration Process Versions to Fix	1	Look at all versions of each orchestration process.
Days to Look for Active Orchestration Processes	180	Look at all active and inactive orchestration processes.

If you remove all the values from these parameters, then the scheduled process will look at all versions of all orchestration processes, starting today and proceeding to the beginning of time. To avoid performance problems, we strongly recommend that you don't do this.

Set a Schedule

If you click **Advanced > Schedule**, enable the Using a Schedule option, and then set the Frequency so it recurs, such as Weekly, then make sure you set the parameters so they work efficiently with the schedule.

Assume you don't like having errors in your system that are more than a week old. You prefer to keep it up to date.

- You run the scheduled process one time and set the Number of Days to Look for System Errors parameter to 180 to clean up all old errors.
- You then set up a schedule to run the process one time each week.
- There's no need to look beyond 7 days because you already fixed all the old errors, so you set Number of Days to Look for System Errors to 7. This helps to make sure the process runs efficiently and doesn't put a big burden your system resources.

You can set other parameters in a similar way to help make the process run more efficiently.

In some cases, you might have a need to recover from a specific set of errors. You can set the parameters and specify the schedule differently depending on what you need to recover:

If You Need to Recover	Then Set These Parameters
Fulfillment tasks	• Set the Task Type or the Task Name. • Set the Scope to Order Fulfillment Errors. • Set the Order Orchestration Function to Fulfillment Task.
Orchestration processes	• Process Name. • Set the Scope to Order Fulfillment Errors. • Set the Order Orchestration Function to Fulfillment Process.
System errors	Set the Scope to System Errors.

Assume you need to recover all sales order that are stuck at the scheduling task. Here are your settings:

- Set Task Type to Schedule

- Set Scope to Order Fulfillment Errors
- Set Order Orchestration Function to Fulfillment Task

Frequency

> **Note:** Don't run this scheduled process more than one time each day for each combination of parameters. For example, assume you do all the required setups for Global Order Promising, and now need to recover the order lines that are stuck at scheduling. You can run the scheduled process one time each day to recover from these errors.

You can run this scheduled process more frequently to meet a specific requirement, say every eight to ten hours. If you do, you must monitor performance closely. We recommend that you don't run it too often, such as every few minutes or once an hour. You can always run it manually to meet a specific need.

Guidelines

- Use a fulfillment view in the Order Management work area to determine the attribute values that you need to use for each sales order.
- Add values to more than one parameter to refine your search. For example, set Customer Name to Computer Service and Rentals, and set Order Orchestration Function to Approvals, and the scheduled process will attempt to fix errors in sales orders that include approvals for your Computer Service and Rentals customer.
- Use Process Options and Advanced to set other options.
- You can use the Days to Retry Recovery for Business Events parameter to specify the number of days to recover from errors that involve business events. See *Manage Order Management Parameters*.
- Set the profile options. For details, see the Percent of Order Lines in Error in Each Hour subtopic in *Use Order Profiles to Control Order Management Behavior*.
- Learn about source orders, source systems, and orchestration processes. For details, see *How Order Management Transforms Source Orders Into Sales Orders*.

Avoid Severe Performance Problems

. Here's how you can reduce the amount of data that the scheduled process must process:

- You must set at a value in at least one parameter. We recommend that you specify as many parameters as possible.
- Set the scope.
- In general, don't run this scheduled process more than one time each day.

You can use the Recover Errors scheduled process only to recover from unplanned errors. You must not use it as part of your regular business flow. For example, assume you frequently don't have enough quantity to reserve an item, and you frequently encounter an error:

```
A reservation was not created because the reservation quantity is greater than the available-to-reserve quantity.
```

You might be tempted to set up Recover Errors so it runs repeatedly on a schedule until there is enough quantity and fulfillment succeeds. However, this technique will almost certainly effect performance.

Instead of using Recover Errors, you must implement another solution. In our example, you could add a Pause step to your orchestration process that pauses the process until the quantity that's available to reserve is greater than the requested quantity on the order line. You would place the pause step immediately before the Reservation step.

How Recover Errors Works

Recover Errors recovers sales orders in a first-in, first-out sequence according to when Order Management created them. You can't modify this behavior.

Assume you:

- Create sales orders for the AS54888 item.
  - Create sales order 1 on Monday with a quantity of 100.
  - Create sales order 2 on Wednesday with a quantity of 200.
  - Create sales order 3 on Friday with a quantity of 1,000.
- The AS54888 is currently out of stock, so sales orders 1, 2, and 3 go into error.
- You place a purchase order that will refill stock to a quantity of 1,000 on Saturday.

On Saturday, you run the Recover Errors scheduled process. It will use this sequence to recover the orders:

1. Recover supply for all of order 1.
2. Recover supply for all of order 2.
3. Split order 3 into two lines. It recovers line 1 with a quantity of 700, then sets the quantity for line 3 to 300 and the status for line 3 to Pending.

The same logic applies for other kinds of errors and data. For example, assume the Reservation task fails for these sales orders. The scheduled process will use the same sequence to attempt to recover the orders, and it will reserve supply according to the same sequence. If you need to prioritize, you might consider creating an allocation rule to prioritize who gets to have their supply reserved. For details, see *Allocation Rule*.

This example describes one way of how Recover Errors works. The exact behavior for your usage might be significantly different.

Related Topics

- Fix Errors in More Than One Sales Order
- Fix Errors in Sales Orders
- Examine Error Messages for Sales Orders
- How Order Management Transforms Order Lines Into Fulfillment Lines

Troubleshoot Global Order Promising for Order Management

Fix problems that happen with Global Order Promising in your Order Management implementation.

Assume you encounter one of these error messages.

- `You can't submit the sales order because the scheduling for item AS54888 isn't set up correctly.`
- `The request to submit failed because you can't use attribute ITEM ID with value 1000990 as a configure option for configured item AS54888.`
- `Order management can't schedule the fulfillment line because item AS54888 isn't associated with organization M1.`

- **Scheduling failed for the item because supply isn't available before the latest acceptable date happens.**
- **You can't use an attribute with a value as a configure option for a configured item.**
- **The item passed to order promising was not found in any organization. Verify the item definition.**

Or assume you encounter one of these problems.

- You create an organization but it isn't available in the Organization attribute when you use the Manage ATP Rules page or the Manage Sourcing Rules page.
- You create an organization but it isn't available in the Warehouse attribute on an order line in the Order Management work area.

To fix these problems, verify your setups.

1. Verify that you have:
    - Set up the item so it supports order promising
    - Set up the inventory organization so it supports order promising

    For details, see *Collect Source System Data*.
2. Verify that you have:
    - Created and assigned a sourcing rule
    - Created and assigned an available-to-promise rule
    - Correctly set up the MSP_DEFAULT_ASSIGNMENT_SET profile
    - Correctly set up the MSC_SRC_ASSIGNMENT_CATALOG profile

    For details, see *Set Up Promising Rules and Sourcing Rules for Order Management*.

Related Topics

- Set Up Promising Rules and Sourcing Rules for Order Management
- Overview of Collecting Promising Data for Order Management
- Collect Source System Data