Oracle Fusion Cloud SCM

Administering Pricing

26B

Oracle Fusion Cloud SCM
Administering Pricing

26B

G52000-01

Copyright © 2011, 2026, Oracle and/or its affiliates.

Author: carl casey

Contents

Get Help i

1 Hello 1
What's New in Administering Pricing 1
Overview of Oracle Pricing 8
How Profiles, Segments, and Strategies Work Together 23
How Pricing Prices Sales Orders 28
How Pricing Calculates the Order Line 44
Roadmap to Manage Oracle Pricing 46

2 Profiles, Segments, and Strategies 51
Profiles and Segments 51
Strategies 55

3 Lists 61
Rules 61
Prices 64
Costs 70
Discounts 73
Shipping Charges 95
Currencies 99
Adjustments 111

4 Charges, Elements, and Rounding 135
Charges and Elements 135
Rounding 141

5 User Interfaces 153
Totals, Order Lines, Lookups, Flexfields 153
Pricing Guidelines 177

Messages 191

6 Administer 197
Various Set Ups 197
Times, Dates, and Values 211
Pricing Matrixes 223
Covered Items 262
Configured Items 266
Migrate Through Updates 273
Performance and Troubleshooting 280

7 Import and Export 297
Import Price Lists 297
Import Discount Lists 316
Import Cost Lists 355
Export 356
Remote 358

8 Integrate 379
Get Price Details from Cost Accounting 379
Get Cost Details from Oracle Costing 381
Do Cost Plus Pricing with Oracle Costing 404
Manage Price Lists That Have Rate Plans 411
Use Financial Orchestration and Oracle Pricing to Price Internal Transfers 417

9 Service Mappings and Pricing Algorithms 427
Mappings 427
Algorithms 460

10 Use Cases 589
Overview of Pricing Use Cases 589
Strategies 589
Lists 612
Extensibles 682
Adjustments 709

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

Some application pages have help icons to give you access to contextual help. 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. If the page has contextual help, help icons will appear.

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.

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

What's New in Administering Pricing

Get a summary about content that's new or significantly revised in each update of Administering Pricing.

To get details about:

- New features in each update, go to *Cloud Applications Readiness*, then click Order Management What's New.
- Known issues for each update, see *Oracle Supply Chain Management Cloud Functional Known Issues and Maintenance Packs (Doc ID 1563075.1)*.

Update 26B

Topic	Description
Use Spreadsheets to Manage Pricing	Revised. If you need to use the Item Class attribute in your spreadsheet, then you must enable a profile option.

Update 25A

Topic	Description
Promote Pricing Algorithms Into the Latest Update	Revised. See the revised caution about promoting.
Manage Pricing's Effectivity Dates	Revised. See the new Price According to Pricing Date subtopic.

Update 24D

Topic	Description
Manage Pricing Promotions	Revised. See the new Apply Strategies and Payment Terms subtopic.
Manage Tax in Oracle Pricing	Revised. See the new Use Amounts in the Transaction Currency to Calculate Tax subtopic.
Manage Price Lists That Have Rate Plans	Revised. See the new Tier You Adjustments on Rate Plan Charges subtopic.
Manage Pricing Algorithms	Revised. See the new Reset Pricing Algorithms to the Predefined Version subtopic.

Update 24C

Topic	Description
Manage Pricing Promotions	New. Set up a simple or tiered discount for one or more items or categories, or for all items in a single promotion.

Topic	Description
Create Discounts That Accumulate or Cascade	Revised. Has some updates for the Retrieve Basis Values step, and the steps that add and delete the running net price.
Troubleshoot Importing Price Lists	Revised. Has some new updates about how to format date data and the operating systems that you can use.

Update 24B

Topic	Description
Adjust Price According to Item's Category	New. Adjust price on a price list or a discount list according to one or more categories in a catalog.
Get Cost Details from Oracle Costing	Revised. If you use Oracle Global Order Promising to promise your sales orders, then you must set up your orchestration process so Promising returns a default value for the warehouse before the process calls Oracle Pricing.
Get Price Details from Cost Accounting	Revised. You can use this feature to integrate only with Oracle Cost Accounting. You can't use it to integrate with any other application.
Manage Pricing Parameters	Revised. You can use the Product Catalogs parameter to select the catalog that you want Pricing to use.

Update 24A

Topic	Description
Guidelines for Importing Cost Lists	Revised. See the new Update a Large Number of Cost Lists subtopic.
Manage Price Lists That Have Rate Plans	Revised. Entirely revised topic. Includes new details about pricing your rate plan according to attribute values and using the Pricing Administration work area to do it.

Update 23D

Topic	Description
Guidelines for Importing Cost Lists	New. Use file-based data import to import a large number of cost lists.
Pricing Matrices	Revised. You can add up to 25 conditions and results in a pricing matrix when you create a pricing rule.
Promote Pricing Algorithms Into the Latest Update	Revised. You must promote all of your pricing algorithms as soon as you perform a quarterly update. We recommend that you promote your algorithms in every update.
Manage Modifications Through Updates	Revised. See the new procedural details.
Manage Price Lists That Have Rate Plans	Revised. See the new Manage Rate Plans for Subscriptions subtopic.
Create Discounts That Accumulate or Cascade	Revised. We updated the sequence of code in one of the examples. Search for *Cascade your discount modifications according to the adjustment basis*.
Troubleshoot Your Pricing Setups	Revised. EnableCache comes predefined as False. Don't set it to True.

Update 23C

Topic	Description
Apply Price According to Minimum and Maximum Quantity	New. Specify the minimum and maximum quantities that you want to use when calculating the price on an order line.
Export Price Lists	New. Export your price lists so you can migrate, convert, and maintain them.
Create Price Lists for Each Customer *Create Discounts That Accumulate or Cascade*	Revised. Contains some updated technical changes.
Use Rate Plans with Your Subscriptions *Use REST API to Manage Pricing Details*	Revised. Has new details about rate plans.
Manage Pricing Charge Definitions	Revised. Has new details about how to set the code and name.
Manage Price Lists	Revised. Don't use the Ceiling Price List or the Floor Price List types.
Override Base List Price	Revised. Use this procedure only with a standard item. You can't use it with a coverage item or subscription item.

Update 23B

Topic	Description
Manage Price Lists That Have Rate Plans	New. Set up a rate plan so you can create and manage charges for your subscriptions.
Currency Conversion Lists	Revised. Use the Pricing Strategies REST API to manage an override currency for a pricing strategy.
Manage Pricing Parameters	Revised. Get details about the new Algorithm Exception Details parameter.
Use REST API to Manage Pricing Details	Revised. See the new Import Rate Plans subtopic and the new Create and Approve a Price List subtopic.
Import Batches of Price Lists	Revised. See the revised End a Charge and Create a New One subtopic.
Import Batches of Price Lists *Import Batches of Discount Lists*	Revised. You can use FBDI to import pricing details on the Microsoft Windows or on the Apple macOS operating systems.
Import Price Lists	Revised. We recommend that you use PriceListsImportBatchTemplate.xlsm instead of PriceListImportTemplate.xlsm.

Update 23A

Topic	Description
Manage Pricing's Effectivity Dates	New. Make sure your effectivity dates work correctly when you price an item in Order Management or when you use the PriceSalesTransaction REST API.
Pricing Rules	Revised. If your rule references another object, then you can't use Edit Rules Table Columns to delete a part of the rule. You must first remove the reference, and then do the delete.
Manage Pricing Parameters	Revised. You must specify All Business Units in the Business Unit attribute for at least one organization.
Import Batches of Price Lists	Revised. Has new details about how to use the End Insert operation.

Topic	Description
Integrate Pricing Algorithms Without Using Integration Cloud Service	Removed. You must use Integration Cloud Service to integrate a Pricing Algorithm. For more, see *Get Price Details from Cost Accounting*.

Update 22D

Topic	Description
Use REST API to Manage Pricing Details	New. Get pricing details for configured items. Troubleshoot your REST API calls.
Assign Pricing Strategy According to Business Unit	Revised. We changed MatrixDomanAMLocal to MatrixDomainAMLocal.
Get Cost Details from Oracle Costing	Revised. If you encounter setup problems, you can now click a link to access a document that will help you troubleshoot your setup.

Update 22C

Topic	Description
Select Choices in Pricing Matrixes	Revised. See the Value Set subtopic for a new example.

Update 22B

Topic	Description
Get Cost Details from Oracle Costing	Revised. In your code, you must use **`type="ItemCostInput" minOccurs="0" maxOccurs="unbounded"`** instead of only **`type="ItemCostInput"`**.
Manage Pricing Matrices	Revised. Get the latest details about the Date Effectivity Enabled option.
Guidelines for Using Pricing Spreadsheets	Revised. Get the latest details to avoid problems.
Select Choices in Pricing Matrixes	Revised. Get values for an attribute on a pricing matrix according to a value set. Use the Domain Type attribute in a pricing matrix or matrix class to specify the value set.
Set Up Roles and Privileges for Pricing Administrators	Revised. Learn how to add an administrator to a business unit.
Troubleshoot Importing Price Lists	Revised. Includes new tips to troubleshoot your imports.
Guidelines for Importing Batches of Discount Lists	Revised. Assume you have an existing price list and charges for that list. If you want to update only the charges, use No-Op for the price list, and use Update for the charges.
Import Batches of Price Lists	Revised. Set up a schedule when you delete pricing data so the scheduled process runs at regular intervals.
Guidelines for Using Pricing Spreadsheets	Revised. Includes some helpful new guidelines.
Overview of Using Spreadsheets to Manage Pricing	Revised. Manage a low to moderate amount of data. If you have a large amount of data or you must do a massive update, don't use Application Development Framework Desktop Integration (ADFDI). Use File-Based Data Import instead.

Update 22A

Topic	Description
Get Details for Pricing from External Systems	Revised. You must use Oracle Pricing to calculate pricing. You can't call your own application to do the calculation.
Import Batches of Price Lists	Revised. • Import or update a large number of pricing tiers. • Improve performance. Delete pricing data from interface tables after you're done importing.
Currency Conversion Lists	Revised. Get details about the Currency Conversion matrix class.
Troubleshoot Pricing Setup	Revised. Make sure your attribute values match each other exactly, including upper case and lower case.

Update 21B

Topic	Description
Use Financial Orchestration and Pricing Administration to Price an Internal Transfer	New. Learn how to use Financial Orchestration and Oracle Pricing to price an item that you transfer between organizations that are part of the same company.
Include Return Amounts in Pricing Totals	New. See how you can include amounts from a return line in the pricing totals for each sales order.
How Rounding and Precision Affects Price	New. See how rounding and precision affects prices on order lines.
Troubleshoot Pricing Setup	Revised. Avoid having to troubleshoot **`An Application Error Occurred`**. Make sure the End Date on your pricing objects hasn't expired.
Select a Technology to Manage Your Pricing Data	Revised. Get details about the different technologies you can use to manage pricing.
Create Discounts That Accumulate or Cascade	Revised. Make sure the algorithm applies your rules in the sequence that you expect. See the Modify the Pricing Algorithm That Applies Pricing Terms subtopic.
Assign Pricing Strategy According to Customer	Revised. Get new details about how to use the customer number or customer name.

Update 21A

Topic	Description
Assign Pricing Strategy According to Customer	New. Examine this use case to see you can assign a pricing strategy according to the value that you set for the Customer attribute on the sales order.
Refund a Shipping Charge	New. Set up pricing so you can refund a shipping charge.

Topic	Description
Apply Discount According to Time	New. Check it out. Its pretty cool.
Get Costs for Pricing from External Systems	Revised. Use a predefined integration instead having to create one. Revised topic also includes some revised code and a revised attribute name.
Pricing for Covered Items	Revised. You can apply a discount on a subscription. You can't apply a discount on a coverage.
Manage Pricing Segments	Revised. We recommend that you add at least one row that Pricing can use as the default pricing segment.
Troubleshoot Pricing Setup	Revised. Includes a new section for importing pricing.
Select a Technology to Manage Your Pricing Data	Revised. If you create a sales order, submit it, and modify the price list that you use to price the sales order, then you can't use REST API to update pricing on the sales order with the modified price list.
Use the Import Template	Revised. See an example that reflects the structure of the Oracle Database.

Update 20D

Topic	Description
Get Costs for Pricing from External Systems	Revised. Updated URLs and namespace details.
Integrate Pricing Algorithms Without Using Integration Cloud Service	Revised. Includes some new details about what you need to do.

Update 20C

Topic	Description
Assign Pricing Strategy According to Business Unit	New. Create a strategy assignment that assigns your pricing strategy according to the pricing segment that you assign to your customer and the business unit on the order header.
Set Up User Roles and Privileges for Pricing Administration	New. Set up user roles and privileges to manage the authentication and authorization that Pricing Administration uses to secure processing for pricing, including web service usage and allowing users to approve price lists.
Select a Technology to Manage Your Pricing Data	New. Evaluate the different technologies you can use to manage and maintain your pricing data.
Create Your Own Condition for a Pricing Guideline	New. Modify a matrix class so you can add your own condition to a pricing guideline.

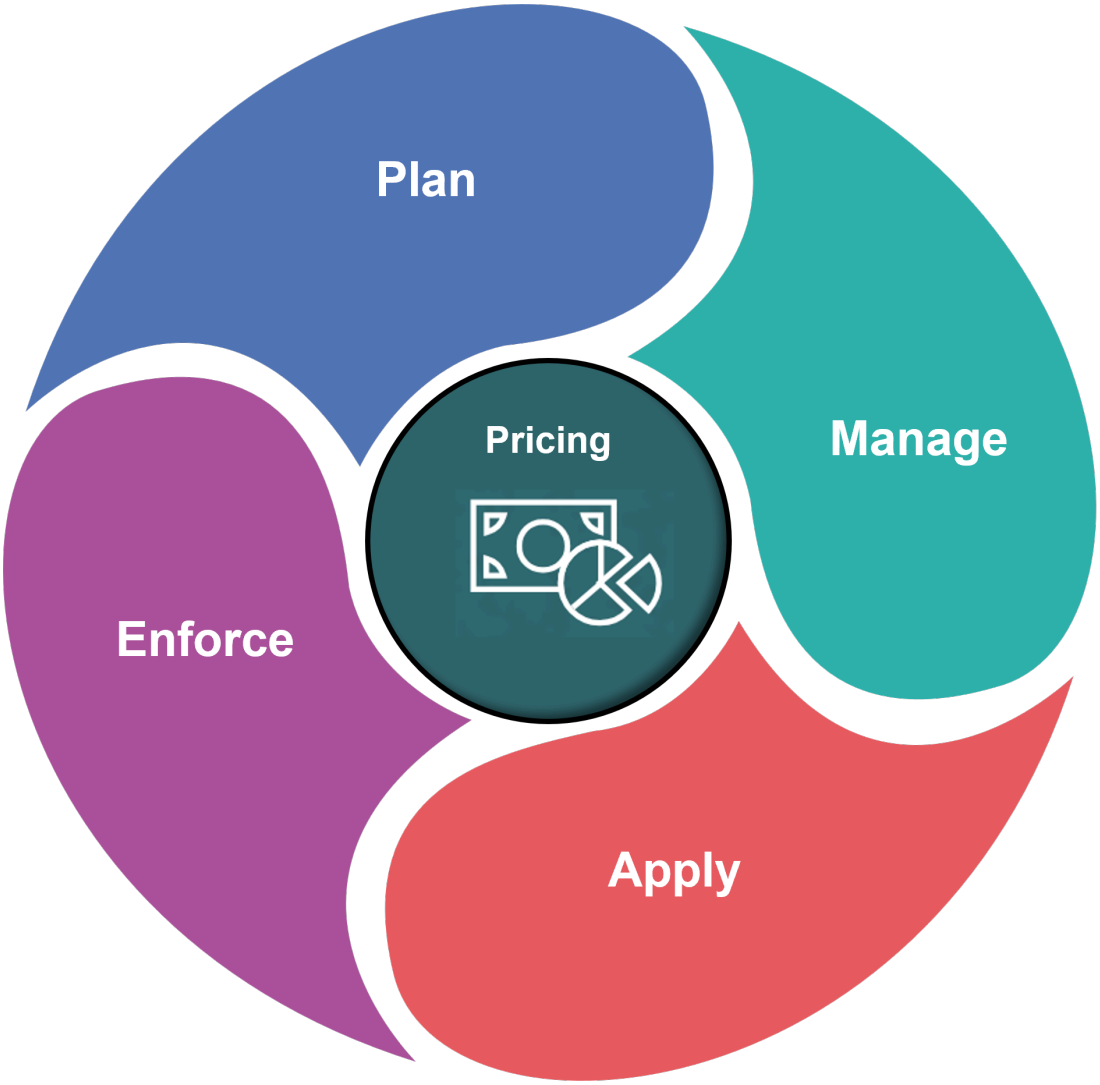
Topic	Description
Add Your Own Lookup to Charge Types	New. Use the Manage Pricing Lookups page to add your lookup to a charge type, then select that lookup when you set up a charge definition on the Manage Pricing Charge Definitions page.
Sort Records in Pricing Strategies	Revised. Use the Up One and Down One arrows on the Edit Pricing Strategy page to change the sequence of records that display according to the Precedence attribute.
Allow Users to Return Items Without Original Sales Order	Revised. You can modify the predefined pricing behavior.
Manage Rounding Rules	Revised. Refresh the Order Promising server each time you add, modify, or remove a rounding rule.
Types of Rounding Rules	If you set the Type attribute to Round to Range on the Manage Rounding Rules page, then you must create at lease one range.
Manage Pricing Guidelines	Revised. If the list for the Component attribute is empty or it doesn't have the component you need, you can add it.
Add Your Own Lookup to a Charge Type	New. Use the Manage Pricing Lookups page to add your lookup to a charge type, then select that lookup when you set up a charge definition on the Manage Pricing Charge Definitions page.
Troubleshoot Pricing Setup	Revised. Troubleshoot an unreferenced return line that doesn't have a price.
Import Price Lists	Revised. If you import a batch of lists, then make sure you set the Import Process parameter to Import Price Lists Batch when you run the Load Interface File for Import scheduled process. See *Load Interface File for Import Process*.

Update 20B

Topic	Description
Set Price Periodicity to a Granular Level	New. Set Price Periodicity to a sufficiently granular value so it meets your customer's requirements.
External Systems	Revised section. Includes new and revised topic about how to get price details from systems that reside outside of Oracle Pricing.
Pricing Rules	Revised. Search for older pricing rules that have expired or are about to expire.
Import Price Lists	Revised. Use an Excel spreadsheet to import a batch of price lists.
Manage Pricing Strategy	Revised. Add a check mark to the Allow Price List Override option only if you modify a pricing algorithm so it supports a price override.

Overview of Oracle Pricing

Use Oracle Pricing to plan, manage, apply, and enforce pricing so its consistent and profitable throughout your order-to-cash process.



Note

- **Plan** pricing strategies so they align with your business objectives. Use characteristics of your customer, the item, and the buying context to identify market segments.
- **Manage** the pricing rules that control how Pricing applies a price adjustment. For example, administer a discount on the list price of an item.
- **Apply** common pricing logic across all sales channels. Achieve consistent and accurate pricing throughout the order-to-cash process.

- **Enforce** spending limits to prevent an excessive discount. Make sure pricing is accurate so it meets your corporate revenue and margin objectives.

Plan

Plan How You Will Segment Customers

- Segment your customers so you can optimize the price you charge for an item according to the buying characteristics of your customers.
- Define pricing segments so you can sell the same item to different customers in different markets according to their willingness to pay.
- Specify who and under what circumstances to vary price to achieve the best returns.

Assume you developed the Heart Wrist Watch, a new item that monitors heart rhythms and can communicate the results immediately to a person of the wearer's choice, such as a coach or physician. You're marketing the device to long-distance runners.

Your market research determines you should price according to geographic region:



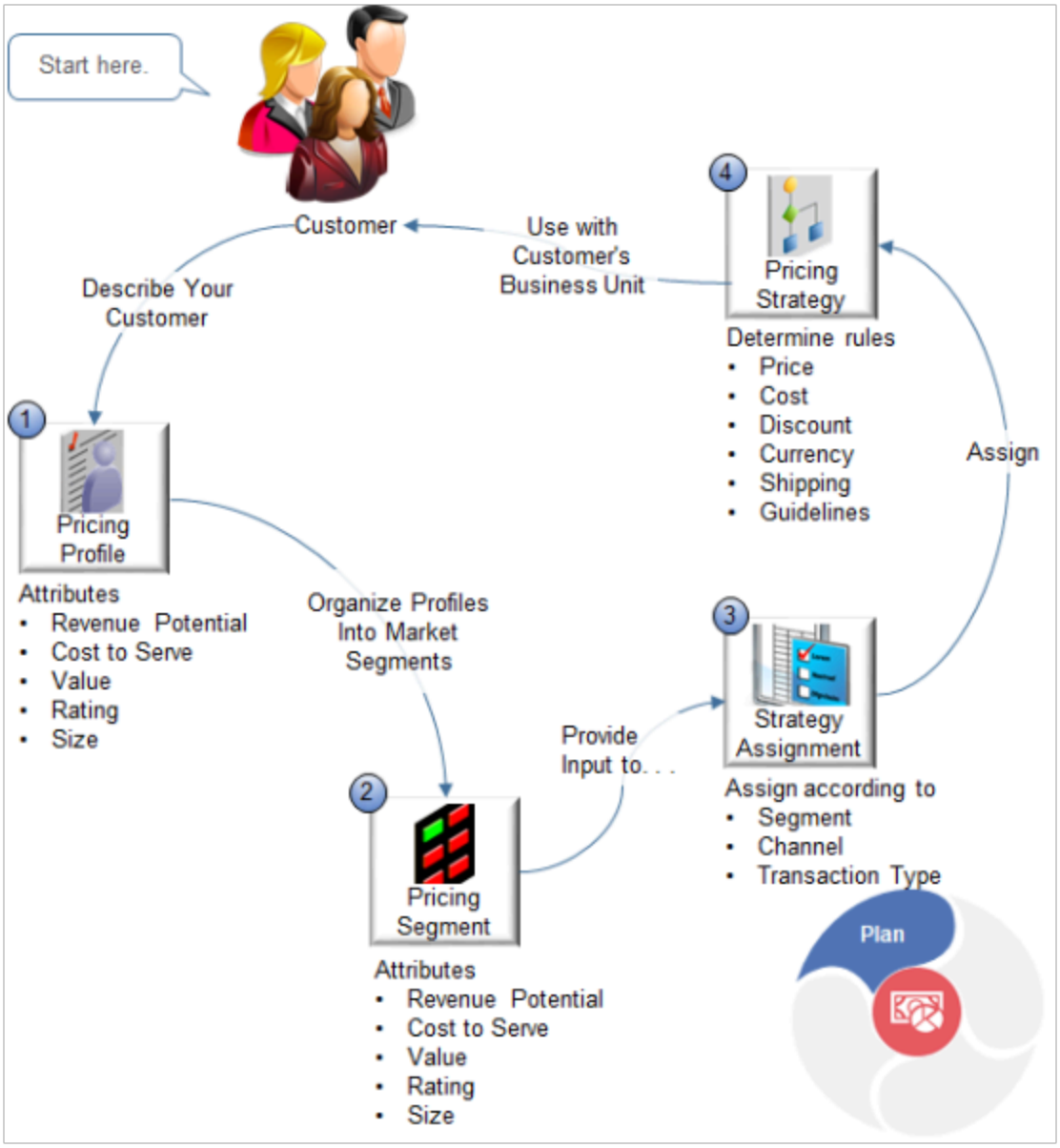
Note

- Use high margin pricing in North America and Europe where your early adopters are willing to pay a premium for the wrist watch.
- Use lower prices in emerging markets in South America, Africa, and India where you're currently entering the market and are using a loss leader strategy.

- Use discounted, competitive pricing in China, Korea, and Japan where you're already in a saturated market and are competing against established and new competitors.

Plan Your Profiles, Segments, Assignments, and Strategies

Determine price according to the characteristics of each customer, rather than using one-size-fits-all pricing:

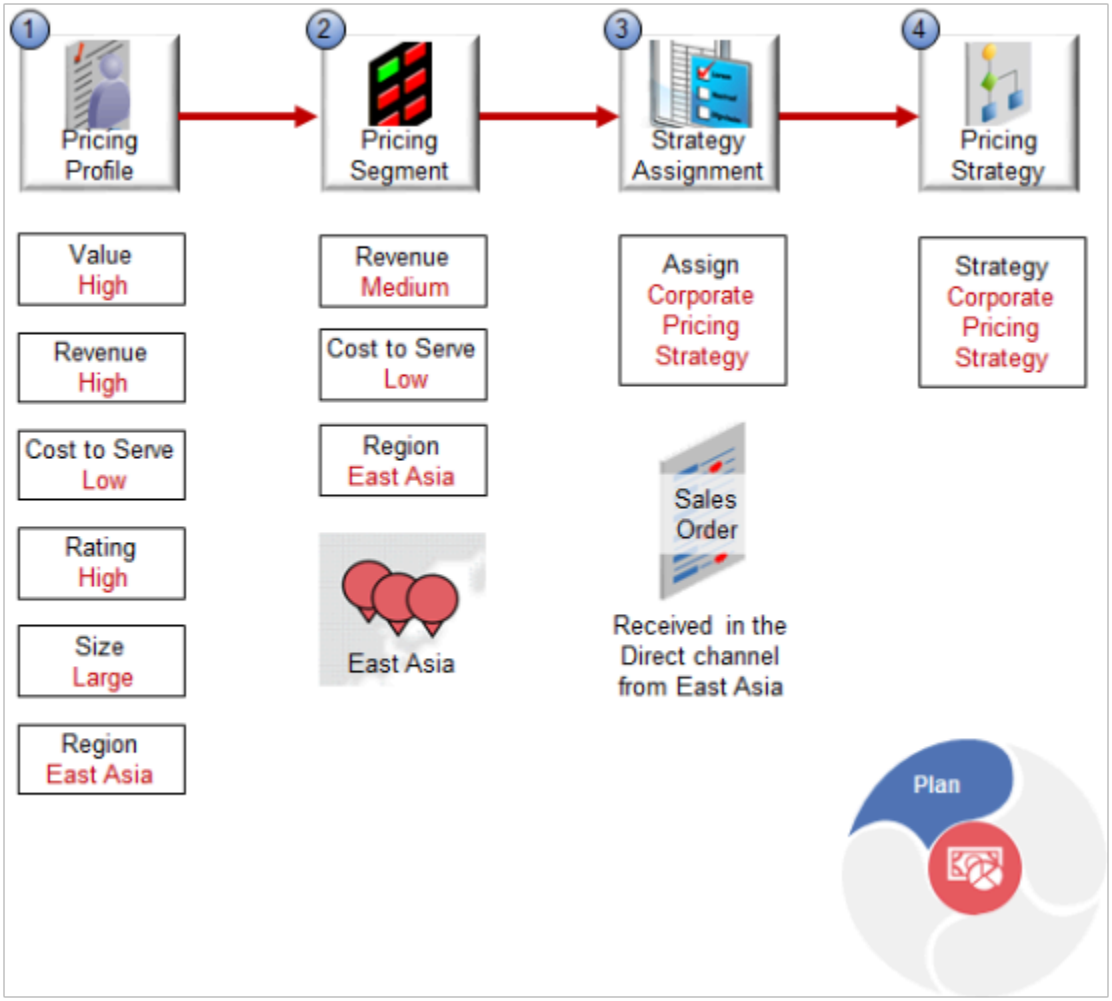


Note

Object	Description
1. Pricing Profile	Create a profile for each of your customers that describes buying characteristics according to customer attributes: • Revenue Potential

Object	Description
	• Cost to Serve • Value • Rating • Size
2. Pricing Segment	Group your customers into market segments that exhibit similar buying behavior according to market attributes: • Revenue Potential • Cost to Serve • Value • Rating • Size You can also add your own attribute, such as Region.
3. Pricing Strategy Assignment	Assign your pricing strategy according to attributes: • **Pricing Segment**. The pricing segment you assigned to the customer. • **Channel**. Where you receive the sales order, such as direct sales or inside sales. • **Transaction Type**. The type of transaction, such as sales order, service contract, or subscription.
4. Pricing Strategy	Create a pricing strategy that determines how to price your item to achieve your objectives: • Set the Business Unit attribute of the pricing strategy to your customer's business unit. Pricing will use this strategy for each sales order that references the business unit. • Create pricing rules that specify price, cost, discounts, and currency conversion. • Create shipping rules that specify shipping charges. • Create guidelines to control modifications that your users can make to price, net price, and margin.

Assume you sell the wrist watch through a direct channel to customer Softgear, located in Seoul, South Korea:



Here's your set up:

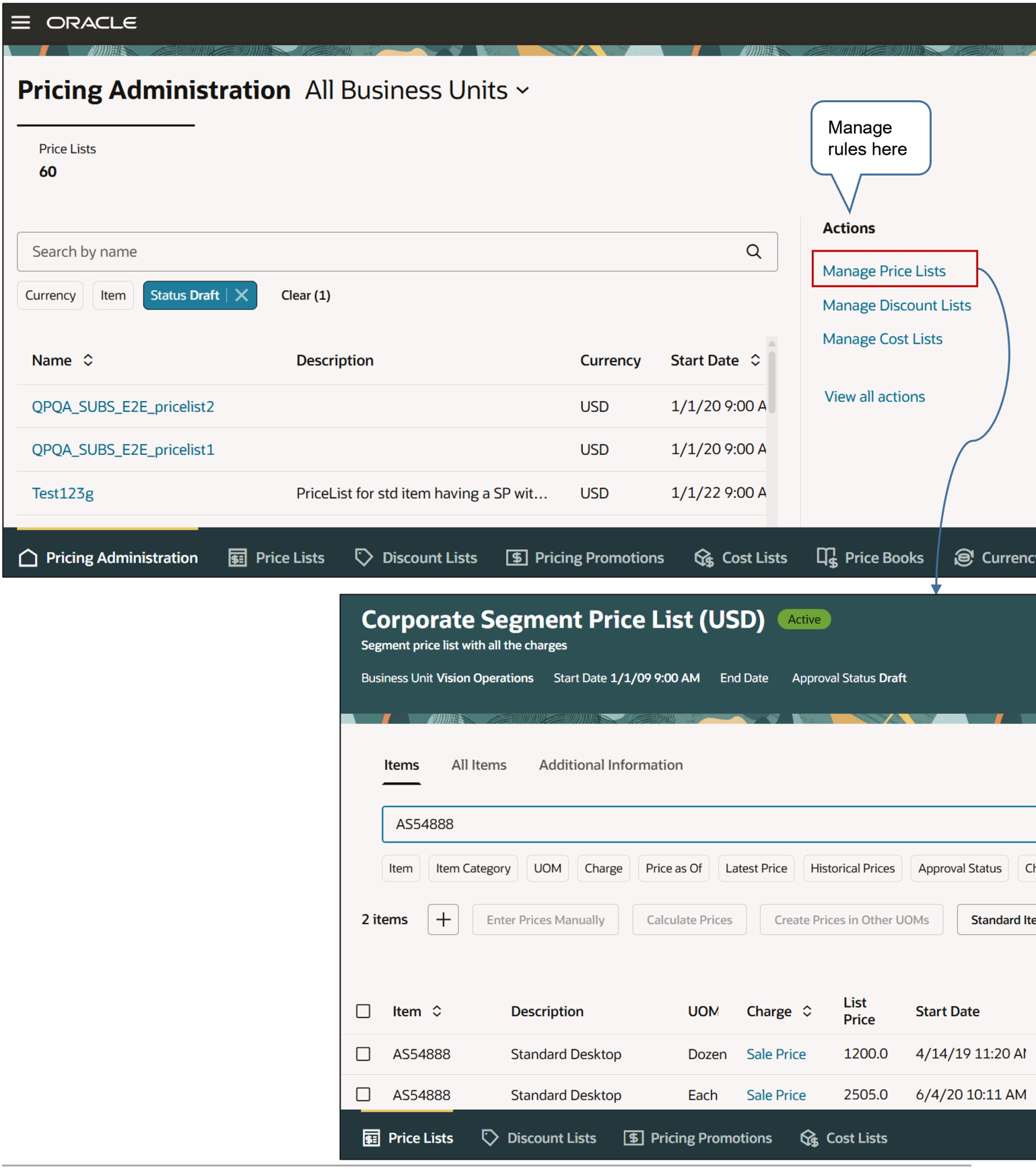
1. Create a pricing profile for Softgear:
    - You have a long-term, good working relationship with Softgear, so set Value to High.
    - Your research indicates that consumers in East Asia highly value the wrist watch and therefore have a high willingness to pay, so set Revenue Potential to High.
    - Consumers in this region are enthusiastic, early adopters of high-tech items and require little support, so set Cost to Serve to Low.
    - Rating is the perceived value of a customer to your business regarding their history, future outlook, relationship with your company, and so on, so set Rating to High.
    - Softgear is a large international company, so set Size to Large, and they're located in South Korea, so set Region to East Asia.
2. Create a pricing segment that represents your East Asian market:
    - The East Asian market for high-tech items is highly competitive. You must price the wrist watch at lower, competitive rates, so set Revenue to Medium.
    - Consumers are enthusiastic early adopters, so set Cost to Serve to Low.
    - South Korea is in East Asia, so set Region to East Asia.

3. Create a pricing strategy assignment that assigns the Corporate Pricing Strategy to direct sales orders that you receive from the East Asia segment.
4. Use the predefined Corporate Pricing Strategy, which already specifies most of the pricing details that your East Asia segment requires. Modify it, as necessary.

At run time, Pricing will use the Corporate Pricing Strategy to price the wrist watch when you receive a direct sales order for the wrist watch from Softgear.

Manage

Use the Pricing Administration work area to manage your rules:


ORACLE
Pricing Administration All Business Units
Price Lists
60
Search by name
Currency
Item
Status Draft
Clear (1)
Name
Description
Currency
Start Date
QPQA_SUBS_E2E_pricelist2
USD
1/1/20 9:00 A
QPQA_SUBS_E2E_pricelist1
USD
1/1/20 9:00 A
Test123g
PriceList for std item having a SP wit...
USD
1/1/22 9:00 A
Manage rules here
Actions
Manage Price Lists
Manage Discount Lists
Manage Cost Lists
View all actions
Pricing Administration
Price Lists
Discount Lists
Pricing Promotions
Cost Lists
Price Books
Corporate Segment Price List (USD)
Active
Segment price list with all the charges
Business Unit Vision Operations
Start Date 1/1/09 9:00 AM
End Date
Approval Status Draft
Items
All Items
Additional Information
AS54888
Item
Item Category
UOM
Charge
Price as Of
Latest Price
Historical Prices
Approval Status
2 items
Enter Prices Manually
Calculate Prices
Create Prices in Other UOMs
Item
Description
UOM
Charge
List Price
Start Date
AS54888
Standard Desktop
Dozen
Sale Price
1200.0
4/14/19 11:20 AM
AS54888
Standard Desktop
Each
Sale Price
2505.0
6/4/20 10:11 AM
Price Lists
Discount Lists
Pricing Promotions
Cost Lists

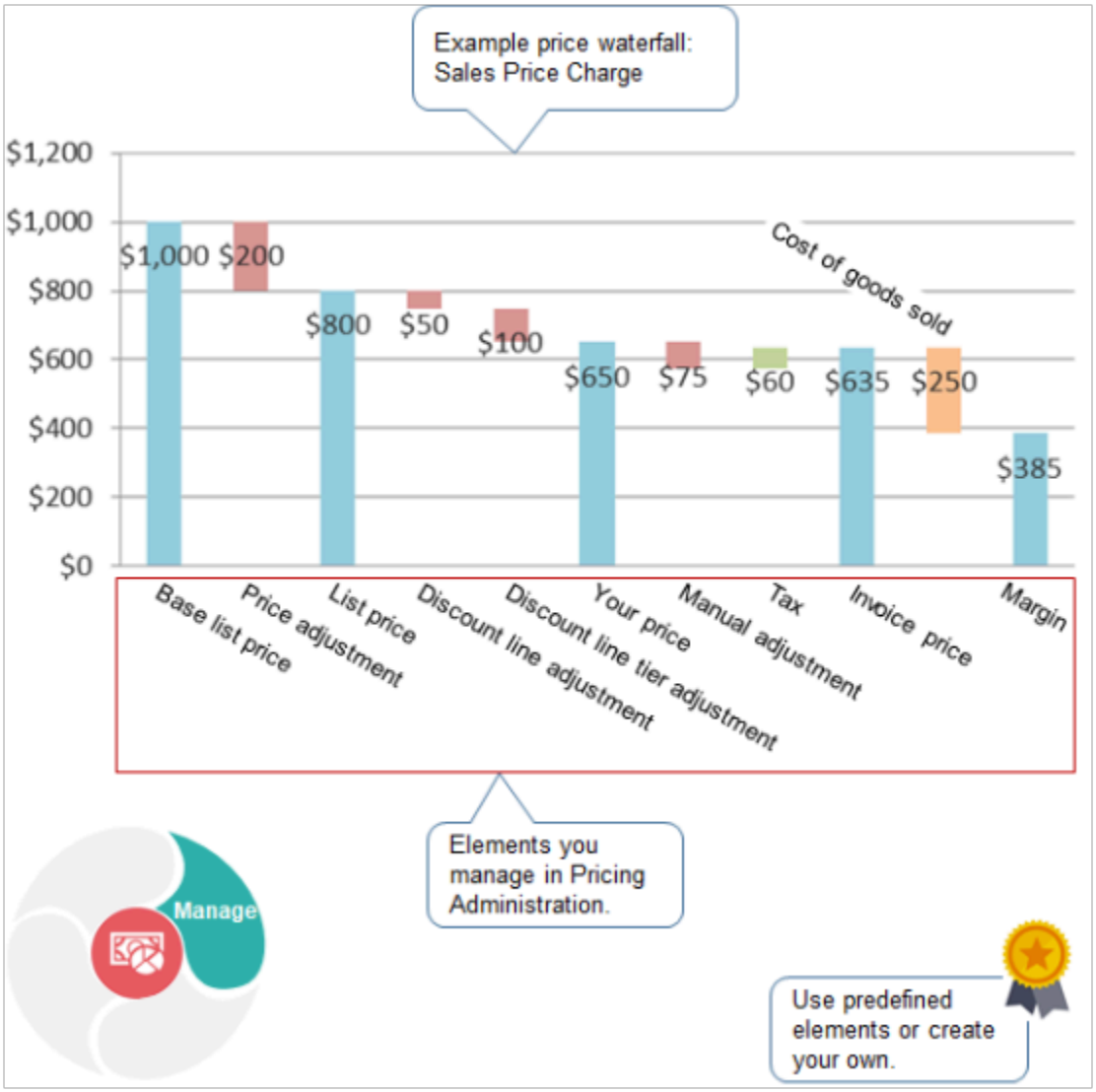
Create rules that determine the price for your items:

- Set up sales price, tiered price, discount, surcharge, freight, and adjustment according to attribute.
- Price a nonconfigured item, configured item, coverage, service, or subscription.
- Use a one-time charge or a recurring charge.
- Mark up or mark down, by dollar amount or percent.
- Apply separate charges for a single transaction, such as a $300 sales price charge for a cell phone, and another charge for the subscription fee for the cell phone.
- Search across all your rules.

Manage Price Points on Price Waterfall

A price waterfall illustrates adjustments and discounts you make on an item, typically from the base list price to your margin.

Here's the Sales Price Charge price waterfall:



Note

- This example illustrates the Sale Price charge for an item on a sales order.
- Include various types of charges that apply to your items and services. For example, installation charge, subscription, delivery fee, sale price, or restocking fee.
- A price element is an object that a pricing algorithm uses to capture different types of prices, costs, adjustments, taxes, or profit margins that it requires to create a charge breakdown or pricing analytic.

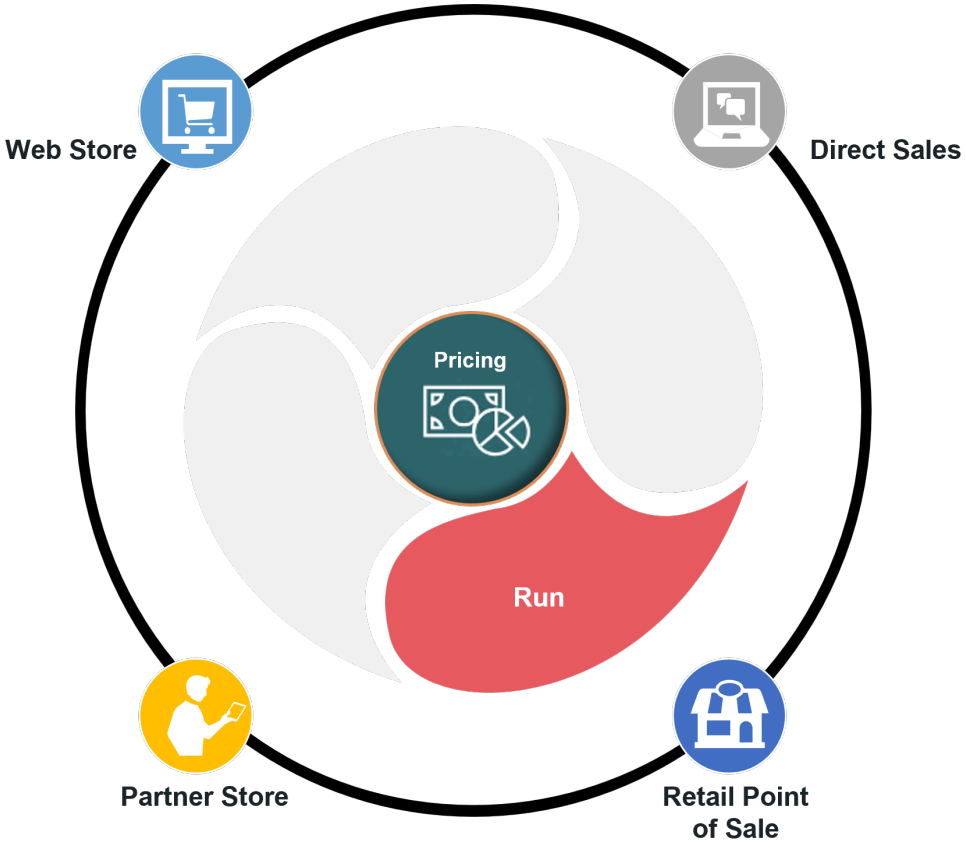
  Base List Price, Net Price Plus Tax, and Margin are each an example of a price element.
- Use the price elements that come predefined with Pricing to set up each price point in your waterfall, or create your own elements. For example:
  - Use the Base List Price element with a value of $1,000 to set the starting price point at $1,000.
  - Add the Price Adjustment element with a value of $200 to set the next price point, which is the List Price element at $800.
  - Continue adding whatever elements you need to the waterfall, such as discounts, tier adjustments, tax, margin, and so on.
- Use a price element as the pricing basis to calculate a price adjustment.
- Use and modify a predefined price definition, or create your own.

For details about tier adjustments, see *Tier Pricing*.

Run

Centralize Your Pricing

Create and enforce pricing rules and policies across your entire enterprise:

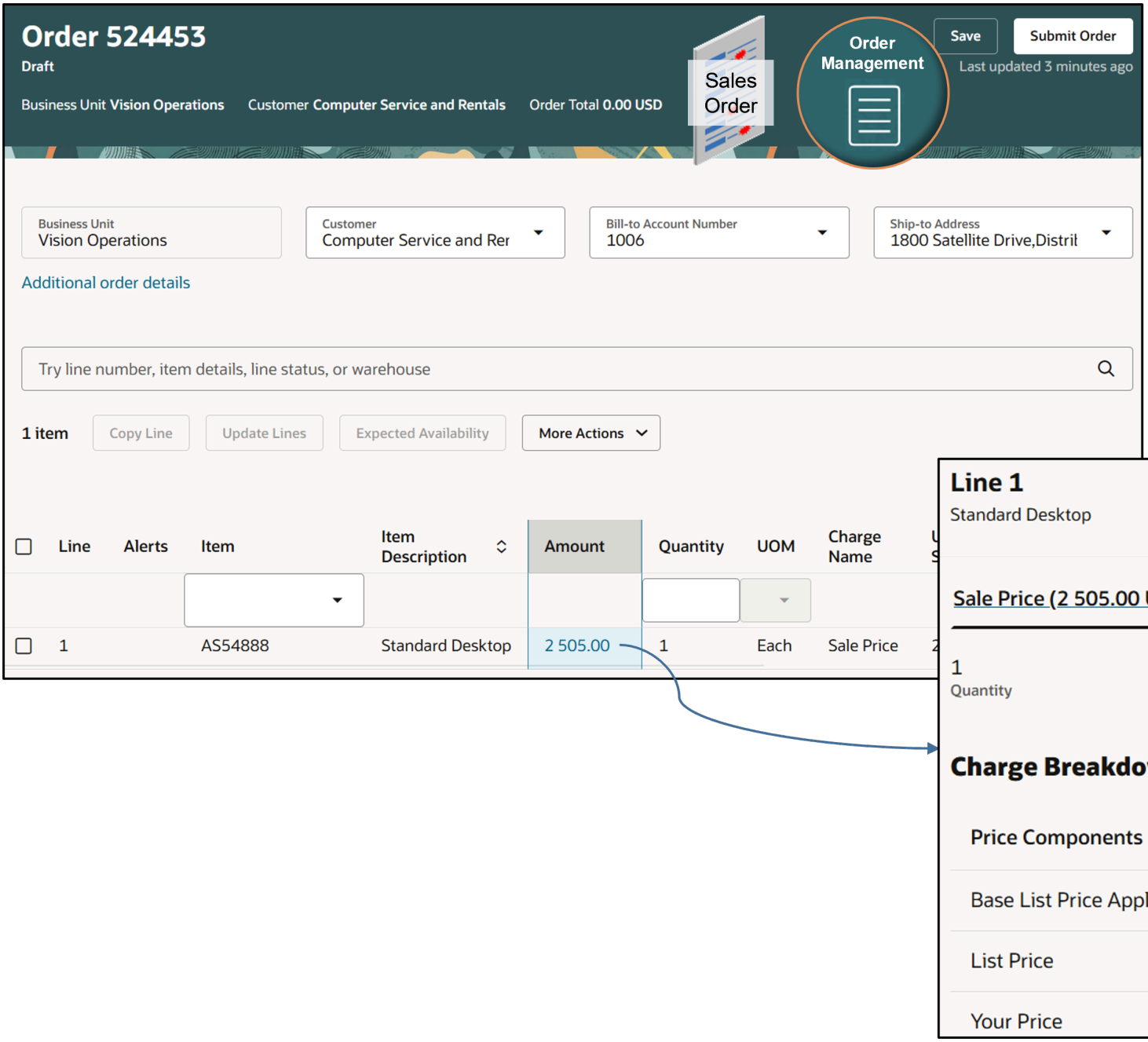


Note

- Deliver consistent and accurate prices across sales channels, such as web stores, direct sales, partner stores, retail points of sale, and more.
- Use a flexible application and services to integrate pricing across systems and applications.
- Use a variety of services, such as Price Sales Transaction, Calculate Order Totals, Validate Prices, and more.

Administer Runtime Pricing for Order Management

Administer runtime pricing for sales orders:



Note

- Calculate list price, net price, shipping, and tax according to your pricing strategy.
- Calculate one or more charges for each order line. For example, sales price plus installation fee.
- Price nonconfigured items, configured items, subscriptions, or services.
- Integrate with Oracle Financials for determine tax charges.

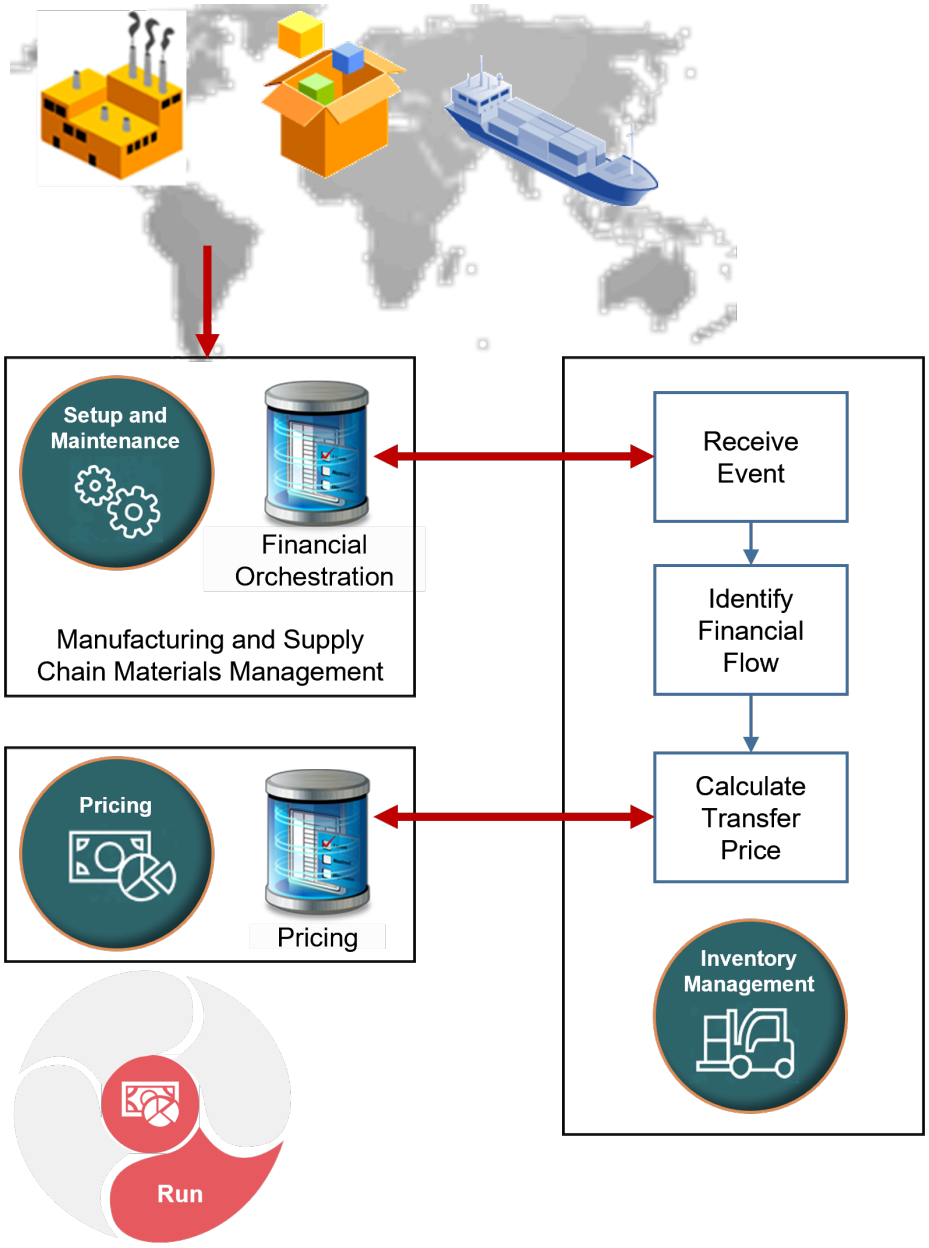
- Display the charge breakdown and order totals in a dialog according to your requirements.

  For example, for the order line, specify how to calculate charges and specify which charges to display:

  - Base List Price Applied from Price List 1
  - List Price
  - Your Price
  - Exclusive Tax (Vat20 20%)
  - Net Price Plus Tax

Administer Runtime Pricing for Oracle Inventory Management

Administer runtime pricing for transfer orders:



Note

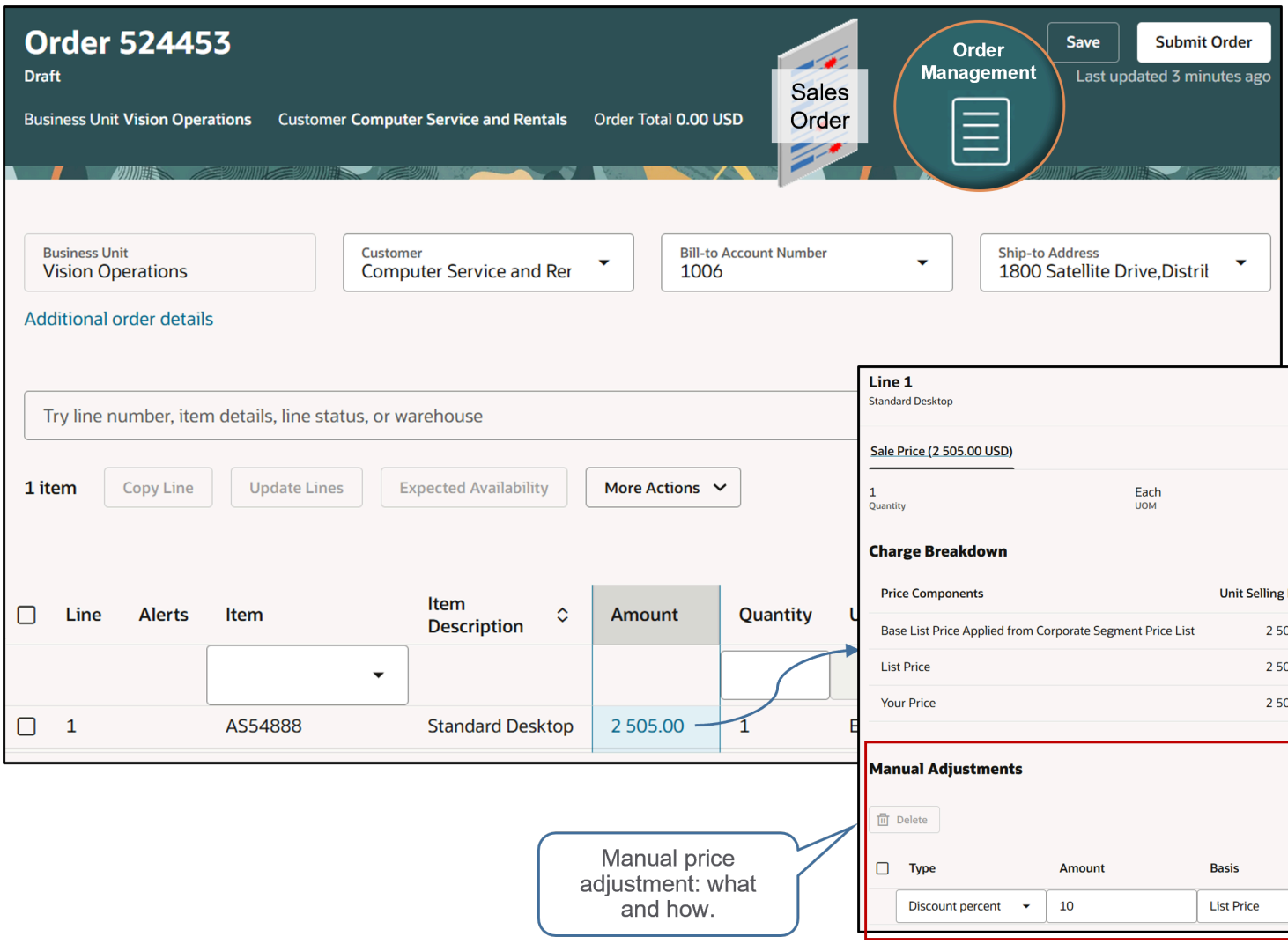
- Use your set up from the Setup and Maintenance work area of the Manufacturing and Supply Chain Materials Management offering to manage financial orchestration data for material transactions that happen in your global supply network.
- Use your set up from the Inventory Management work area to receive the event, identify the financial flow to use for the event, and calculate the transfer price.
- Calculate pricing for transfer orders while orchestrating financials, then store it as pricing data in Pricing.
- Create your pricing strategy according to a predefined matrix class specifically for financial orchestration.
- Modify the predefined pricing algorithm that Pricing uses for financial orchestration.
- Mark up or mark down price of the transfer order according to the price list, cost list, or source document price.

- Create pricing rules for nonconfigured items or configured items.

Enforce

Enforce Controls on Manual Price Adjustments

Enforce controls on the manual price adjustments that your users can make on an order line:

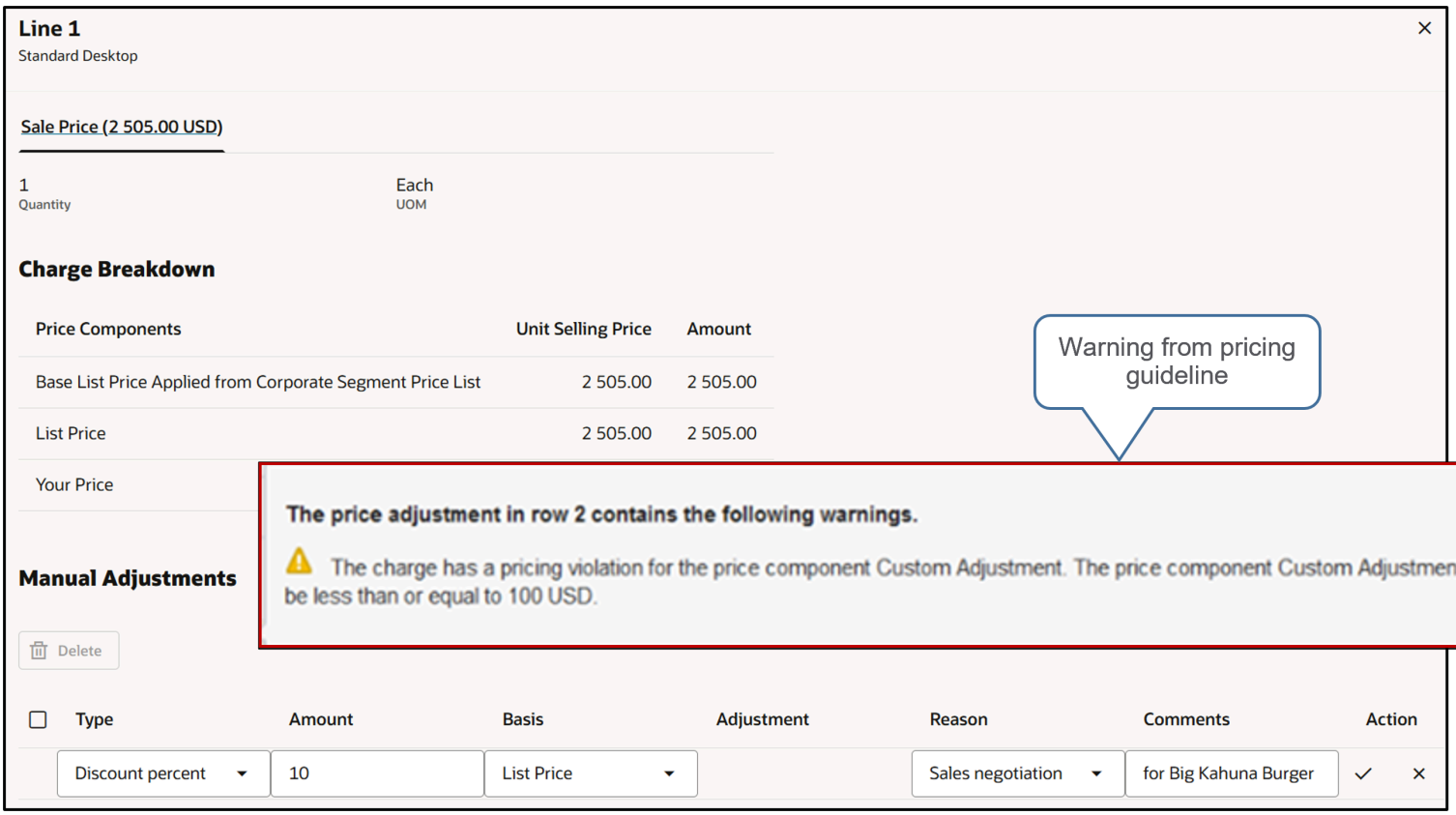


A manual price adjustment is an adjustment that the user manually sets on an order line, such as adding a discount for a sales campaign:

- Administer the price list to enforce controls on each manual price adjustment.
- Apply one or more adjustments on the net price.
- Apply adjustment as a markup or markdown on each new price against a charge.
- Apply adjustment as a percent or amount.
- Control adjustments on order lines for buy lines or return lines.
- Apply adjustments to each rolled up charge.

Use Guidelines to Enforce Pricing Policies

Use a pricing guideline to control changes your users make that affect price:



In this example, the Order Entry Specialist set the Discount Percent to 5 on the List Price. The adjustment results in a 125 USD discount, which exceeds the maximum of 100 that the guideline specifies. Here's the warning that this example displays.

```
The price adjustment in row 2 contains these warnings. The charge has a pricing violation for the price
component Custom Adjustment. The price component Custom Adjustment must be less than or equal to 100 USD.
```

A pricing guideline is a rule you create that controls changes your users can make to price, net price, margin, and so on. You apply it on an item, user role, customer detail, or time period. Pricing evaluates each guideline when it validates the sales order:

- Manage profitability, discount discipline, or compliance.
- Make sure sales orders conform to the price and discount strategies that your corporate pricing policies require.
- Prevent excessive discounts or markups.
- Prevent unexpected reductions of your profit margin. For example, write a rule that makes sure margin is 20% or more.
- Specify to display a warning or error when the modification exceeds the guideline.

Related Topics

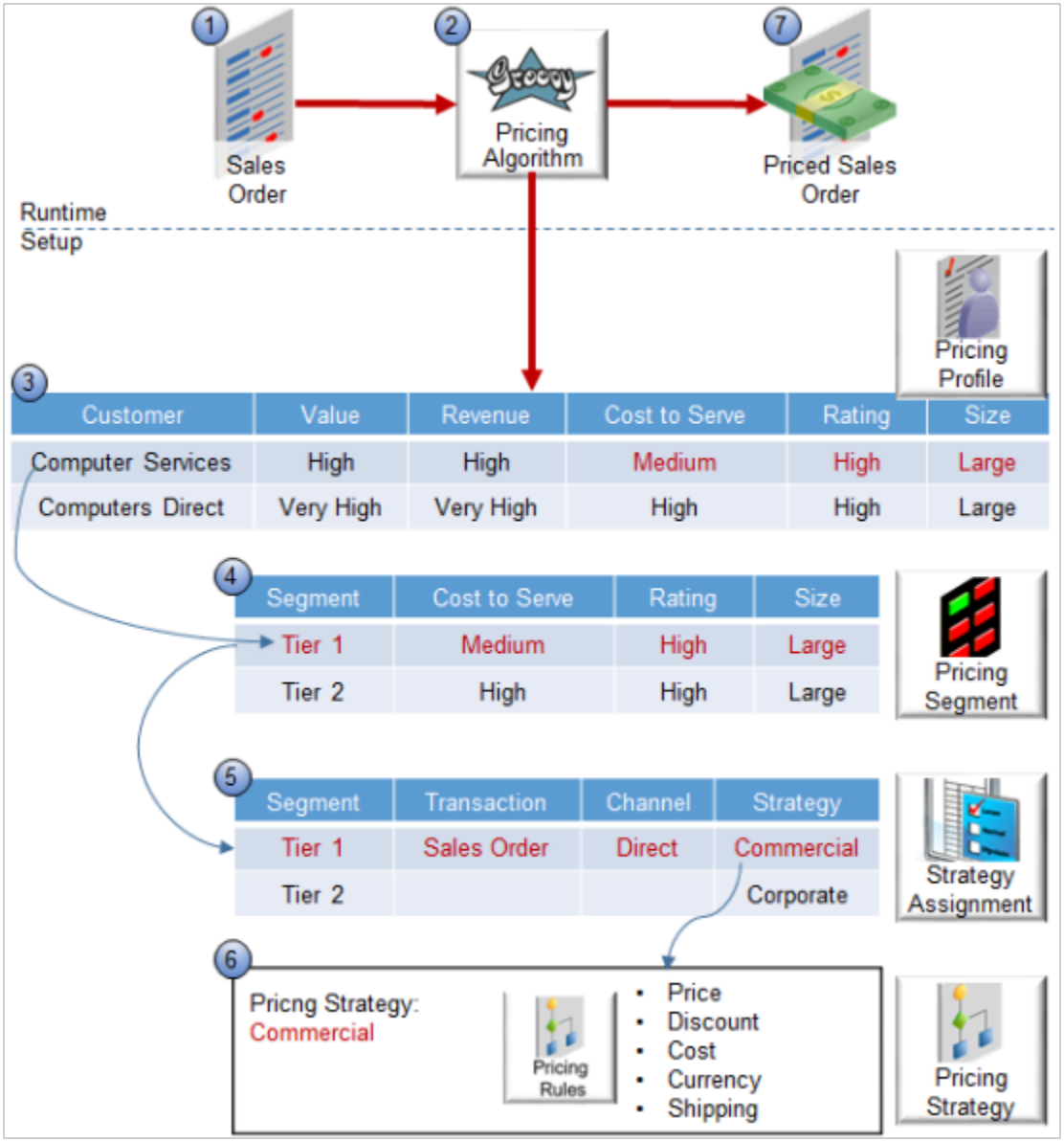
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Service Mappings
- Pricing Algorithms

How Profiles, Segments, and Strategies Work Together

You set up a customer pricing profile, pricing segment, pricing strategy assignment, and pricing strategy to price an item in a sales order.

These objects, and the price list, cost list, and discount list, are each an example of a pricing entity. A pricing entity is an object that stores details that Oracle Pricing uses to price an item.

In this example, Pricing prices a sales order it receives from Oracle Order Management for customer Computer Services, who places orders through the direct channel:



How it works:

1. Receive a sales order from an Oracle Application, such as Oracle Order Management.
2. Use a pricing algorithm that prices the sales order at runtime. The algorithm uses these objects:
    - Service mapping
    - Pricing profile
    - Pricing segment
    - Pricing strategy assignment
    - Pricing strategy

    For example, the pricing algorithm calculates the list price, applies discounts, and applies taxes. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

3. Use the pricing profiles you set up to create a relationship between the pricing attributes that describe buying behavior, and each customer.

   In this example, Pricing receives a sales order for Computer Services, who is a large, highly valued, and highly rated customer who exhibits high revenue potential and medium cost to serve. For details, see *Manage Pricing Profiles*.
4. Pricing compares each attribute of the pricing profile to each pricing segment until it finds a match. It uses the pricing segments you create to group customers who exhibit a similar set of characteristics and buying behaviors.

   For example, you can group large, highly rated customers who exhibit a medium cost to serve into a single segment, such as Tier 1. The characteristics of the segment and customer match each other:

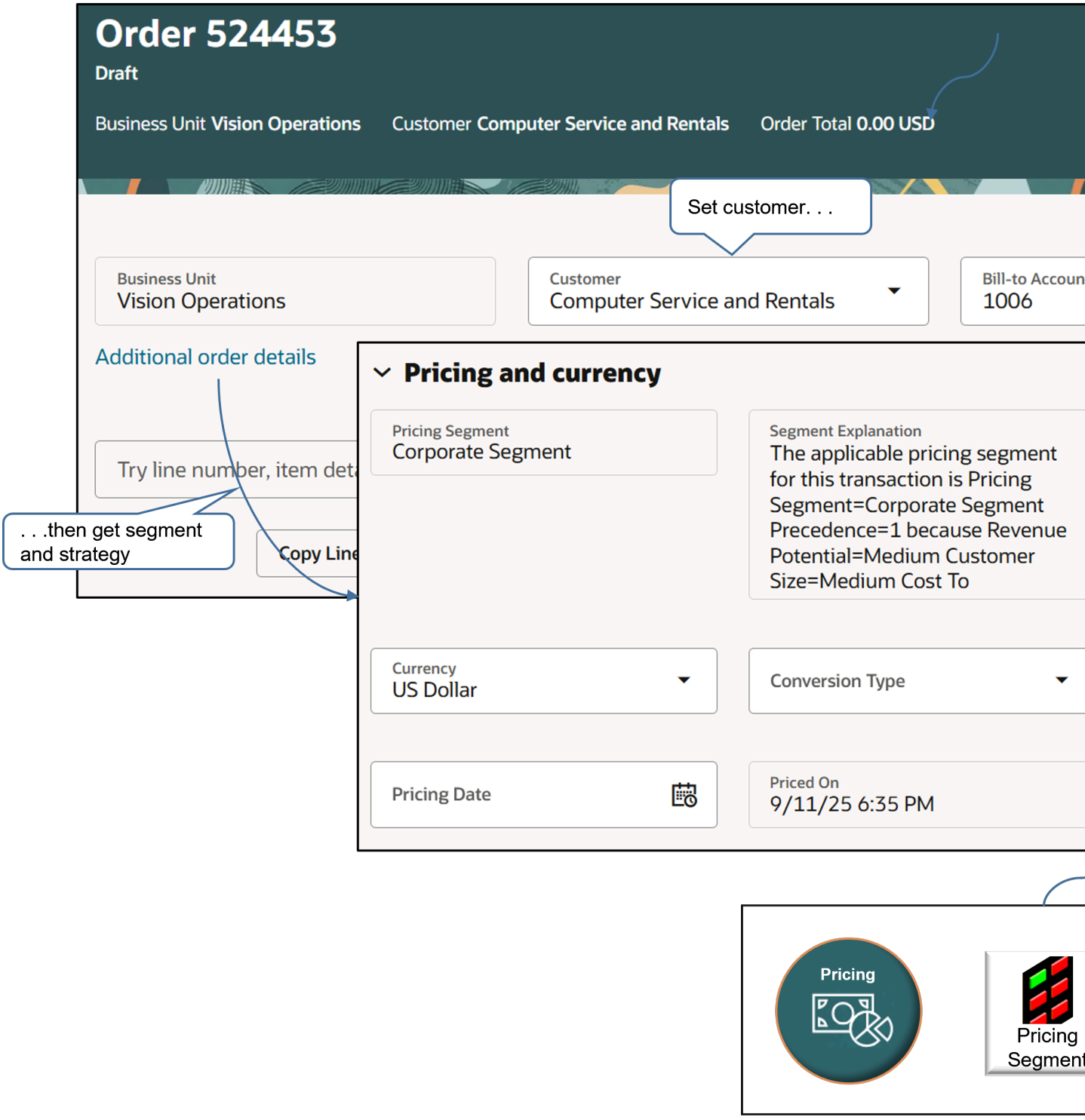
	Cost to Serve	Rating	Size
Customer: Computer Services	Medium	High	Large
Segment: Tier 1	Medium	High	Large

   So, Pricing uses the Tier 1 pricing segment for Computer Services.
5. Use the pricing strategy assignment to assign a pricing strategy to the customer. You can use different strategies for the same pricing segment to support more than one selling scenario, or you can use different strategies for different segments. For example:
   - If the Pricing Segment is Tier 1, and if the Transaction is Sales Order, and if the Channel is Direct, then use the Commercial pricing strategy.
   - If the Pricing Segment is Tier 2, then use the Corporate pricing strategy.
6. Use the pricing strategy to make sure pricing meets your pricing objective. The pricing strategy references rules you create on each list. Pricing uses the lists to calculate price:

Type of List	Description
Price	Get the list price for the item.
Discount	Get adjustments on the price.
Cost	Apply cost plus pricing, when required.
Currency Conversion	Determine currency and conversion rate to use, when required.
Shipping Charge	Get the shipping charges.

Type of List	Description

You set the customer in the sales order. Order Management sends a request to Pricing to get the segment and strategy, then displays details in the pricing strategy and segment:



7. The user adds an item to the sales order, Pricing calculates the sales order total, then sends the priced sales order, including each pricing charge and charge component, to Order Management.

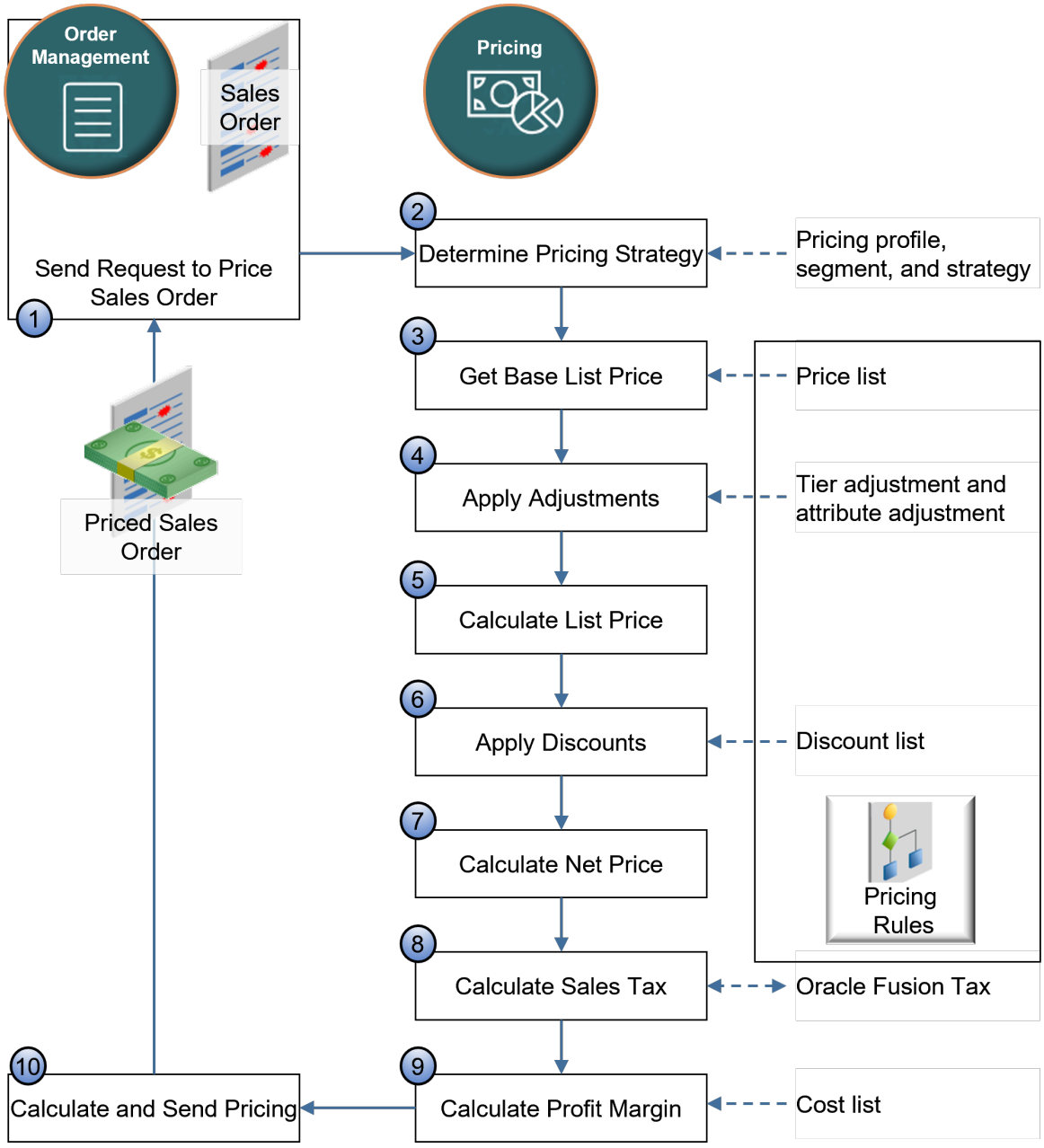
Related Topics

- Roadmap to Manage Oracle Pricing
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Pricing Profiles

How Pricing Prices Sales Orders

Oracle Pricing uses pricing entities, such as price lists and discount lists, to calculate the price of an item and the sales order total.

Assume you integrate Oracle Pricing with Oracle Order Management, and Pricing must determine charges for the AS54888 Desktop Computer item:



How it works:

1. You add an order line in Order Management. Order Management sends a request to Pricing to price the sales order, including attribute details that affect pricing:

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

Attribute	Value
Item	AS54888
Unit of Measure	Each
Line Type	Buy
Quantity	2

2. Use pricing profiles, pricing segments, and pricing strategies to identify the pricing strategy to use for this item.

   For details, see *How Profiles, Segments, and Strategies Work Together*.
3. Use a price list that the pricing strategy references to get the base list price for the item.

   A price list sets the price for each item you sell. For details, see *Manage Price Lists*.

   You set up rules that Pricing uses to calculate charges. You set them up on the price list, adjustments, discount list, cost list, and so on.
4. Reference a tier adjustment or an attribute adjustment, then apply it to the item.

   You can specify tier pricing or a pricing matrix to adjust the price that a pricing rule calculates.

   For details, see *Adjust Price for Pricing Rules*.
5. Calculate list price.
6. Reference a discount list that applies a discount on the base list price so it can determine net price.

   For example, apply a 10% percent discount on the list price.

   You can set up a discount list that applies a flat rate discount, a percent discount, or uses a pricing matrix to set up a discount according to an attribute of the item. For details, see *Manage Discount Lists*.
7. Calculate net price and return the charge and charge components.
8. Call Oracle Tax to calculate sales tax for the item.

   Pricing uses the tax details that Oracle Tax returns to create charge components for taxes. For details, see *Configure Tax Calculation and Accounting*.
9. Calculate profit margin according to costs that the cost lists define.

   You can create a cost list that references a variety of charges, such as item cost, sales commission, or labor cost. For details, see *Manage Cost Lists* and *Cost Plus Pricing*.
10. Calculate sales order totals, then communicate the results to Order Management.

Pricing sends price details to Order Management:

Pricing	Value
Base List Price	$1,000. From the Corporate price list.
Tier Adjustment	Minus $200.

Pricing	Value
List Price	$800.
10% Discount	Minus $100.
Net Price	$700.
Extended Amount	$1400.

Note

- Depending on your setup, Order Management sends a request to Pricing every time you do something on the sales order that affects price, such as:
    - Set the Customer attribute on the order header.
    - Add an item to the order line.
    - Change the quantity or unit of measure.
    - Click the value in the order line's Amount column to manually adjust the price.
    - Set the Shipping Method attribute shipment details, or set other shipping attributes that affect shipping cost, such as Ship Lines Together.
    - Save or submit.
- Pricing repeats the entire process to recalculate the sales order every time it receives a request from Order Management, including pricing for each order line and the sales order total.
- An item can include one or more charges, and Pricing returns each of these values as a charge. For example, it returns the value of the unit price as the Sale Price Charge.
- Order Management updates the charge breakdowns and other attributes that display price, such as Amount on the order line and Total on the order header.
- Pricing comes predefined to perform most of this process without requiring you to modify pricing logic.
- This topic describes a simplification of the predefined Price Sales Transaction pricing algorithm, which is the primary algorithm that Pricing uses to price an item. You can modify this algorithm. Pricing uses different algorithms to calculate different prices. For example, it uses the Calculate Sales Order Totals pricing algorithm to calculate sales order total.
- This example doesn't include details about taxes or profit margins that the pricing algorithm calculates.
- Most Oracle Applications use User Preferred Time Zone when they perform a calculation that includes a date value. However, Pricing uses the server time zone. This might affect the value for some price calculations. For example, calculations that include Start Date and End Date.

For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

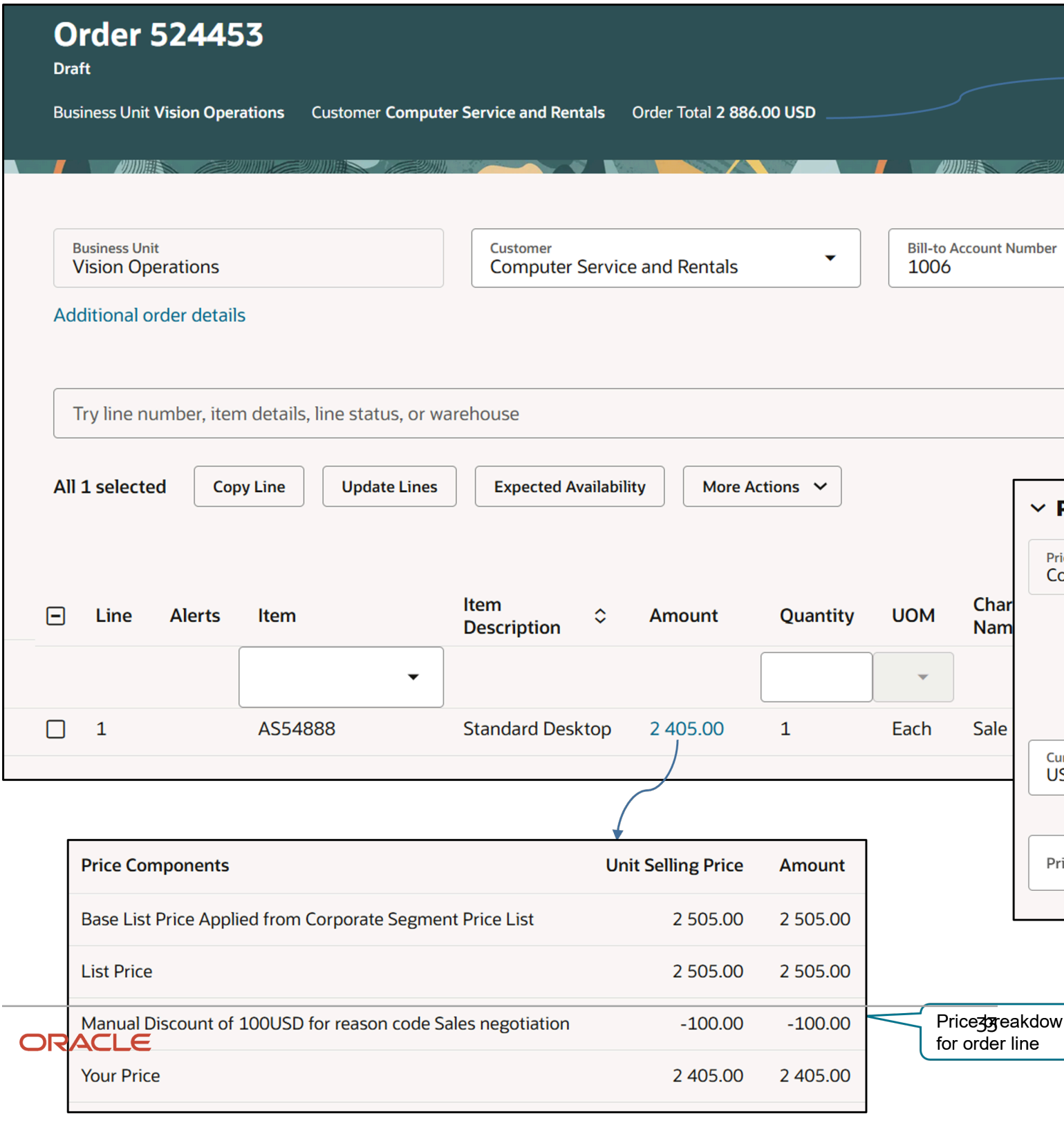
Get technical details:

- *Technical Reference for Oracle Pricing (KB171412)*
- See the Pricing chapter in *Tables and Views for Oracle Fusion Cloud SCM*.

A More Detailed Example

Let's use a different example to take a closer look at pricing.

Each sales order includes pricing details:


Order 524453
Draft
Business Unit Vision Operations
Customer Computer Service and Rentals
Order Total 2 886.00 USD
Business Unit
Vision Operations
Customer
Computer Service and Rentals
Bill-to Account Number
1006
Additional order details
Try line number, item details, line status, or warehouse
All 1 selected
Copy Line
Update Lines
Expected Availability
More Actions
Line
Alerts
Item
Item Description
Amount
Quantity
UOM
1
AS54888
Standard Desktop
2 405.00
1
Each
Price Components
Unit Selling Price
Amount
Base List Price Applied from Corporate Segment Price List
2 505.00
2 505.00
List Price
2 505.00
2 505.00
Manual Discount of 100USD for reason code Sales negotiation
-100.00
-100.00
Your Price
2 405.00
2 405.00
Price breakdown for order line

Note

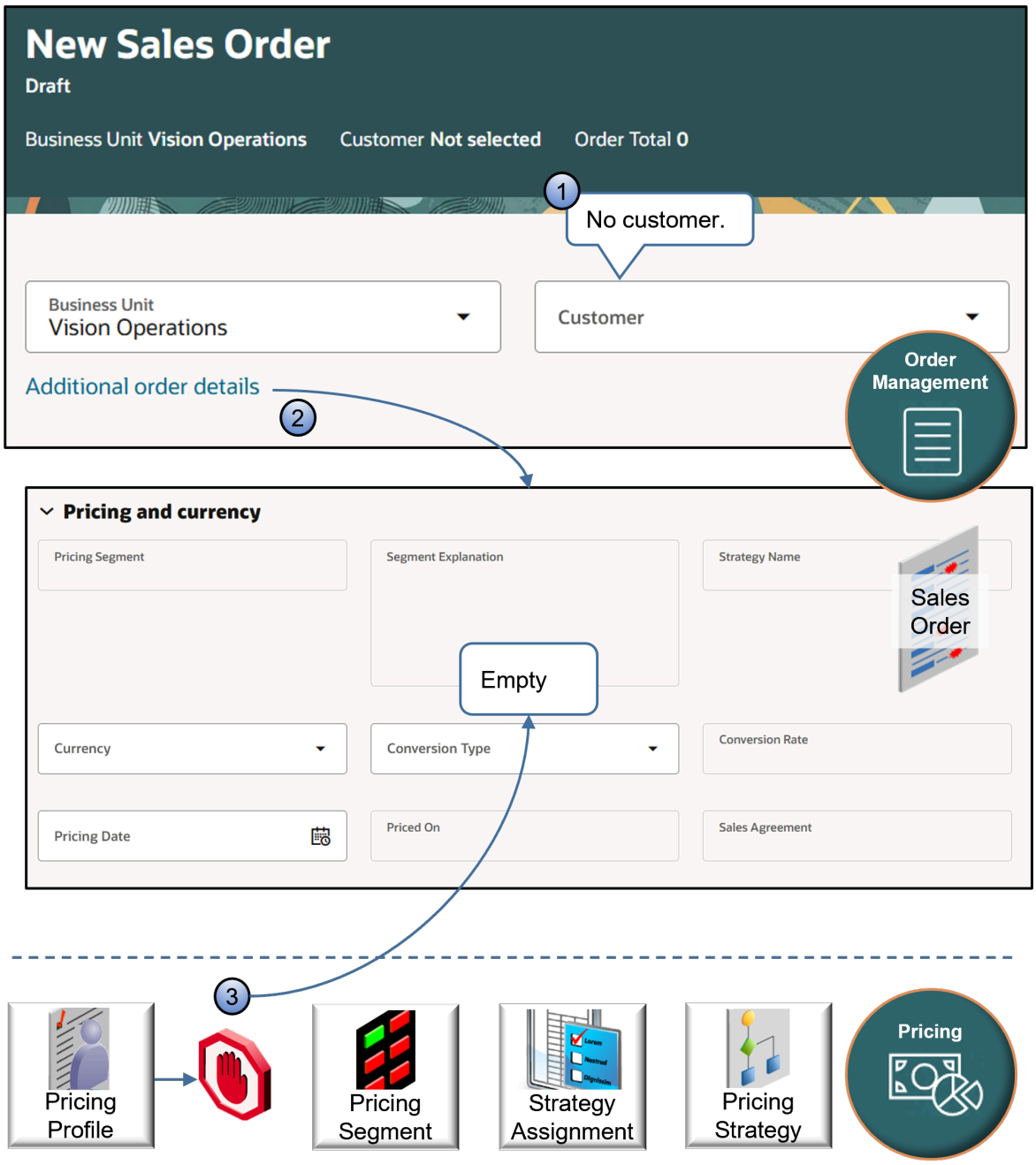
- Click **Additional order details** to examine details about the pricing strategy and pricing segment that Pricing applies to the sales order.
- Click the **price total** at the top of the sales order to view the charge breakdown for the entire sales order.
- Click the **amount** on the order line to view the charge breakdown for the order line.

How does it work? Try it:

1. Create a sales order in Order Management but don't set the customer.

2. Click **Additional order details**.

   Order Management calls Pricing to get the strategy and segment, but the call doesn't include the customer:



3. Pricing can't populate the profile without the customer, so its blocked from proceeding to determine the segment, assign the strategy, and apply the strategy. It returns nothing or it might return null in the strategy explanation.

   ```
   The applicable pricing strategy for the transaction is Pricing Strategy = Corporate Pricing Strategy,
    Precedence = 100 because Pricing Segment = null.
   ```

   Pricing sets the strategy to Corporate Pricing Strategy when it doesn't have enough detail, by default.

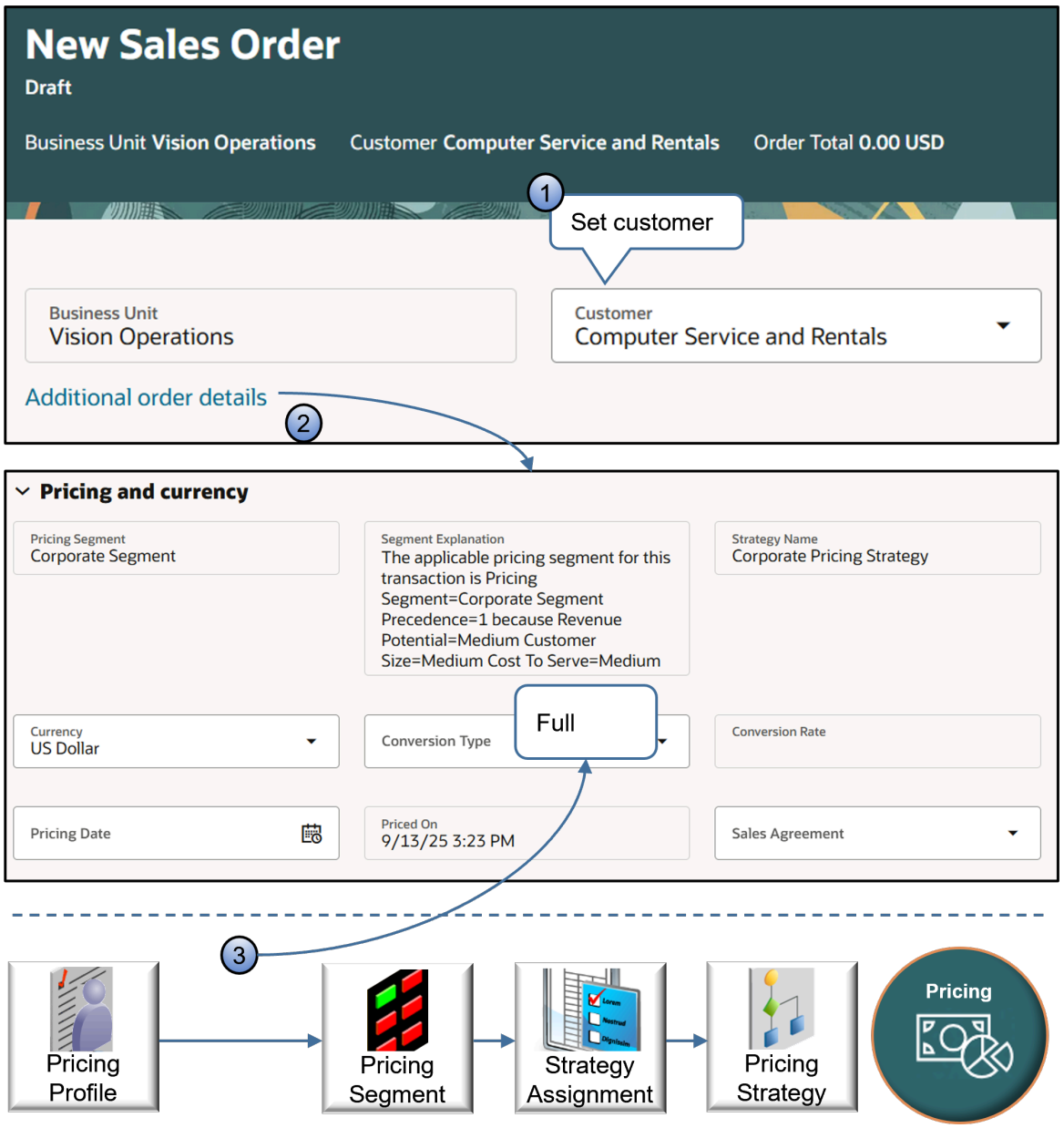
   Also, Pricing can't populate the currency because the strategy sets currency.

Now, try this:

1. Add a customer.
2. Click **Additional order details**.

3. This time, Pricing can populate the profile because it has the customer, it proceeds to determine the segment, assign the strategy, and apply the strategy.

   Pricing sends the results, and Order Management populates the dialog with strategy and segment details, and sets the currency:



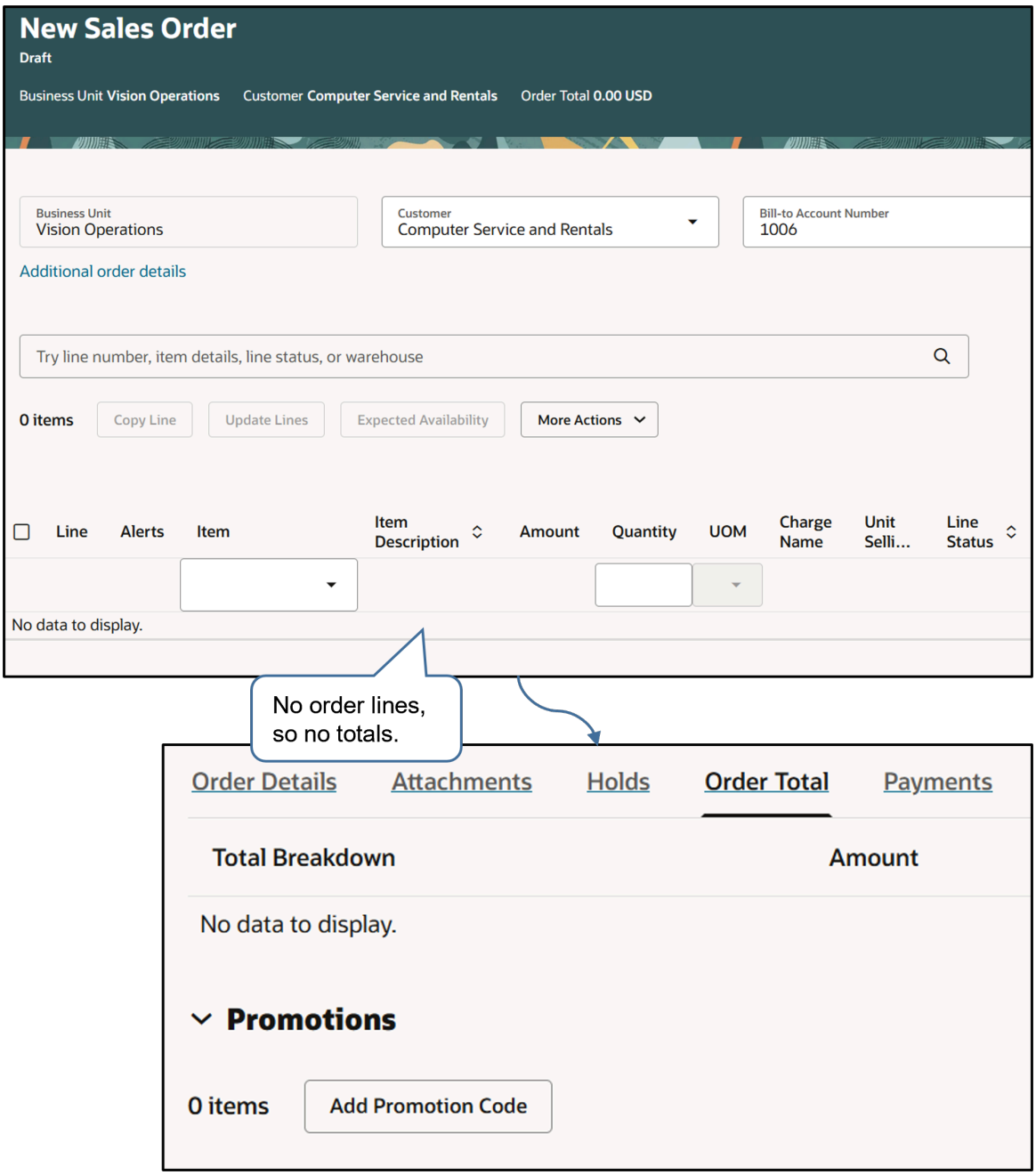
For example, it sets the segment to Corporate Segment and provides an explanation:

```
The applicable pricing segment for this transaction is Pricing Segment=Corporate Segment Precedence=1 because
Revenue Potential=Medium Customer Size=Medium Cost To Serve=Medium Customer Value=Medium Customer Rating=Medium
```

It sets the strategy to Corporate Pricing Strategy and provides an explanation.

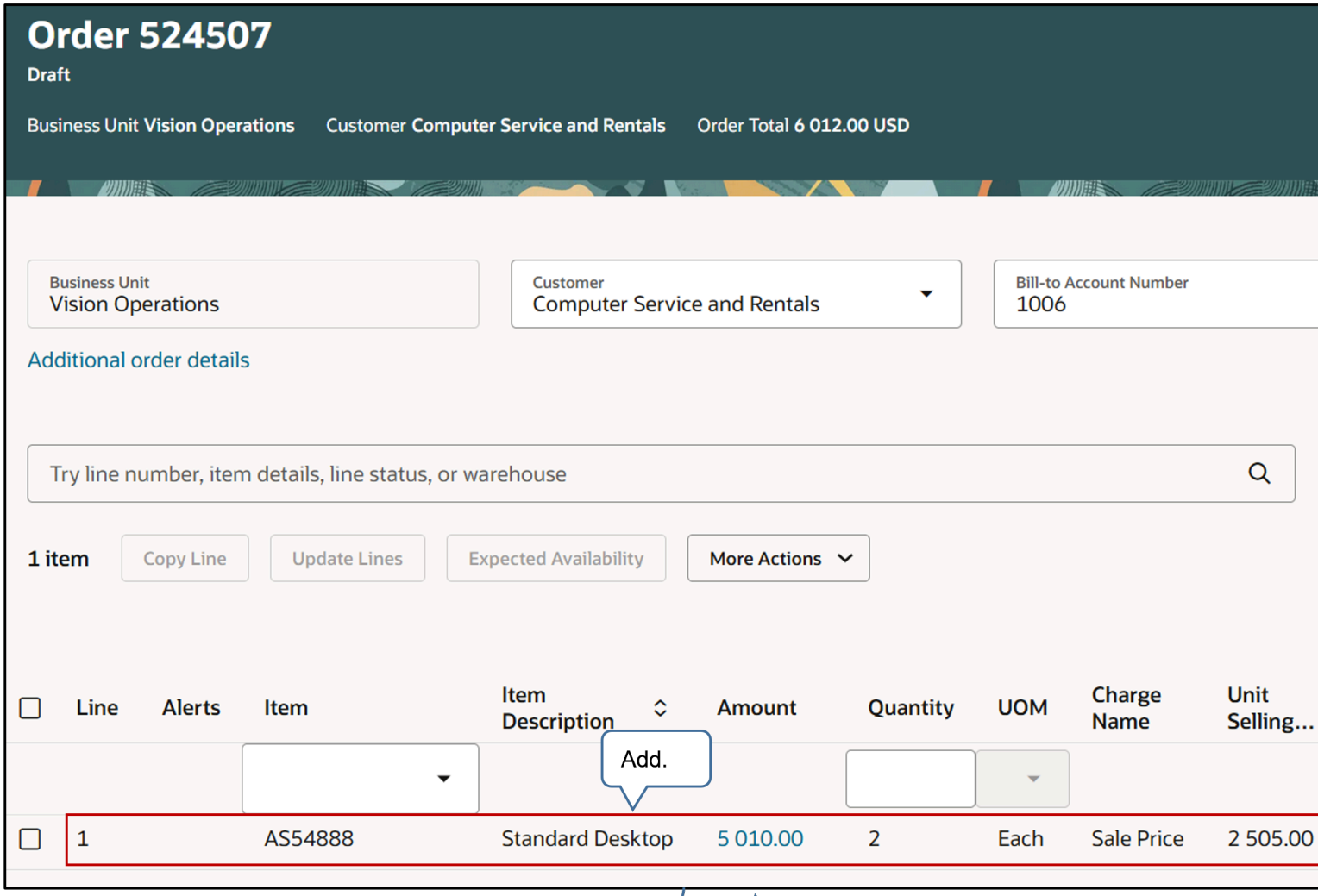
```
The applicable pricing strategy for this transaction is Pricing Strategy=Corporate Pricing Strategy because
Channel Method=null Transaction Type=null Pricing Segment=Corporate Segment
```

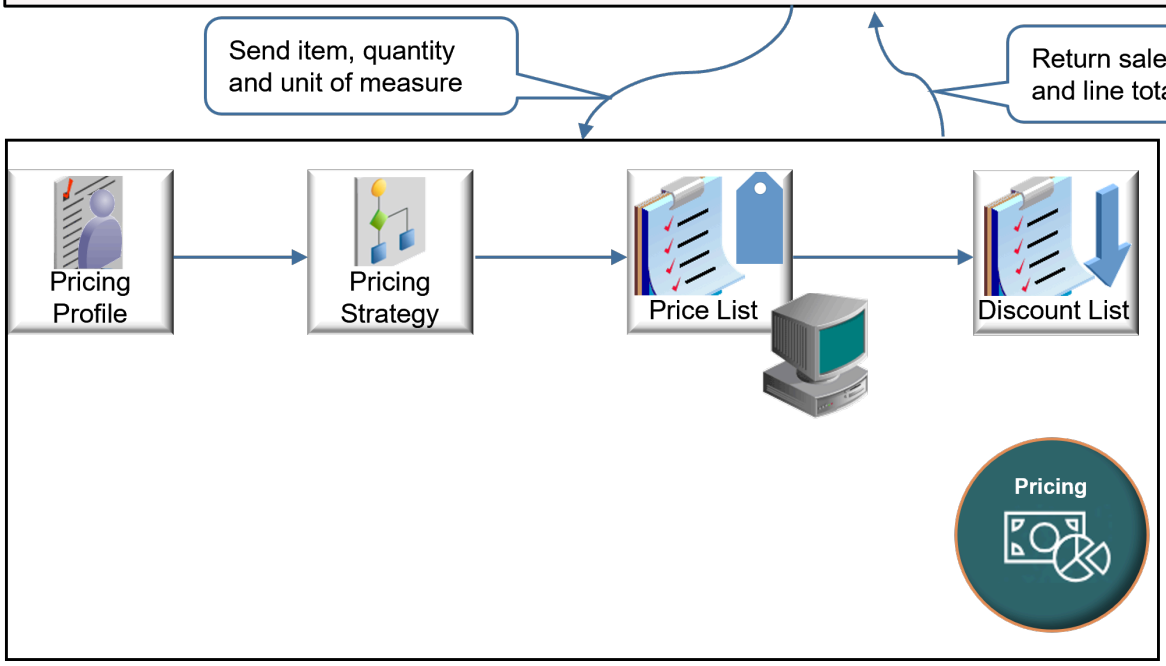
Next, notice the totals are zero because you haven't added an order line:



Assume you set up a price list named Corporate Segment Price List, added the AS54888 to it with a base list price of 2505, and added the price list to the Corporate Pricing Strategy.

You add an order line for the AS54888 item with a quantity of 2:

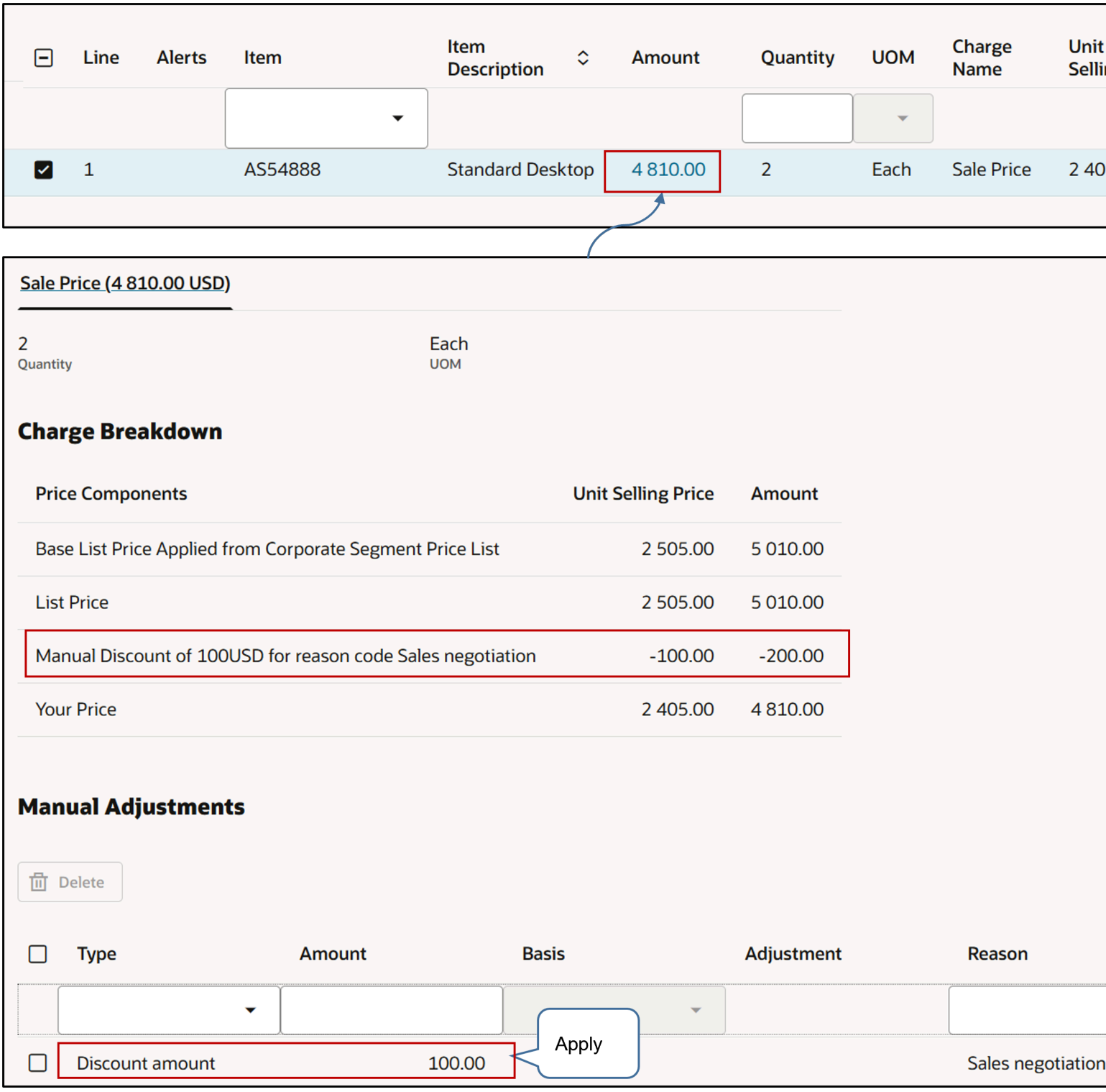

Order 524507
Draft
Business Unit Vision Operations
Customer Computer Service and Rentals
Order Total 6 012.00 USD
Business Unit
Vision Operations
Customer
Computer Service and Rentals
Bill-to Account Number
1006
Additional order details
Try line number, item details, line status, or warehouse
1 item
Copy Line
Update Lines
Expected Availability
More Actions
Line
Alerts
Item
Item Description
Amount
Quantity
UOM
Charge Name
Unit Selling...
Add.
1
AS54888
Standard Desktop
5 010.00
2
Each
Sale Price
2 505.00


Send item, quantity and unit of measure
Return sale price and line total
Pricing Profile
Pricing Strategy
Price List
Discount List
Pricing

Order Management sends the item, quantity, and unit of measure to Pricing with a request to price the item.

Pricing determines the strategy, uses the price list to determine the price, the discount list to determine the discount, prices the item, then returns the sale price charge and the order line total.

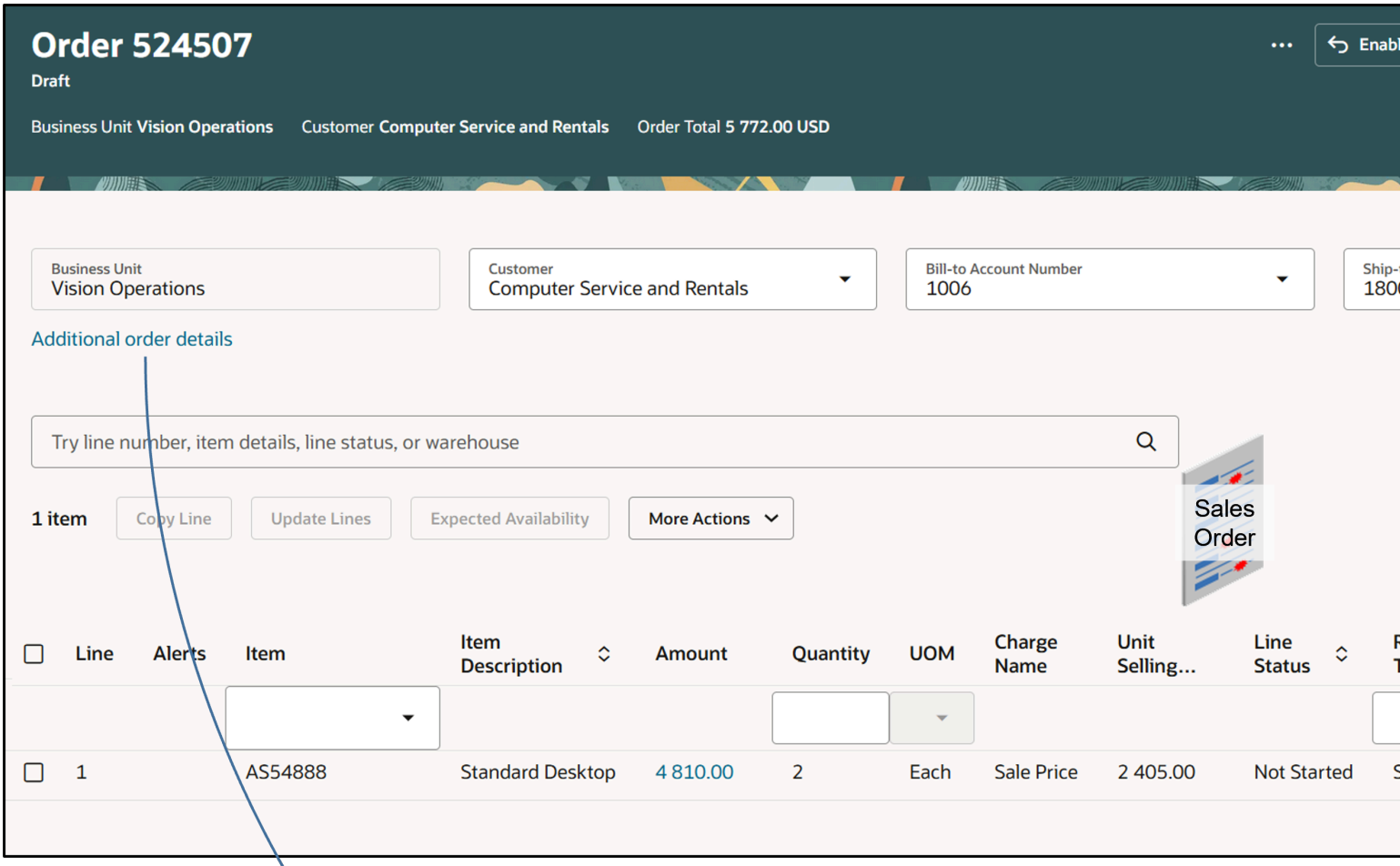
Next, click the 4,800 total on the order line, then manually apply a $100 discount. You added a quantity of 2, so the Line dialog includes unit price and amount column. The dialog identifies the price list that Pricing used:

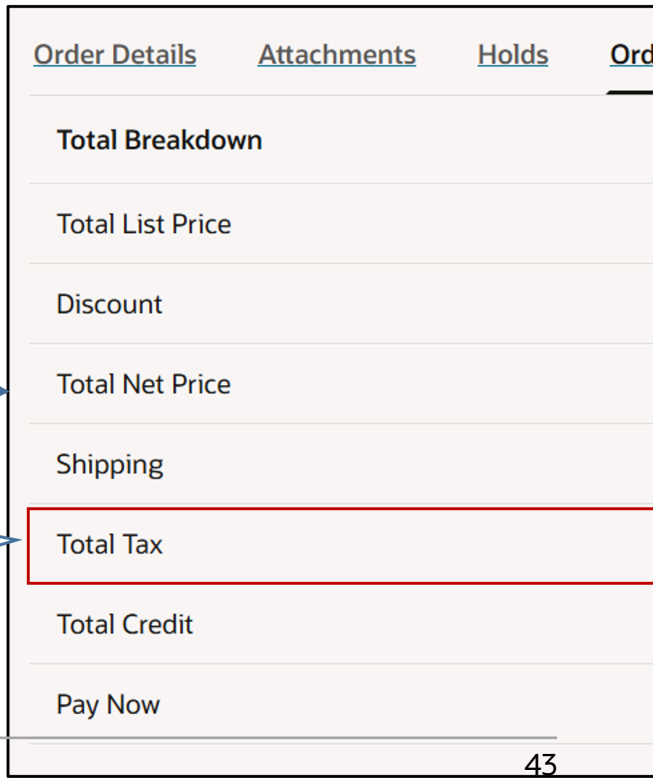


The Line dialog displays these values:

Price Component	Unit Price	Amount
Base List Price Applied From Corporate Segment Price List	2,505	5,010
List Price	2,505	5,010
Manual Discount of 100USD for reason code Sales negotiation	-100.00	-200.00
Your Price	2,405.00	4,810.00

Pricing doesn't calculate shipping and tax until you save or submit the sales order. Submit or save to update the total:





Pricing adds shipping, tax, and so on.

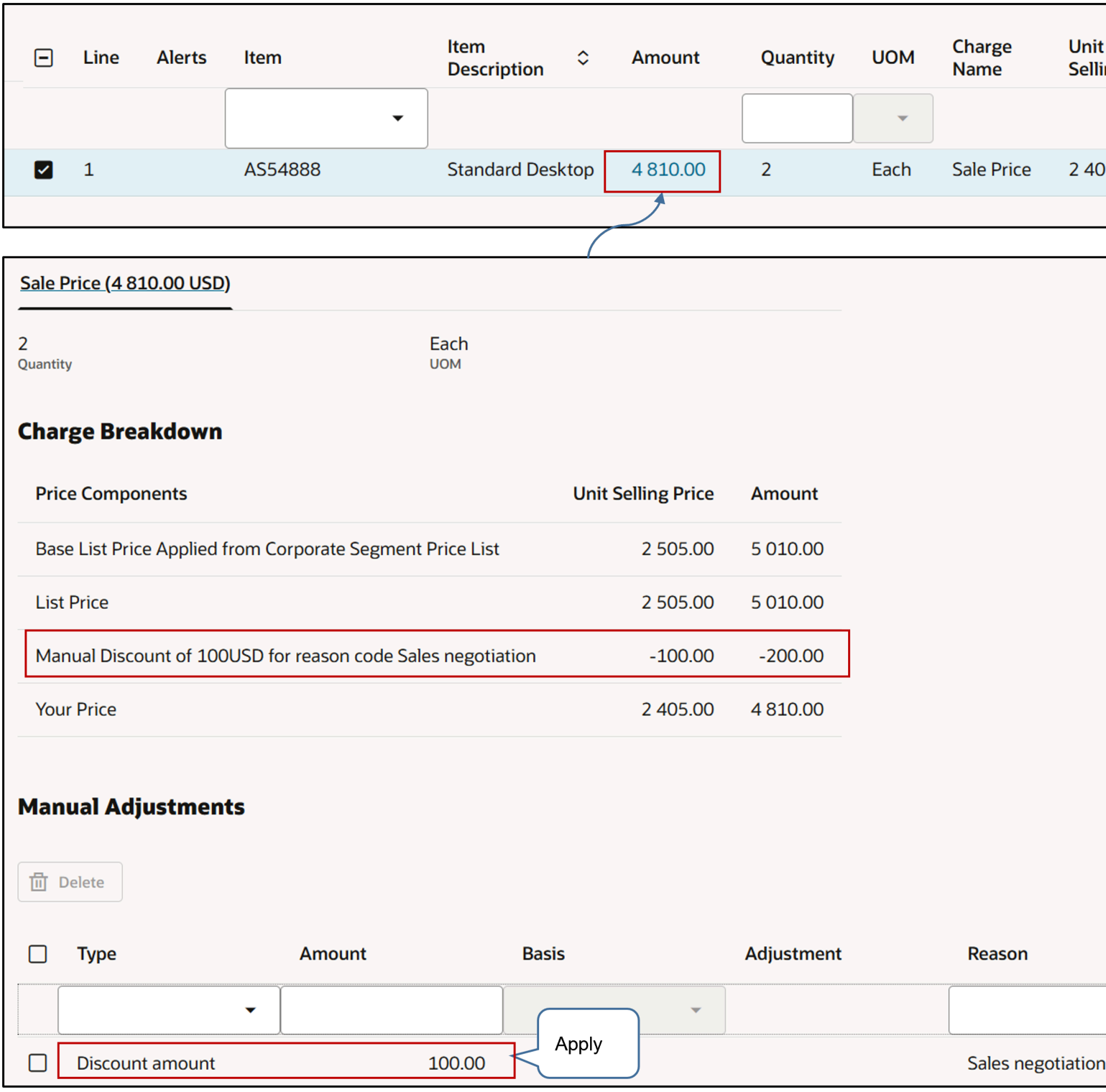
Pricing calculates shipping, tax, and other charges, depending on how you set up pricing. For example, it might add credit, margin, and so on.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- How Profiles, Segments, and Strategies Work Together
- Adjust Price for Pricing Rules
- Manage Pricing Promotions

How Pricing Calculates the Order Line

The user searches for an item on the order line.



Note

- The charge breakdown on the order line displays charges that Pricing calculates for the item according to strategy, segment, rules, and so on.

- Different charges display depending on how you set up pricing. For example, if you don't add a discount or markup, then the breakdown doesn't include the price list adjustment. In this example, you added a discount.
- Some charges are price points and some are adjustments. For example, Base List Price Applied from Corporate Segment Price List indicates that Pricing uses the Corporate Segment Price List to calculate the price point for the base list price.
- If the quantity is greater than 1, then the Unit Price column displays the price for one unit, and the Amount column displays the extended amount for the charge.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- How Profiles, Segments, and Strategies Work Together
- Adjust Price for Pricing Rules
- Overview of Define Tax Configuration

Roadmap to Manage Oracle Pricing

Use this road map to get the big picture of how to manage Oracle Pricing, including setup and administration.

- Sign into Oracle Applications. Make sure you have the privileges that you need to administer pricing. For details, see the Pricing Administrator chapter in *Security Reference for Order Management*.
- Do the setup first, then the administration.
- Do the work in the same sequence that each roadmap lists the topics.
- All steps are optional. The roadmap provides a generic approach to manage Pricing. You might find you only need to do a few steps, depending on the Pricing functionality you need.

Set Up Pricing

Get started. See these topics:

1. Manage Pricing Charge Definitions
2. Manage Price Elements
3. Manage Pricing Bases
4. Manage Pricing Parameters
5. Manage Rounding Rules

Set up user interfaces:

1. Manage Pricing Totals
2. Manage Pricing Results Presentations
3. Manage Pricing Messages
4. Manage Pricing Message Tokens
5. Manage Pricing Lookups
6. Manage Pricing Descriptive Flexfields

Administer Pricing

Administer profiles, segments, strategies, and lists:

1. Manage Pricing Profiles.
2. Manage Pricing Segments.
3. Manage Pricing Strategies. Create a pricing strategy but don't add any lists.
4. Manage Price Lists.
5. Manage Cost Lists.
6. Manage Discount Lists.
7. Manage Shipping Charge Lists.
8. Manage Currency Conversion Lists.
9. Manage Pricing Strategies. Add lists and other objects you created in steps 4 through 8 to the pricing strategy you created in step 3.
10. Assign Pricing Strategies.
11. Test your set up. Use an Oracle Application, such as Oracle Order Management, to verify pricing works correctly.

Administer algorithms, service mappings, and matrixes:

1. Manage Pricing Matrix Types
2. Map Pricing Operations to Pricing Algorithms
3. Manage Pricing Algorithms
4. Manage Service Mappings
5. Manage Matrix Classes

Quick Start

Use this section only if must do a few tasks, depending on the Pricing functionality you need.

Modify How Order Management Displays Pricing Details

Description	Topic
Modify how Pricing calculates the price details that Order Management displays for an order line, such as the unit price, list price, sale price, and discount.	Modify Prices on Order Lines
Specify the details that Order Management displays in the Line dialog.	Modify Pricing Algorithm Variables
Specify the details that Order Management displays in the charge breakdown.	Manage Pricing Totals
Specify how to display each price element in a charge breakdown. For example, how to display the rounding adjustment in the charge breakdown that Order Management displays in the Amount Sale Price dialog.	Manage Pricing Results Presentations

Modify Pricing Logic

Description	Topic
Specify the logic that Pricing uses to calculate price. For example, remove the tax calculation.	Modify Pricing Algorithms
Specify the logic that Pricing uses to calculate totals. For example, calculate credit before you calculate tax.	Manage Pricing Totals
Add an attribute to the pricing calculation. For example, add freight details to an order line.	Manage Service Mappings
Apply conditional logic. For example, apply a different percent discount depending on whether the customer resides in a North, South, East, or West geographical region.	Managing Matrix Classes
Specify how Pricing rounds the values it calculates. For example, round each price to 0.97 or 0.99 for companies that reside in the United States, and round each price to 88 for companies that reside in China.	Manage Rounding Rules
Create a pricing rule that controls how Pricing calculates the price for each item. For example: • Set the base price to $500 USD for a cell phone, and allow the user to manually adjust the price. • Add a 5% increase to the invoice price to capture cost of goods sold. • Provide an 8% discount if the customer purchases a recurring service for 12 months, such as a monthly service call. • Price shipping for a Desktop Computer at $50 for Standard Delivery and $100 for Express Delivery. • Set up the base price for a computer monitor at $400 USD (United States Dollar), and use a 1.38 conversion rate to offer it for sale in CAD (Canadian Dollar) at $553.88.	Pricing Rules
Use tier pricing or a pricing matrix to adjust the price that a pricing rule calculates.	Adjust Prices

Description	Topic
Specify a pricing charge definition that determines the total price of an item. For example, add a handling fee to an item that currently includes only a sale charge and an administration charge.	Manage Pricing Charge Definitions
Specify the basis that Pricing uses to calculate an adjustment according to a percent or amount. For example, sell to a distributor who ships items and handles freight and shipping, but exclude freight charges from your pricing basis.	Manage Pricing Bases

Modify Pricing for Strategies and Customers

Description	Topic
Use a pricing profile to categorize customers who exhibit similar characteristics. For example, categorize a customer as high size and high revenue.	Manage Pricing Profiles
Use a pricing segment to categorize customers, understand their business motivations, and to offer a pricing solution. Track pricing performance to improve revenue and profit margins.	Manage Pricing Segments
Create a pricing strategy that helps you achieve a profitability goal.	Manage Pricing Strategies

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- How Profiles, Segments, and Strategies Work Together
- Overview of Oracle Pricing

2 Profiles, Segments, and Strategies

Profiles and Segments

Manage Pricing Profiles

Use a pricing profile to describe the buying characteristics of each of your customers.

For example, profile a customer as high customer size, high customer value, medium customer rating, and high revenue potential.

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

Manage pricing profiles:

1. Determine the characteristics that describe each of your customers.
   In this example, assume you work for Vision Corporation, a fictitious company that sells desktop and laptop computers. You must administer pricing for these customers:
   - **Computer Service and Rentals, and Professional Computing Solutions**. Each of these businesses are large, long-established customers who have been selling to commercial, domestic corporations for a long time.
   - **Computer Associates International**. A medium size, up-and-coming, new customer who sells to governments and commercial enterprises in domestic and international markets.
   - **Computers Direct to U**. A brand new, just-launched company who sells direct to consumers.

   Vision Corporation determined that customers exhibit these characteristics:

Customer	Revenue Potential	Cost to Serve	Value	Rating	Size
Computer Service and Rentals	Very High	Medium	Very High	Very High	Large
Professional Computing Solutions	Very High	Medium	Very High	Very High	Large
Computer Associates International	High	High	Medium	High	Medium
Computers Direct to U	Low	Low	Low	Medium	Small

2. Sign into Oracle Pricing with administrative privileges.

3. Go to **Home > Order Management > Pricing Administration > Manage Customer Pricing Profiles**.
4. On the Manage Customer Pricing Profiles page, click **Actions > Create**.
5. In the Create Customer Pricing Profiles dialog, add values from the table in step 1 for Computer Service and Rentals, then click **Save and Close**.

   For the Start Date and End Date attributes, note that you can create only one pricing profile for each customer for each time period. You can't add a customer with more than one pricing profile in the same time period or in time periods that overlap one another.
6. Repeat steps 4 and 5 for Professional Computing Solutions.
7. Repeat steps 4 and 5 for Computer Associates International.
8. Repeat steps 4 and 5 for Computers Direct to U.
9. Add a pricing segment that Pricing can use to match each customer to a segment.

   For details, see *Manage Pricing Segments*.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments

Manage Pricing Segments

Use a pricing segment to categorize sets of customers, understand their business motivations, and offer a pricing solution. Use it to track pricing performance and analyze similar customer situations to help you improve revenue and margin.

Assume you must add a corporate segment, international segment, and a domestic segment.

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

Manage pricing segments:

1. Go to **Home > Order Management > Pricing Administration > Manage Pricing Segments**.
2. On the Manage Pricing Segments page, click **Create Pricing Segment Matrix**.

   If the Manage Pricing Segments page already displays a pricing matrix, then you must click **Delete Pricing Segment Matrix**, click **Yes** in the Warning dialog, close and reopen the Manage Pricing Segments page, then click **Create Pricing Segment Matrix**. Oracle Pricing can maintain only one pricing matrix for each deployment.
3. In the Create Pricing Segment Matrix dialog, add a check mark for each condition you must include in the matrix, then click **OK**.

   Add a check mark to each condition that you require. If a row in your matrix doesn't require a condition, then you can leave it empty, and Pricing ignores the condition only for this row. Adding each condition simplifies future maintenance if you must expand your matrix at some time in the future.
4. On the Manage Pricing Segments page, click **Actions > Add Row**.

5. Add values for the first row of this matrix:

Revenue Potential	Customer Size	Cost to Serve	Customer Value	Customer Rating	Pricing Segment	Precedence
Very High	Large	Medium	Very High	Very High	Corporate Segment	1
High	Medium	High	Medium	High	International Segment	2
Low	Small	Low	Low	Medium	Domestic Segment	Not applicable

6. Click **Actions > Add Row**, then add values from the second row of the matrix in step 5.
7. Click **Actions > Add Row**, then add values from the third row of the matrix in step 5.
8. Create a pricing strategy to make sure your pricing logic meets the pricing objective.

   For details, see *Manage Pricing Strategies*.

Note

- Pricing compares the pricing profile to each row of the pricing matrix until it finds a match. If more than one row matches, then Pricing uses the row that contains the lowest value in the Precedence.

  For example, if the first and second row match, and if the first row contains a precedence of 1, and the second row contains a precedence of 2, then Pricing uses the first row. For details, see *How Profiles, Segments, and Strategies Work Together*.
- Pricing might ignore each attribute in the pricing matrix that doesn't include a value, depending on whether the matrix class allows an empty value. For example, consider this row, where - indicates an empty value:

Revenue Potential	Customer Size	Cost to Serve	Customer Value	Customer Rating	Pricing Segment
-	-	-	-	Very High	Corporate Segment

  If the matrix class allows an empty value, then this row assigns the Corporate pricing segment to every pricing profile that includes a Very High rating. It ignores the empty attributes, such as revenue, size, and value.

Add a Default Pricing Segment

We recommend that you add at least one row that Pricing can use as the default pricing segment:

- Leave values in the condition columns empty. This way, Pricing won't apply any conditions when determining whether to use the pricing segment.
- Set the Precedence to a value that's lower than any other segment. This way, Pricing will use your default segment only after it has determined that no other segments apply.
- Make sure you assign a pricing strategy to your default segment. For details, see *Assign Pricing Strategy*.

This will help to avoid run time errors that might happen, such as:

- `Pricing strategy was not determined for the transaction`
- `No matching pricing segment found for the transaction`

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Manage Pricing Strategies
- Assign Pricing Strategy

Set Up Pricing Without Customer Profile

Set up pricing without managing customer profiles under certain conditions.

If

- Your set up includes only one price list, one discount list, and one shipping list, and.
- All your customers use only one pricing strategy.

You don't need to create a pricing profile for each of your customers.

Assume you use the Corporate Pricing Strategy for all your customers. Here's your set up:

1. In the Pricing Administration work area, click **Tasks > Manage Pricing Strategy Assignments**.
2. Make sure the Manage Pricing Strategy Assignments page includes only one row:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	All

3. Create an assignment matrix that includes only one row.

   Leave all the condition columns empty. Set only the pricing strategy:

Attribute	Value
Pricing Strategy	Corporate Pricing Strategy

Strategies

Pricing Strategies

A pricing strategy is a set of pricing rules that you set up to achieve a profitability goal for selling and pricing an item.

- The pricing strategy specifies the price, cost, discount, currency conversion, shipping rules, and return price that Pricing uses to help meet the pricing objective of the pricing strategy.
- The pricing objective is the goal of your pricing policies for a customer, such as to create interest about your items, increase sales volume, or maintain a market position that you already established.

  For example, create a pricing strategy according to competitor pricing, to sell at a loss so you can enter a market, to price an item differently for each pricing segment, or to price it high so that customers perceive it favorably.
- The pricing strategy references the prices and pricing policies that Pricing applies to help you accomplish the objective.
- You can group pricing rules in a pricing strategy to control pricing behavior.
- Add a price list, cost list, discount list, or currency conversion list that you haven't approved to a strategy, then approve the strategy. However, Pricing only uses approved lists at run time.
- Specify whether you can override the default value for the price list.

Currency

You can specify a default currency and specify whether you can override it when you create a sales order. You can override the default currency only with a currency that the pricing strategy references.

Behavior for repricing a sales order is different depending on when you set the currency:

Set Currency When You Create the Sales Order	Don't Set Currency When You Create The Sales Order
Order Management uses the original currency that you set on the order even if you change the currency on the pricing strategy in the Pricing Administration work area or directly on the sales order.	Order Management uses the currency that's currently defined on the pricing strategy in the Pricing Administration work area or that you set on the sales order.

For more:

- Set currency on a sales order, see *Manage Other Behavior on Sales Orders*.
- *Round Values in Header Currency for Coverages and Configured Items*

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Assign Pricing Strategy
- Manage Price Lists
- Pricing Guidelines
- Manage Other Behavior on Sales Orders

Manage Pricing Strategies

Create a pricing strategy to achieve a profitability goal.

Summary of the Set Up

1. Create the pricing strategy.
2. Add lists to the pricing strategy.

Here are details about the pricing strategy that you set up in this topic:

- Maximize profit.
- Reference the Operations business unit of your Computer Service and Rentals customer.
- Reference various lists in Pricing to set the price, cost, discount, shipping charge, and currency conversion.

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

Create the Pricing Strategy

1. Go to **Home > Order Management > Pricing Administration > Manage Pricing Strategy**.
2. On the Manage Pricing Strategy page, click **Actions > Create**.
3. In the Create Pricing Strategy dialog, set the values, then click **Save and Edit**:

Attribute	Value
Name	Enter `Corporate Pricing Strategy`.
Business Unit	Computer Service and Rentals Operations.
Default Currency	USD. Select the currency that Pricing displays at run time for items that you associate with this pricing strategy.
Default GL Conversion Type	Leave empty. Select the currency conversion type that Pricing must use if it can't find the conversion list when it converts currency in the general ledger.

Attribute	Value
Objective	Profit Maximization. Set the Objective to help clarify the purpose of the strategy. This attribute doesn't affect pricing calculations.
Allow Price List Override	Leave empty. Allow you to select some other price list at run time. Add a check mark to this option only if you modify a pricing algorithm so it supports a price override. For details, see *Override Base List Price*.
Allow Currency Override	If you set Default Currency to USD, and if you: o Set the Allow Currency Override attribute to Yes, then Pricing uses USD by default, but also allows you to convert USD to GBP (Pound Sterling) or to EUR (Euro). If you change the currency, then Pricing uses the currency that the Order Entry Specialist selects when it calculates the price. o Import a source order that isn't already priced, and if the source order uses GBP or EUR, then Pricing will use GBP or EUR to price the source order regardless of how you set Default Currency or Allow Currency Override. If you don't set up a pricing strategy, then Pricing allows you to select USD, GBP, or EUR, and Pricing will use the currency that you select when it calculates the price.

Add Lists to the Pricing Strategy

Add the lists that this pricing strategy must reference, such as price lists, discount lists, and so on. You must create these lists before you can add them to this pricing strategy:

1. Create the lists that the pricing strategy must reference:

List Name	Get Details
Price List for Computer Service and Rentals	*Manage Price Lists*
Cost List for Standard Desktop	*Manage Cost Lists*
Discount List for Standard Desktop	*Manage Discount Lists*
Shipping Charge List for Standard Desktop	*Manage Shipping Charge Lists*

List Name	Get Details
Currency Conversion List for Standard Desktop	*Manage Currency Conversion Lists*

2. On the Edit Pricing Strategy page, in the Pricing Rules tab, in the Segment Price Lists area, click **Actions > Select and Add**.

   Use each area in the Pricing Rules tab, such as Segment Price Lists, to add a price list.
3. In the Select and Add dialog, search for Price List for Computer Service and Rentals, then click **OK**.
4. Go to the Price List page, then close it.
5. On the Edit Pricing Strategy page, click **Approve**.

   Pricing sets the pricing strategy status to In Progress when you create a pricing strategy, by default. You must approve the pricing strategy so you can use it in other parts of Pricing Administration.
6. In the Segment Price Lists area, in the Name column, click **Price List for Computer Service and Rentals**.
7. On the Price List page, click **References**, then verify that the Name displays Price List for Computer Service and Rentals.

   Pricing displays each pricing strategy that references this price list.
8. Repeat steps 2 through 7 for each of the other lists you added in step 1.

   For example, to add Cost List for Standard Desktop, on the Edit Pricing Strategy page, click **Costs Lists**, add the cost list, navigate to the Edit Cost List page, click **Close Tab**, and so on:

   - You can add the same list to a pricing strategy more than one time, and you can assign a different time frame each time you add it, or you can add can add more than one list and set different time frames for them.

     For example, add a discount list that provides a 15% discount, and set the dates so its in effect from January through March the first time you add it, and in effect June through July the second time you add it.
   - If you add more than one list in an area of the Pricing Rules tab or Shipping Rules tab, then Pricing applies them according to the lowest number you set in the Precedence attribute.

     For example, assume you add two lists in the Segment Price Lists area. If you set the precedence for list x to 1, and precedence for list y to 2, then Pricing applies list x first, and then list y.
9. Assign the pricing strategy.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Assign Pricing Strategy
- Manage Price Lists
- Pricing Guidelines

Assign Pricing Strategy

Assign the pricing strategy that Pricing uses to price an item according to attributes, such as the pricing segment, channel, and type of transaction.

Assume you must assign all sales orders that are inside sales for all customers who reside in the Corporate segment. You must assign these sales orders to the Corporate strategy.

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

Assign a pricing strategy:

1. Go **Home >Order Management > Pricing Administration > Manage Pricing Strategy Assignments**.
2. Examine the predefined assignments. If a predefined assignment meets your business requirement, then don't create a new one, and exit this procedure.
3. Immediately below the page title, click **Actions > Add Row**, then set the values:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	Sales Orders
Start Date	Select today's date. The combination of Assignment Level, Pricing Context, and Transaction Type must specify a unique value for each time period. You can't create identical pricing strategy assignments that include dates that overlap one another.

4. Click **Save**.
5. Click **Create Assignment Matrix**.
6. In the Create Assignment Matrix dialog, add a **check mark** to each option, then click **OK**.
7. In the Header-Sales area, click **Actions > Add Row**, then set the values:

Attribute	Value
Channel Method	Inside Sales
Pricing Segment	Corporate Segment
Transaction Type	Sales Order

Attribute	Value
Pricing Strategy	Corporate Strategy If you create more than one row in the assignment matrix, then the combination of Assignment Level, Pricing Context, and Transaction Type must specify a unique value for each time period, and these dates must not overlap one another.

Use the matrix class when you set up Pricing to determine the attributes that you can specify in the assignment matrix. For details, see *Manage Pricing Matrix Types*.

8. Click **Save**.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Strategies
- Manage Pricing Matrix Types

Sort Records in Pricing Strategies

Use the Up One and Down One arrows on the Edit Pricing Strategy page to change the sequence of records that display according to the Precedence attribute.

Use them in different areas of the Pricing Administration work area:

- Segment Price List
- Ceiling Price List
- GSA Price List
- Floor Price List
- Discount List
- Shipping Charge List
- Charge Guidelines
- Currency Conversion List

To make sure the arrows work correctly, remove all filter criteria from Query by Example, save any new records that you added, then sort your records in ascending order by Precedence.

3 Lists

Rules

Pricing Rules

Set up a pricing rule to control how Oracle Pricing calculates the price for each item.

A pricing rule is a statement that controls how Pricing applies a price adjustment on an item. For example, how to apply a discount on the list price of an item.

You add a pricing rule when you create a list, such as a price list:

- Apply more than one charge to an item. For example, create a charge for a one-time sales price for a desktop computer in one pricing rule, then apply another charge for the maintenance service that recurs monthly in another pricing rule.
- Create a rule that calculates the base price, list price, price adjustment, discount, return charge, shipping charge, and so on according to a set of conditions and results.
- Create a currency conversion rule that manages price for a currency.

A pricing entity is an object that stores the details that Pricing uses to price an item. A pricing strategy, pricing segment, customer pricing profile, price list, cost list, and discount list are each an example of a pricing entity.

You can apply a rule to a pricing entity:

Pricing Entity	Example Rule	Get Details
Price list	Set the base price to $500 USD for a cell phone, and allow you to manually adjust the price.	*Manage Price Lists*
Cost list	Add a 5% increase to the invoice price to capture cost of goods sold.	*Manage Cost Lists*
Discount list	Provide an 8% discount if the customer purchases a recurring service for 12 months, such as a monthly service call on a commercial kitchen appliance.	*Manage Discount Lists*
Shipping charge list	Price shipping for a Desktop Computer at $50 for Standard Delivery and $100 for Express Delivery.	*Manage Shipping Charge Lists*
Currency conversion list	Specify the base price for a computer monitor at $400 USD (United States Dollar), and use a 1.38 conversion rate to offer it for sale in CAD (Canadian Dollar) at $553.88.	*Manage Currency Conversion Lists*

Guidelines

- You use a separate page to specify each type of pricing list. For example, you use the Price Lists page to specify a price list.
- The Pricing Administration page of the Pricing Administration work area displays pricing rules that you recently updated. You can search it to locate other pricing rules.
- You must click **Approve** on the page that you use to create the list so you can add it to a pricing strategy and so the pricing algorithm can use it.
  Use can sign in with Oracle Pricing's or Oracle Order Management's administrative privileges to edit a pricing rule. However, Pricing will display Approve only if you sign in with pricing privileges, and only if you didn't already approve the list.
- You can set up a list, add it to a pricing strategy, and then verify your work. The behavior is similar for any type of list that you create. For details, see *Manage Pricing Strategies*.
- You can create a tier pricing rule that adjusts price according to the number of items that the customer orders. For example, apply a 10% discount on all items when the customer buys two or three computers, and apply a 15% discount on all items when the customer buys four or more computers. For details, see *Adjust Price for Pricing Rules*.
- Add a pricing matrix. Use it to adjust the charges that you specify in each list according to a set of conditions and the value of an attribute. For details, see *Adjust Price for Pricing Rules*.
- You use the Edit Rules Table Columns action to edit a rule, such as the rule's conditions.
- If your rule references another object, then you can't use Edit Rules Table Columns to delete a part of the rule. You must first remove the reference, and then do the delete. For example, assume you want to delete a condition column from an assignment rule, the column references the Cost List Charge Adjustment matrix type. You must first delete the reference to the matrix type, and then delete the condition column.

Access Set

You must click **Access Set** when you create a list, and then add an access set.

The access set identifies the sales orders that Pricing will process. For example, if you select Vision Operations Business Unit Set for the access set, then Pricing will only process sales orders that reference the Vision Operations business unit.

Set the Access Set attribute to Common to process sales orders for all business units.

Create Duplicate Lists

Copy list details to a new list, including pricing rules, modify the new list, then save your modifications. Copying a list helps you create a new list and avoid errors. You click the Duplicate icon to duplicate a list.

For example:

1. Go to **Home >Order Management > Pricing Administration > Discount Lists**.
2. Search for the list you must copy.
3. In the search results, click **Duplicate**.
   Use the dialog that displays to specify how to copy the list, such as whether to copy discount lines from the original discount list.

Note

- If you modify the currency when you duplicate a list, then you might not be able to copy the charges that the original list references.

For example, assume you duplicate a cost list and modify the currency to CAD (Canadian Dollar). You might not be able to copy the charges because Pricing created them for some other currency, such as USD (United States Dollar).

- If you duplicate charges, then you can copy all charges or only the most current set of charges.

Examine Old Pricing Rules

The Pricing Administration page in Pricing Administration comes predefined to display pricing rules that you recently updated. You can also search for older pricing rules that have expired or are about to expire:

1. Go to the Pricing Administration work area, then click **Search > Advanced**.
2. On the Search Pricing Rules page, in the Advanced Search area, set the values, then click **Search**.
3. In the search results, use Rule End Date to filter the results, such as Before 12/28/19 12:00 AM.

Related Topics

- Manage Price Lists
- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Strategies
- Adjust Price for Pricing Rules
- Troubleshoot Your Pricing Setups

Attributes You Use in Pricing Rules

Get details about the Attributes that you can use in pricing rules.

Attribute	Description
Business Unit	Business unit where Pricing applies the pricing strategy. The business unit helps you organize and report on pricing and charges according to a management hierarchy, and to make sure Pricing processes data securely for the business unit.
Item	The item that you must add to the list, such as the AS54888 Desktop Computer. If you can't locate the item you must add, then you must add it in the Product Management work area. For details, see *Implementing Product Management*.
Pricing UOM	The unit of measure, such as Each.
Primary Pricing UOM	Add a check mark to use the value that you set in the Pricing UOM attribute as the primary pricing unit of measure. For details, see the Defining Units of Measure section of this topic.
Service Duration Period	If the item you add in the line is a service, such as a maintenance service agreement for a computer system, then set the service duration period.

Attribute	Description
Service Duration	If the item that you add in this line is a service, then set the service duration.
Associated Items	If the item you add in this line is a configured item, then click the icon in Associated Items to view the configurable options.
Line Type	Select a value: • **Buy.** The customer is buying the item. • **Return.** The customer is returning the item.
Start Date and End Date	Specify the time period when the rule is available for processing. To create a rule that never expires, leave the End Date empty.

Related Topics

- Manage Price Lists
- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Strategies
- Adjust Price for Pricing Rules
- Troubleshoot Your Pricing Setups

Prices

Manage Price Lists

Set up a price list that sets the price for each item that you sell.

A price list is a collection of prices for items that you target for a set of customers, and for a period of time. Use it to set the base list price and make other adjustments for each item:

- Associate more than one price list with more than one pricing strategy.
- Create more than one price list, then use pricing profiles, pricing segments, and pricing strategies to reference your prices lists to one or more customers.
- If you add a charge to an approved price list, then Pricing automatically approves the pricing entities that the charge references. It approves them for use in the price list.
- Set up the pricing charges that a price list contains for each item that the list references. You can set up a variety of charges. For example:

Type of Charge	Description
One Time Charge	A one-time charge for an item, such as a one-time fee to establish a phone service.

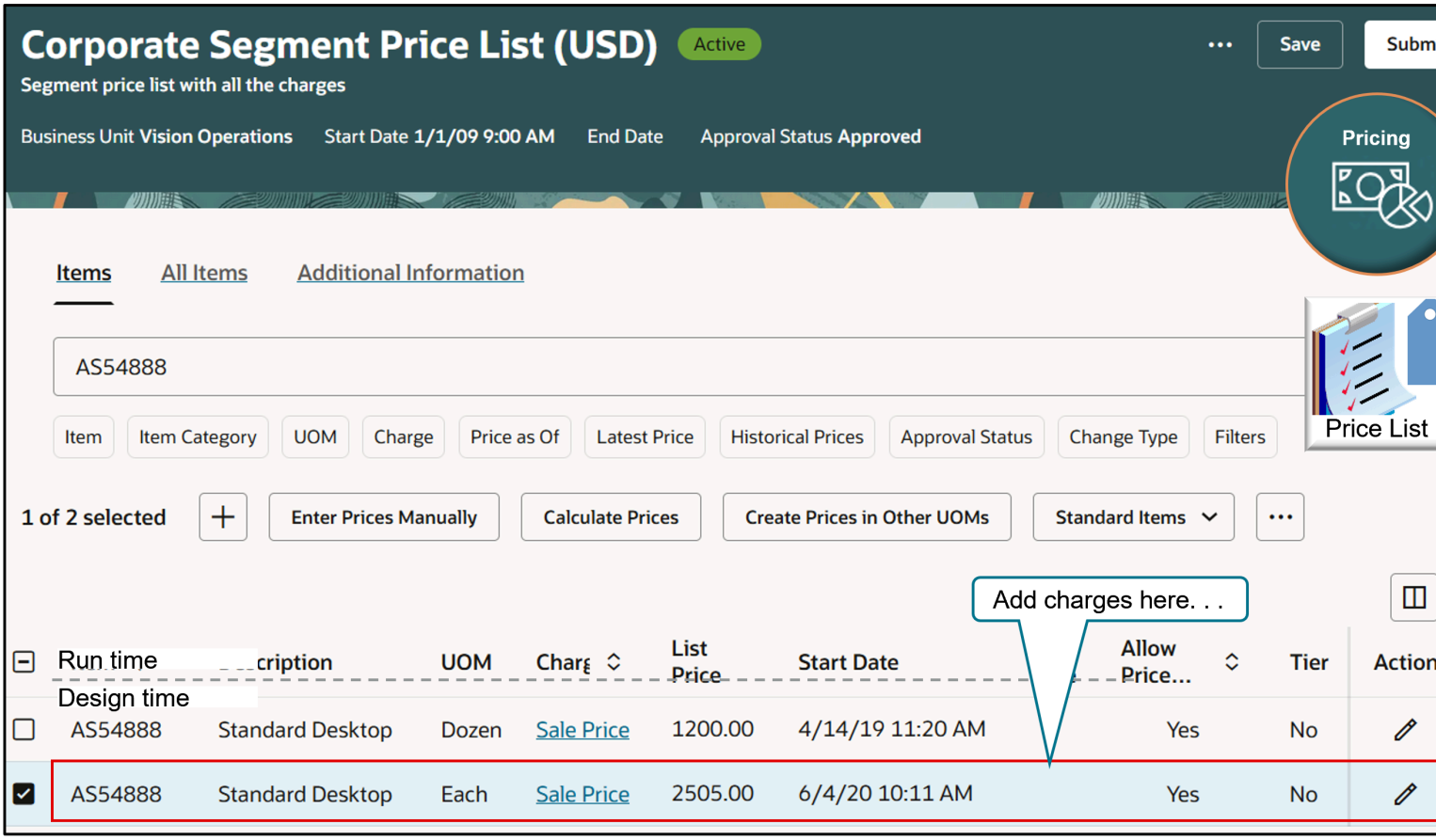
Type of Charge	Description
Recurring Charge	A periodic charge, such as the monthly recurring charge for a phone service.

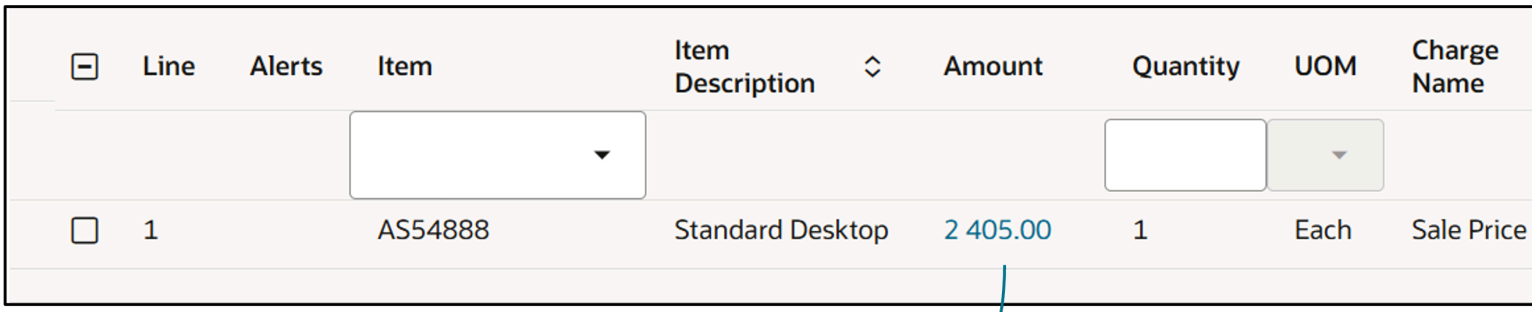
- Learn about the technologies that you can use to manage price lists. For details, see *Select a Technology to Manage Your Pricing Data*.

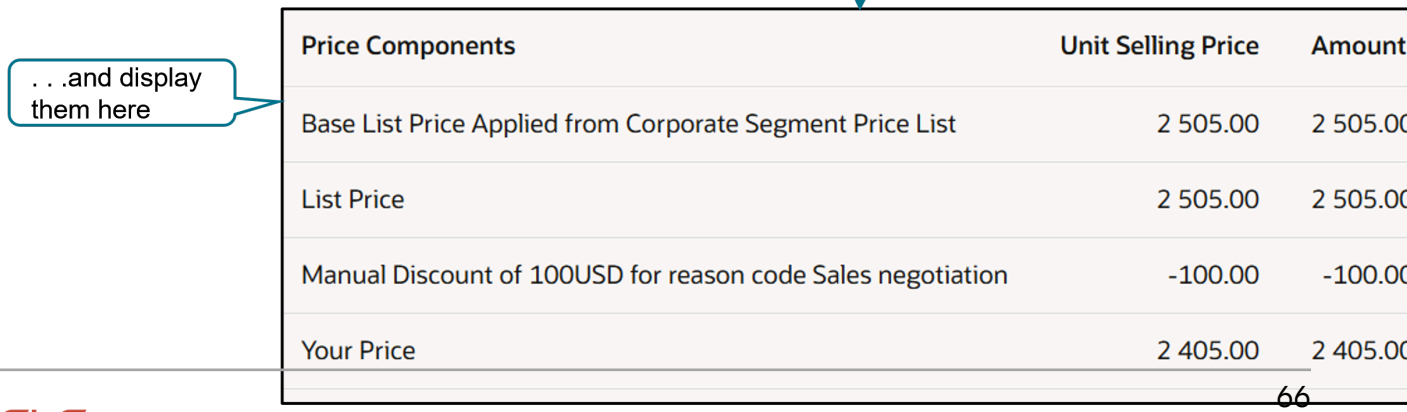
You will add a price list that sets the base price for the AS54888 Desktop Computer:

- $2,500 for each computer
- $10,000 for a box of five computers

You use the Price List page in the Pricing Administration work area at design time to calculate price on the New Sales Order page in the Order Management work area at run time:


Corporate Segment Price List (USD)
Active
Segment price list with all the charges
Business Unit Vision Operations
Start Date 1/1/09 9:00 AM
End Date
Approval Status Approved
Save
Subm
Pricing
Items
All Items
Additional Information
AS54888
Item
Item Category
UOM
Charge
Price as Of
Latest Price
Historical Prices
Approval Status
Change Type
Filters
Price List
1 of 2 selected
Enter Prices Manually
Calculate Prices
Create Prices in Other UOMs
Standard Items
Add charges here. . .
Run time
Design time
Item
Description
UOM
Charge
List Price
Start Date
Allow Price...
Tier
Action
AS54888
Standard Desktop
Dozen
Sale Price
1200.00
4/14/19 11:20 AM
Yes
No
AS54888
Standard Desktop
Each
Sale Price
2505.00
6/4/20 10:11 AM
Yes
No


Design time
Run time
Line
Alerts
Item
Item Description
Amount
Quantity
UOM
Charge Name
1
AS54888
Standard Desktop
2 405.00
1
Each
Sale Price


. . .and display them here
Price Components
Unit Selling Price
Amount
Base List Price Applied from Corporate Segment Price List
2 505.00
2 505.00
List Price
2 505.00
2 505.00
Manual Discount of 100USD for reason code Sales negotiation
-100.00
-100.00
Your Price
2 405.00
2 405.00

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

This topic describes how to create a pricing rule. For details, see *Pricing Rules*.

Summary of the Setup

1. Create price list.
2. Add price list to pricing strategy.

   You only add the price list to a pricing strategy the first time you create the price list. Pricing automatically adds any changes that you make after you add the price list.

Create Price List

1. Go to **Home > Order Management > Pricing Administration > Price Lists**.
2. On the Price Lists page, click **Create Price Lists**, set these values, then click **Create**:

Attribute	Description
Name	Price List for Computer Service and Rentals
Business Unit	Vision Operations
Currency	USD
Start Date	Any day before today
Default Charge	Sale Price

3. On the Price List for Computer Service and Rentals (USD) page, click **Add**, then add the AS54888 item.

   Note
   - You can select one or more items, then add them all to your price list with a single click.
   - Use keywords and filters to search for the items that you want to add.
   - If you set up the Primary Catalog pricing parameter, then the dialog will display all the items that are in the catalog. See *Manage Pricing Parameters*.
   - Add items that are in the primary UOM.
   - If you already added the item to your price list, then the Add Items dialog won't display it.
4. Select the line that you just added, click **Calculate Prices**, then modify charges, as necessary. For example, modify the Sale Price charge, add a recurring charge, an activation fee, and so on.

Add Price List to Pricing Strategy

Add your price list to Corporate Pricing Strategy. See *Manage Pricing Strategies*.

More

See these topics to get more detail about how you can manage your price list:

- *Redwood: Manage Price Lists*
- *Redwood: Set Up Approval Flows for Price Lists*

- *Redwood: Apply Pricing Formulas with Mass Actions on Price Lists*
- *Redwood: Use More Entities on Price Lists for Approvals*
- *Redwood: Do More Mass Actions on Price Lists*
- *Import or Update a Large Number of Components for Price Lists*

Related Topics

- Pricing Rules
- Manage Pricing Charge Definitions
- Cost Plus Pricing
- How Profiles, Segments, and Strategies Work Together
- Select a Technology to Manage Your Pricing Data

Display Only Recurring Charges from Price Lists

Some items include a charge that recurs regularly, such as a subscription that recurs one time every month. The charge for other items, such as a desktop computer, don't recur, but instead include a one-time sale price.

An order line might display the recurring charge and the sale price. This order line has a subscription with a 30.00 recurring charge:

Charge	Value
Recurring Sale Price (Month)	30.00
Amount	0.00

You probably don't want the sale price to display for a subscription. You can set up Pricing to remove the Sale Price from the order line:

1. Create a price list that includes the subscription, and add the recurring charge to the subscription.

   For this example, name the price list My_Price_List. Use the All Items tab instead of the Items tab when you add the charge to My_Price_List.
2. Add My_Price_List to your pricing strategy.
3. On the Edit Pricing Strategy page, click the **row** that includes My_Price_List, click **Top of List**, then verify that the Precedence attribute is 1 for My_Price_List.
4. At run time, Order Management will use the price list with the highest precedence and ignore the others.

Related Topics

- Manage Pricing Totals
- Overview of Oracle Pricing

Manage Tax in Oracle Pricing

Manage how Oracle Pricing applies tax when it prices an item.

Include or Exclude Tax on List Price

Specify whether to include or exclude tax on the list price:

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Tax Regimes**.
2. On the Manage Tax Regimes page, open your tax regime for editing.
3. On the Edit Tax Regime page, set the inclusion method:

Tax Inclusion Method	Description
Standard Inclusive Handling	Include tax as part of the list price.
Standard Noninclusive Handling	Don't include tax as part of the list price. Instead, add it to the list price.

Assume the list price for the item is 100:

Tax	Price in Sales Order
Inclusive tax of 10%.	Total Net Price, $90 Total Tax, $10 Pay Now, $100
Exclusive tax of 10%.	Total Net Price, $100 Total Tax, $10 Pay Now, $110

Use Amounts in the Transaction Currency to Calculate Tax

You can use Oracle Pricing to calculate tax so it more closely matches the tax calculations in Oracle Receivables. Control whether to use amounts in the transaction currency or the charge currency when you calculate tax for each transaction.

Pricing uses the charge currency to calculate tax, by default. If you want to use the transaction currency instead, then:

1. Go to the Setup and Maintenance work area, then select the Order Management offering.
2. On the Opt In page, click the **pencil** in the Pricing row.
3. Click Change Feature Opt In.

4. Enable the Use Amounts in the Transaction Currency to Calculate Tax feature.

If you use more than one currency in your transactions, then we recommend that you create different series of test data to determine whether the transaction currency or the charge currency provides the optimal results for your tax calculations.

Use Extended Precision

See *Use Extended Precision to Round Unit Tax*.

Related Topics

- Example Pricing Algorithm, Part 7, Calculate Tax
- Inclusive Taxes
- Considerations for Configuring Inclusive Taxes

Costs

Manage Cost Lists

Modify a predefined cost list or create your own to set up a variety of charges, such as item cost, sales commission, or labor cost.

- Use a cost list to set up cost plus pricing.
- Use a cost list to calculate part of the profit margin for a charge.
- Separate charges to help you manage and optimize charges and profit margins for an item, and help improve your pricing strategy. For example, create a separate charge for the transport charge, installation charge, recurring service charge for maintenance, and a one-time administration fee for 50 desktop computers. Pricing adds each cost as a charge to the cost list.
- Manage charges at different levels, such as individual items or all items.
- Get item costs from Oracle Cost Management in real time.

For details, see *Redwood: Manage Cost Lists*.

Related Topics

- Pricing Rules
- Manage Pricing Charge Definitions
- Cost Plus Pricing
- Manage Pricing Strategies

Cost Plus Pricing

Use cost plus pricing to calculate and analyze the profit margin that your company earns for an item in terms of the pricing charges that the item references.

Use it to optimize pricing so it meets the pricing objective you define.

If you use cost plus pricing, then Oracle Pricing calculates the item price according to attributes you set on the price list and the cost list. The cost of an item is the sum of the charges that you specify for the item on these lists. Pricing includes only the charges you enable for cost plus pricing as part of the cost when it calculates price.

Here's an example that uses cost plus pricing:

Cost Plus Pricing	Cost Amount	Cost Calculation Type	Markup	Selling Price
Contains a check mark	345	Fixed	55	345 plus 55 equals $400
Does not contain a check mark	345	Fixed	55	345

Assume you typically sell a cellular phone according to the price you specify in the Base Price attribute of the price list of $445, and you set the Calculation Method in the price list to Price. You offer the phone to your customer for $400 using cost plus pricing. Here's your setup:

1. In the price list, set these attributes:

Attribute	Value
Calculation Method	Cost
Calculation Type	Markup Amount
Cost Calculation Amount	55

2. In the cost list, set these attributes:

Attribute	Value
Cost Calculation Type	Fixed
Cost Amount	345
Cost Plus Pricing	Add a check mark

Here's the calculation that Pricing will do at run time:

- Cost of $345 plus markup of $55 equals a base price of $400

Set Frequency to a Granular Level

Set Frequency to a sufficiently granular value so it meets your customer's requirements.

Assume you add a cost list in the Pricing Administration work area. It includes a $420 annual fee for maintenance that you provide as a service on the AS54888 Desktop Computer. You click Create Cost Charge, then set the values:

Attribute	Value
Pricing Charge Definition	Recurring
Frequency	Year
Cost Amount	420

Your customer, Computer Service and Rentals, places an order for the AS54888 and requests maintenance for 1.5 years. You create a sales order in the Order Management work area, add the AS54888 Desktop Computer to the order, set the Duration to 1.5, the Period to Year, the Contract Start Date to January 1, 2020, then submit. However, Order Management can't use a fraction such as 1.5 to calculate the Contract End Date. In can use only a whole number. So Order Management displays an error requesting you to change duration to a whole number.

To avoid this problem, and to meet Computer Service and Rentals' request, set Frequency to a more granular value:

Attribute	Value
Pricing Charge Definition	Recurring
Frequency	Month
Cost Amount	35

Then set Duration on the order line to 18.

If you must keep Frequency at Year, then leave Duration empty and set the Contract End Date to June 30, 2021 when you create the order line. However, this approach requires you to manually calculate the end date every time you add an order line because the date changes, and you must calculate it accurately to the day. Avoid this problem. Make the Frequency more granular instead.

Related Topics

- Pricing Rules
- Manage Pricing Charge Definitions
- Cost Plus Pricing
- Manage Pricing Strategies

Discounts

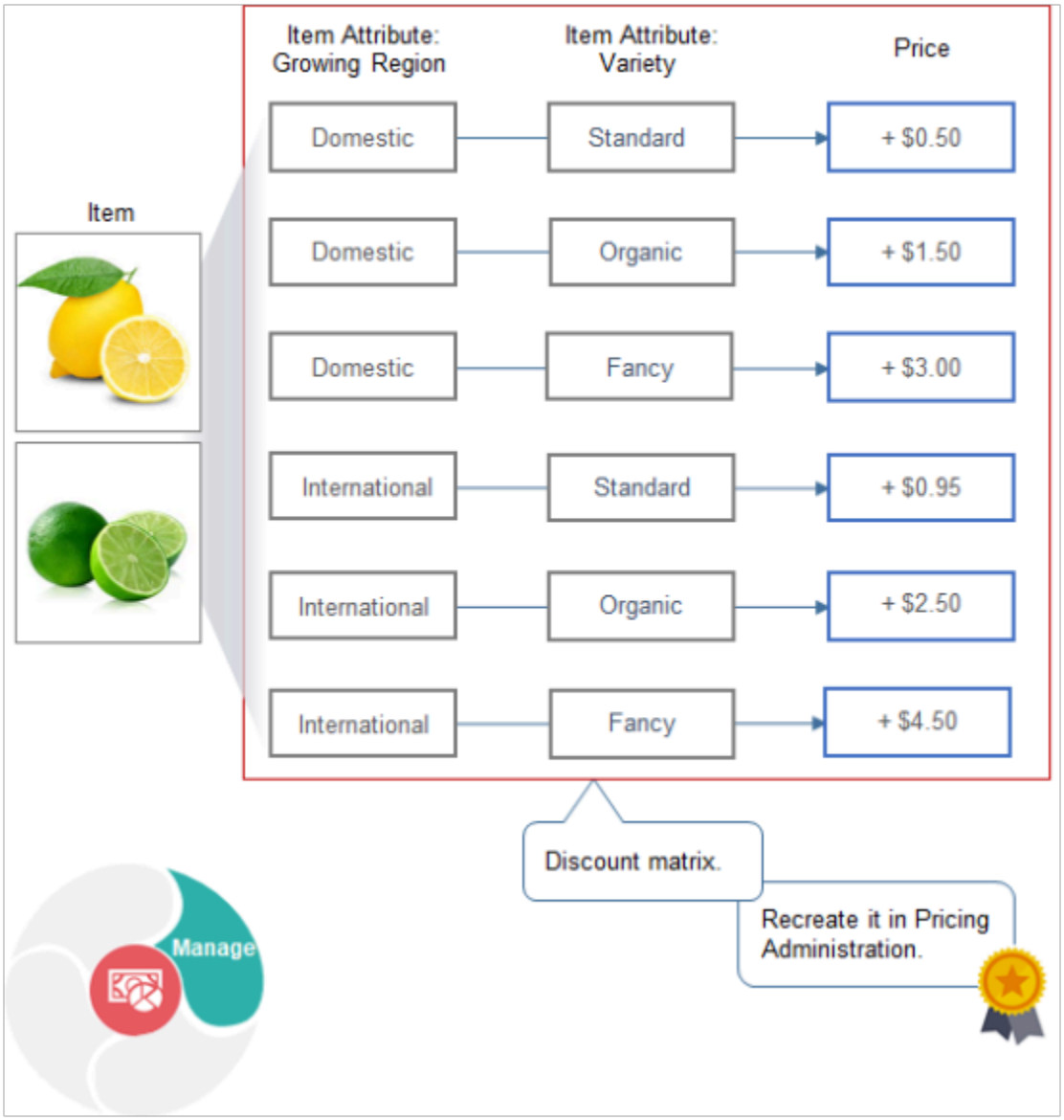
Discount Lists

Create a discount list to set up discounts, profit margins, and price overrides for an item.

- Set up a discount for a variety of reasons, such as a seasonal holiday discount, or volume discount according to price or quantity.
- Apply a flat rate discount or a percent discount.
- Use a pricing matrix to set up a discount according to an item attribute and apply it to the list price at run time. For example, mark up the price of an extra large, red shirt by $10, and apply a 7.5% discount for a blue, large shirt.

Discount Matrix

Use a discount matrix to manage discounts:



Note

- Use dynamic, declarative rules to set up conditions and results that determine price. For example, use the Growing Region attribute and the Variety attribute of the Lemon item as the condition, and the adjustment as the result.

  ```
  If Growing Region is Domestic, and if Variety is Standard, then adjust price by +$0.50.
  ```

- Create a price list matrix or discount matrix that adjusts base price or list price.

- Create rules according to:
    - **Transactional attribute**. An attribute that happens on the sales order.
    - **Item extensible attribute**. An attribute that you specify in the Product Management work area to meet your specific business requirements. For example, Growing Region could be a transactional attribute and Variety could be an item extensible attribute.
    - Define matrix rules that apply to all items on the order line. Or set up matrix rules that apply only to some items on the order line according to the value of an attribute.
    - .For example, if an order line includes 10 domestic standard lemons, and 10 domestic organic lemons, then add $0.50 to the standard lemons, and add $1.50 to the organic lemons.
    - Use a matrix class to create a template that you can reuse with different pricing entities, such as pricing charge, discount, and so on.

For details, see:

Redwood: Manage Discount Lists

Redwood: Manage Discount Rules for Component Items

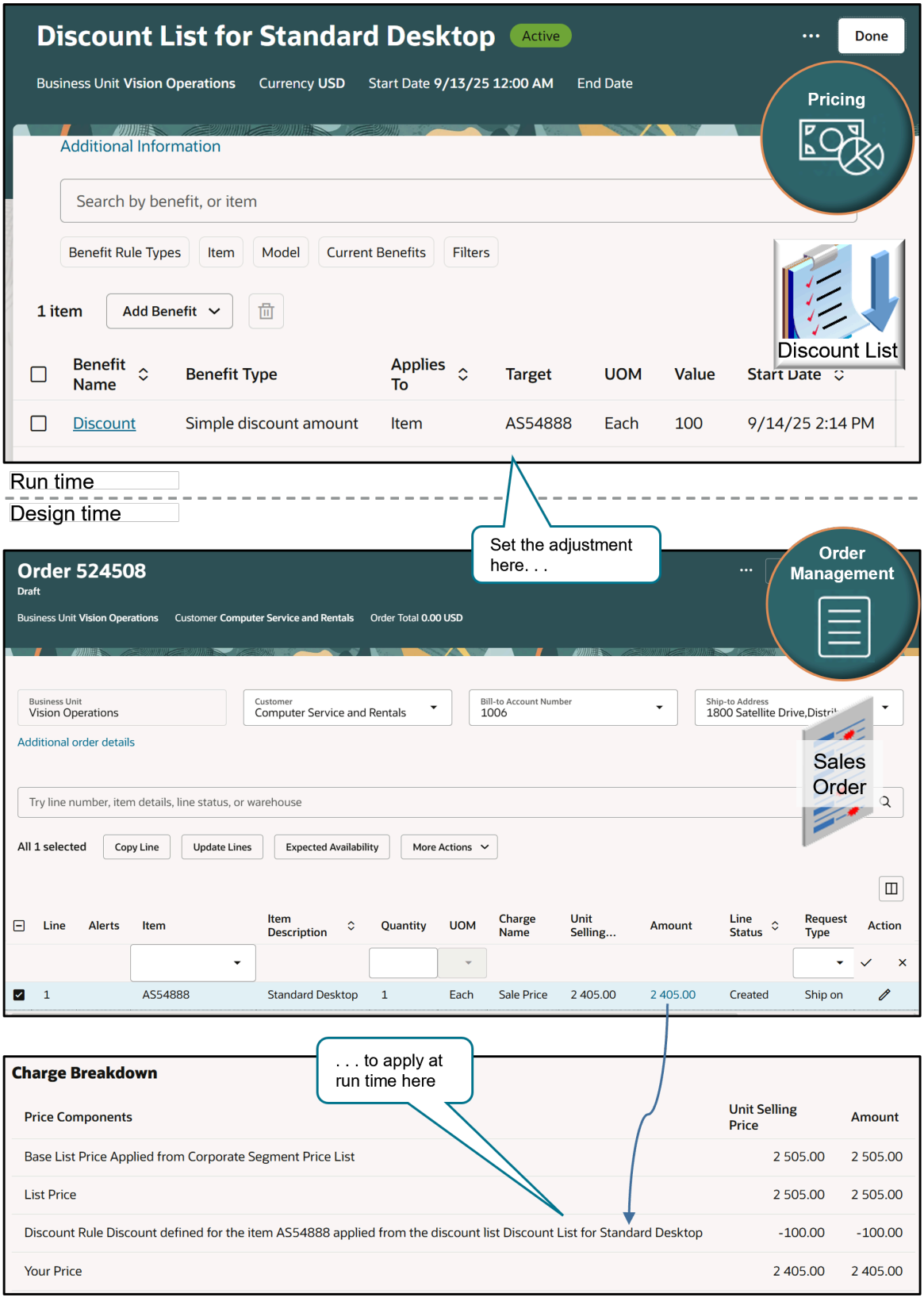
Related Topics

- Manage Cost Lists
- Manage Pricing Strategies
- Pricing Rules
- Tier Pricing

Manage Discount Lists

Create a pricing rule that provides a one-time, $100 discount on the sale price of a new computer.

You use the on the discount list in the Pricing Administration work area at design time to calculate a discount on the New Sales Order page in the Order Management work area at run time:


Discount List for Standard Desktop
Active
Done
Business Unit Vision Operations
Currency USD
Start Date 9/13/25 12:00 AM
End Date
Pricing
Additional Information
Search by benefit, or item
Benefit Rule Types
Item
Model
Current Benefits
Filters
1 item
Add Benefit
Discount List
Benefit Name
Benefit Type
Applies To
Target
UOM
Value
Start Date
Discount
Simple discount amount
Item
AS54888
Each
100
9/14/25 2:14 PM
Run time
Design time
Set the adjustment here. . .
Order 524508
Draft
Business Unit Vision Operations
Customer Computer Service and Rentals
Order Total 0.00 USD
Order Management
Business Unit
Vision Operations
Customer
Computer Service and Rentals
Bill-to Account Number
1006
Ship-to Address
1800 Satellite Drive,Distri
Additional order details
Sales Order
Try line number, item details, line status, or warehouse
All 1 selected
Copy Line
Update Lines
Expected Availability
More Actions
Line
Alerts
Item
Item Description
Quantity
UOM
Charge Name
Unit Selling...
Amount
Line Status
Request Type
Action
1
AS54888
Standard Desktop
1
Each
Sale Price
2 405.00
2 405.00
Created
Ship on
. . . to apply at run time here
Charge Breakdown
Price Components
Unit Selling Price
Amount
Base List Price Applied from Corporate Segment Price List
2 505.00
2 505.00
List Price
2 505.00
2 505.00
Discount Rule Discount defined for the item AS54888 applied from the discount list Discount List for Standard Desktop
-100.00
-100.00
Your Price
2 405.00
2 405.00

Assume you're rolling out a sales campaign to bring in the new year and increase sales for the AS54888 desktop computer. You're offering a $100 discount. This topic shows you how to do it.

Summary of the Steps

1. Create discount list.
2. Add item to discount list and strategy.
3. Test your setup.

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

For details, see *Pricing Rules*.

Create Discount List

1. Go to **Home > Order Management > Pricing Administration > Discount Lists**.
2. Create a new discount list:

Attribute	Value
Name	Discount List for Standard Desktop
Business Unit	Vision Operations
Currency	USD
Start Date	Any time before the current date.
End Date	Leave empty.

Add Item to Discount List and Strategy

1. Click **Add Benefit > Simple Rule**, then set the values:

Attribute	Value
Name	Discount
Benefit Type	Simple Discount Amount
Value	100
Price Type	One Time
Applies To	Item

Attribute	Value
Item	AS54888
Standalone Item	Selected

2. Click **Additional Information > Access Sets**, then add this set:

Attribute	Value
Set Code	Business unit of the selling organization. For this example, set it to Vision Operations. You can also use the Common set.

3. Click **Activate**.
4. Add Discount List for Standard Desktop to the Corporate Pricing Strategy. For details, see *Manage Pricing Strategies*.

Test Your Setup

-Open another browser application, then sign into Oracle Applications with the privileges that you need to manage sales orders:

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

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations
Order Type	Standard Orders
Bill-to Account	1006

2. On the order line, search for AS54888.
3. Wait for the result to display, then click the sale price.
4. Verify that the charge breakdown includes your new $100 discount.

```
Discount Rule Discount defined for the item AS54888 applied from the discount list Discount List for
Standard Desktop
```

Related Topics

- Manage Cost Lists
- Manage Pricing Strategies
- Pricing Rules
- Tier Pricing

Apply Discount According to Time

Learn how you can apply discount according to time.

Apply Discount for a Finite Time Period

For example, create a price list that sets the base price of a cell phone at $445 with no other adjustments, which sets the list price at $445. You can set up a seasonal discount of 10% from December 10 to January 1, resulting in a net price of $400.50:

1. Create your discount list and set the attributes:

Attribute	Value
Name	Happy Holidays
Start Date	12/10/20 12:00 AM
End Date	1/1/21 11:59 PM

2. Add the Happy Holidays discount list to your pricing strategy.

   Pricing will apply the discount from the Start Date to the End Date.

Apply Two Discounts

What if I already discount my item by 10%, but want to add another 5% for this weekend only, which starts on 02/06/21, and then revert to 10% next week?

1. Create your normal discount list and set the attributes:

Attribute	Value
Name	Normal Discount
Start Date	Anytime before today

Attribute	Value
End Date	Leave empty

2. Create your weekend discount list and set the attributes:

Attribute	Value
Name	Weekend Happy Daze
Start Date	02/06/21 12:00 AM
End Date	02/07/21 11:59 PM

3. Add the discount lists to your pricing strategy:

Name	Precedence
Weekend Happy Daze	1
Normal Discount	2

Weekend Happy Daze has the highest precedence, so Pricing will apply it but only while Weekend Happy Daze is active. Pricing will apply Normal Discount before 02/06/21 12:00 AM and after 02/07/21 11:59 PM.

Related Topics

- Manage Cost Lists
- Manage Pricing Strategies
- Pricing Rules
- Tier Pricing

Apply Price According to Minimum and Maximum Quantity

You can specify the minimum and maximum quantities that you want to use when calculating the price on an order line.

Assume you need to give a different discount for the AS54888 item according to the customer that you're selling to and the quantity on the order line.

Try it.

1. Go to the Pricing Administration work area, then click **Manage Matrix Classes**.

2. On the Manage Matrix Classes page, in the Name column, click **Pricing Term Adjustment**.
3. On the Edit Matrix Class page, in the Condition Columns area, set the values:

Name	Source Code Name	Comparison	Compare to Attribute	Required	Allow Null
MinQuantity	MinQuantity	<	Line.LineQuantity	Contains a check mark.	Doesn't contain a check mark.
MaxQuantity	MaxQuantity	>=	Line.LineQuantity	Contains a check mark.	Doesn't contain a check mark.

Note

Code	Description
`Line`	Specify the order line.
`LineQuantity`	Specify to examine the runtime value of the Quantity attribute on the order line.
`MinQuantity < Line.LineQuantity`	Specify to compare the MinQuantity that you set on the discount rule to the quantity on the order line. If MinQuantity on the rule is less than the quantity on the order line, then the rule evaluates to true, and Pricing applies the discount.
`MaxQuantity >= Line.LineQuantity`	Specify to compare the MaxQuantity that you set on the discount rule to the quantity on the order line. If MaxQuantity is equal to or greater than the quantity on the order line, then the rule evaluates to true, and Pricing applies the discount.
Make sure you set Compare to Attribute to `Line.LineQuantity`	You must use this value when you apply a pricing rule according to the quantity on the order line. Don't use any other value.
`Domain`	Don't modify the Domain attribute.

4. Click **Manage Discount Lists**, then search for and open Discount List for Standard Desktop. For this example, assume you already created this list and added the AS54888 item to the list. For details, see *Manage Discount Lists*.
5. In the Discount Rules area, click **Actions > Create > Attribute Based Rule**.
6. In the dialog that displays, select MinQuantity and MaxQuantity, then click **Finish**.
7. In the dialog that displays, set the values, then click **OK**:

Customer	MinQuantity	MaxQuantity	Adjustment Type	Adjustment Amount
Computer Service and Rentals	0	5	Discount Percent	5

Customer	MinQuantity	MaxQuantity	Adjustment Type	Adjustment Amount
Vision Operations	0	20	Discount Percent	10

Assume your user sets the customer to Computer Service and Rentals and the quantity to 2 on the order line at run time:

Design Time on Matrix Class	Design Time on Rule	Runtime Example
MinQuantity is less than Line.LineQuantity.	MinQuantity equals 0.	LineQuantity equals 2, so the rule evaluates to true because 0 is less than 2.
MaxQuantity is greater than or equal to Line.LineQuantity.	MaxQuantity equals 5.	LineQuantity equals 2, so the rule evaluates to true because 5 is greater than or equal to 2.

8. Test your set up. Go to the Order Management work area, create a sales order, and verify:
    - You give Computer Service and Rentals a 5% discount only when they order a quantity of 5 or less.
    - You give Vision Operations a 10% discount only when they order a quantity of 20 or more.

Manage Pricing Promotions

Set up a simple or tiered discount for one or more items or categories, or for all items in a single promotion.

You can create a pricing promotion as an incentive to motivate your customer to buy your item. For example, to buy more quantity or a type of item. Sell items and services together to provide an incentive when your customer buys one or more item to get a discount or rebate:

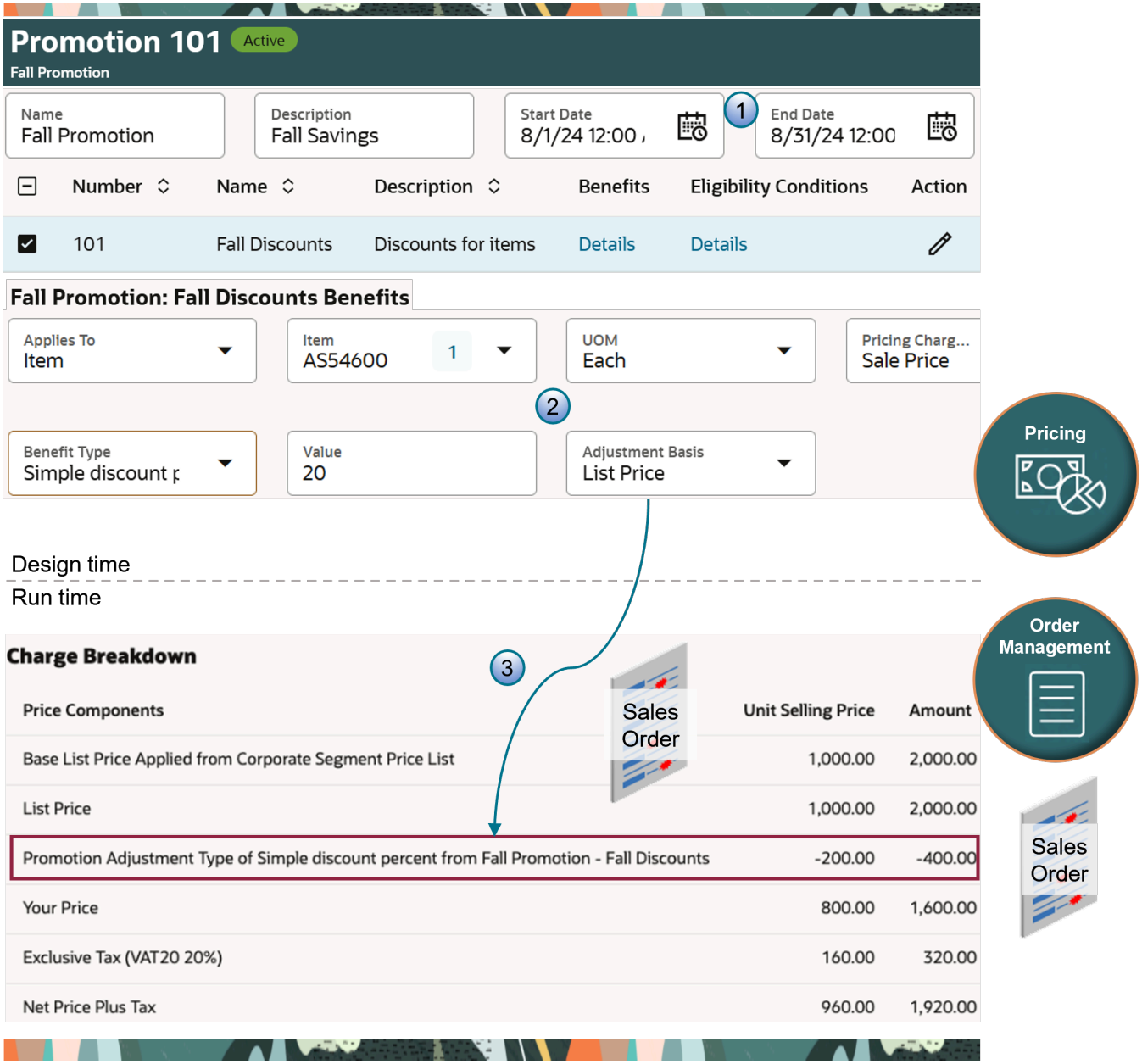
- Set up a rule that specifies a condition. The condition determines whether the sales order is eligible or not eligible for the promotion.
- Use a friendly rule builder to set up a more complex eligibility condition for a promotion according to an order line, customer, or the item's attributes.
- Pricing will automatically apply the pricing promotion to your sales order independent of your pricing strategy.

Create a Simple Promotion

Assume you sell supplies and equipment to students of all ages. You have a fall promotion that you want to run during the entire month of August:

```
Apply a 20% discount off the list price of each AS54600 item that my customer buys on August 1, 2024 through
August 31, 2024.
```

Here's your set up:



Try It:

1. Go to the Pricing Administration work area, then click **Pricing Promotions**.
2. On the Pricing Promotions page, click **Create Promotion**, then set the values:

Attribute	Value
Number	101 Note o You can enter any alphanumeric value.

Attribute	Value
	○ The value must be unique across promotions. ○ If you don't provide a value, then Pricing will automatically assign one after you save your work. ○ You can't change the number after you save. ○ The same behavior applies for a promotion line's number.
Name	Fall Promotion
Description	Fall Savings
Start Date	August 1, 2024 Use the promotion's start date and end date to determine when you want the promotion to be in effect.
End Date	August 31, 2024

3. Click **Add Row** to add a promotion line, set the values, then click **Apply**:

Attribute	Value
Promotion Line Number	101
Promotion Line Name	Fall Discounts
Promotion Line Description	Discounts for items

4. A *benefit* is an *incentive* that provides a reason to motivate your customer to purchase an item. Click **Add** in the Benefits column, then set the values:

Attribute	Value
Applies To	Item
Item	AS54600
UOM	Each
Pricing Charge Definition	Sale Price Use this attribute to identify the charge where you want to apply the promotion. Assume you select Sale Price. If the item on the order line has a sale price charge, then Pricing will apply the promotion on that charge. For this example, you select only Sale Price, so Pricing will apply the promotion only on the sale price.

Attribute	Value
Benefit Type	Simple Discount Percent
Value	20
Adjustment Basis	List Price

5. Click **Activate** to make the promotion available so Pricing can use it to price your sales order. Use this button to help manage your set up. For example, you might need to add a number of lines and conditions before you want Pricing to use this promotion.

   You can't deactivate or delete a promotion. If you no longer want Pricing to use it, then set the End Date to a date that has already occurred.

If the sales order meets the condition, then Pricing will apply the rule at run time on the sales order in Order Management. The charge breakdown on the order line will include the promotion name, the promotion line's name, the type of adjustment, and the adjustment amount:

```
Promotion Adjustment Type of Simple Discount Percent from Fall Promotion – Fall Discounts
```

Promote an Entire Category of Items

Use a category to focus where you want to apply a promotion instead of applying the promotion on each individual item:

- Apply your promotion to one or more categories in a catalog.
- Evaluate and apply your rule to each item that's in the category's hierarchy.
- Use the category as the eligibility criteria to include or exclude a promotion.

As part of the fall promotion, assume you want to offer a 10% discount on any pencil and any stationery that you sell. Here's your rule:

Attribute	Value
Applies To	Item Category
Category	Color Pencils School Stationary
UOM	Each
Pricing Charge Definition	Sale Price
Benefit Type	Simple Discount Percent
Value	10
Adjustment Basis	List Price

If you add more than one benefit on the same promotion line, then Pricing will apply each benefit to any item that meets each rule. Assume the line has two benefits that specify these rules:

- If the item is the AS54600, then apply a 20% discount off the AS54600's list price.
- If the item is in the color pencil category or the school stationery category, then apply a 10% discount off the list price. For example, if the list price is $100, then apply a 10% discount on $100, which is $10.

Note

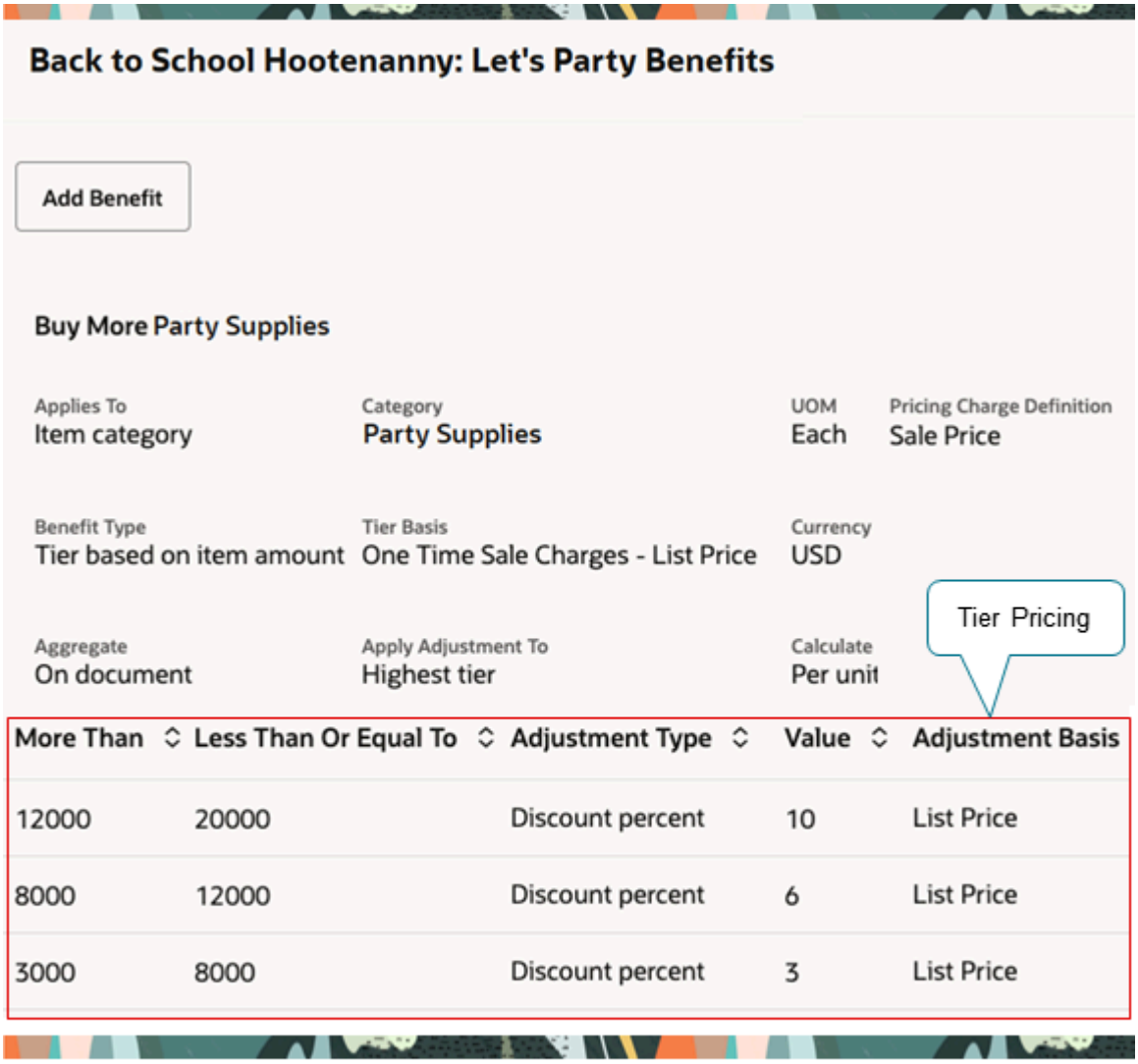
- If you want to apply a promotion according to a category, then you must assign the catalog that contains that category as the primary catalog. For more, see the Primary Catalog parameter and the Product Catalogs parameter in *Manage Pricing Parameters*.
- The rules aren't mutually exclusive. Assume you create an order line that has the AS54600, and then another line in the same sales order that has a pencil. Pricing will apply a 20% on the AS54600 and a 10% discount on the pencil.
- Pricing applies each promotion that meets the condition even if these promotions are for the same item. Assume the AS54600 is in the computers category, you create one rule that applies a 20% discount on the AS54600, and then another rule that applies a 10% discount on the Computers category. You create an order line and add the AS54600 to the line. Pricing will apply a net 30% discount on the order line: 20% for the item and 10% for the category.

For background about the pricing charge definition, see *Manage Pricing Charge Definitions*.

For background about the adjustment basis, see *Manage Pricing Bases*.

Tier Your Adjustments

You can add a tiered discount according to the list price and the category that the item is in:



If the total amount of one time charges for all party supplies on the sales order is more than:

- $12,000 but less than $20,000, then apply a 10% discount off the list price.
- $8,000 but less than $12,000, then apply a 6% discount off the list price.
- $3,000 but less than $8,000, then apply a 3% discount off the list price.

Tell Pricing how to calculate the rule:

Attribute	Value
Calculate	You can set it to one of:

Attribute	Value
	• Lump Sum. Apply the adjustment to all quantities. This is similar to the Block option in tier pricing. • Set the Remainder Quantities in the New Lump Sum attribute to Yes or No. This is similar to the Partial Block option in tier pricing. • Per Unit. Apply the rule to each unit. For example, if the order line's quantity is 4, and if the discount is $1 for each item, then the total discount is $4. For this example, set it to Per Unit.

For background about tier pricing and how to set some of these attributes, see *Tier Pricing* and *Add Tiers to Pricing Rules*.

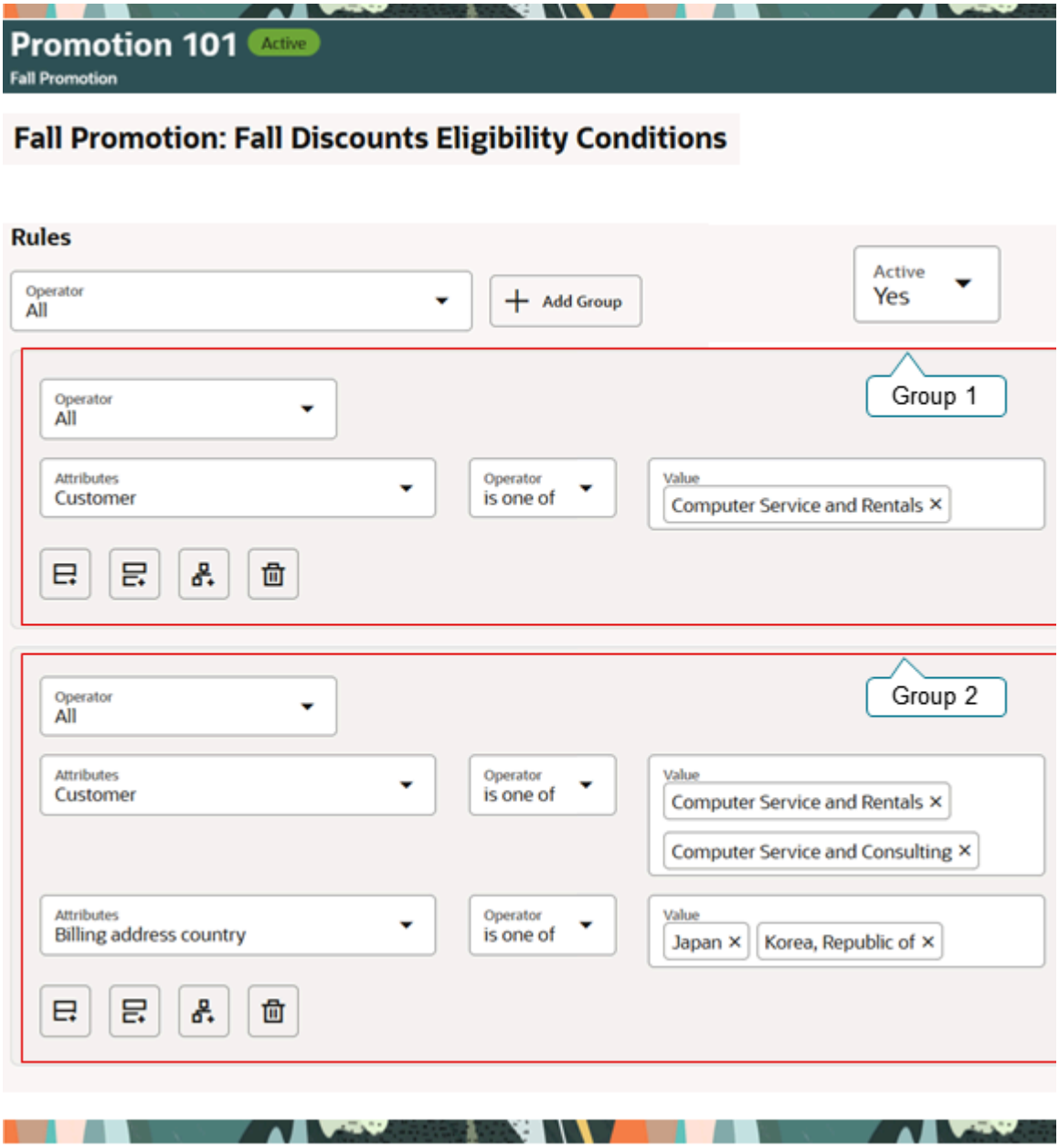
Use Eligibility Conditions to Focus Your Promotion

Assume this rule is too broad for your needs:

```
Apply a 20% discount off the list price of each AS54600 item that my customers buy on August 1, 2024 through August 31, 2024
```

You can use a set of predefined customer attributes, order line attributes, and item attributes to focus your rule according to those attribute's values.

Consider this example:



Adding a group adds OR logic to your eligibility conditions. The Rules section in this example has two groups:

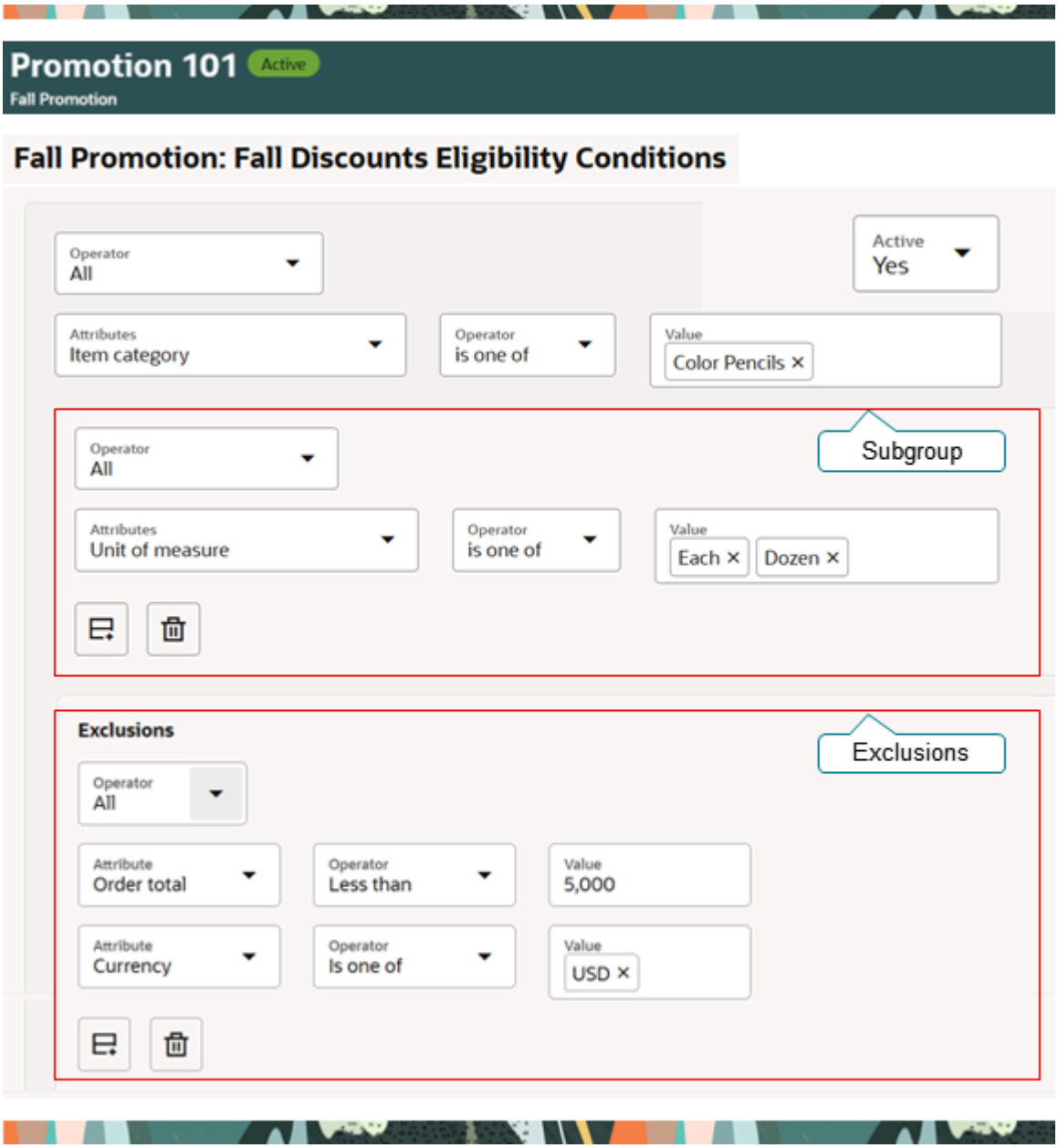
Group	Rule
1	If the Customer attribute contains Computer Service and Rentals, then apply the promotion.
2	Or if the Customer attribute contains Computer Service and Rentals, or if it contains Computer Service and Consulting, and if the billing address is in Japan or Korea, then apply the promotion.

Note

- If you reference an attribute that might have more than one value, such as Billing Address City, then include another condition that examines a different attribute, such as Billing Address State or Billing Address Country. This will help to further qualify the promotion.
- The Eligibility Conditions dialog displays the attribute values that you can set.

Group and Exclude

Use subgroups and exclusions to focus your promotion even more. A subgroup works like adding an AND operator to the group, while an exclusion works like adding a NOT operator to the group. Consider this example:



The category specifies Color Pencils as the group, an order line attribute specifies the subgroup, and we also have an exclusion. The example applies this rule:

Condition	Rule
Group	If the item is in the Color Pencils category. . .
Subgroup	. . . And if the order line's Unit of Measure attribute contains Each or it contains Dozen. . .
Exclusion	. . . And if the total amount on the sales order isn't less than 5,000 and the currency is USD, then apply the promotion.

Note

- If you specify a condition according to Line Amount or Order Total, then include the currency as a separate condition.
- If you specify a condition according to Line Quantity, then include the UOM as a separate condition.
- Some customer attributes might have more than one value, such as Billing Address City. If you reference an attribute that might have more than one value, then include another condition, such as Billing Address State or Billing Address Country. This will help to further qualify the promotion.
- You can use the Is One Of operator to specify more than one value in a condition. This example's subgroup uses the Is One Of operator to apply the condition when the attribute's value is Each or Dozen.

Apply Strategies and Payment Terms

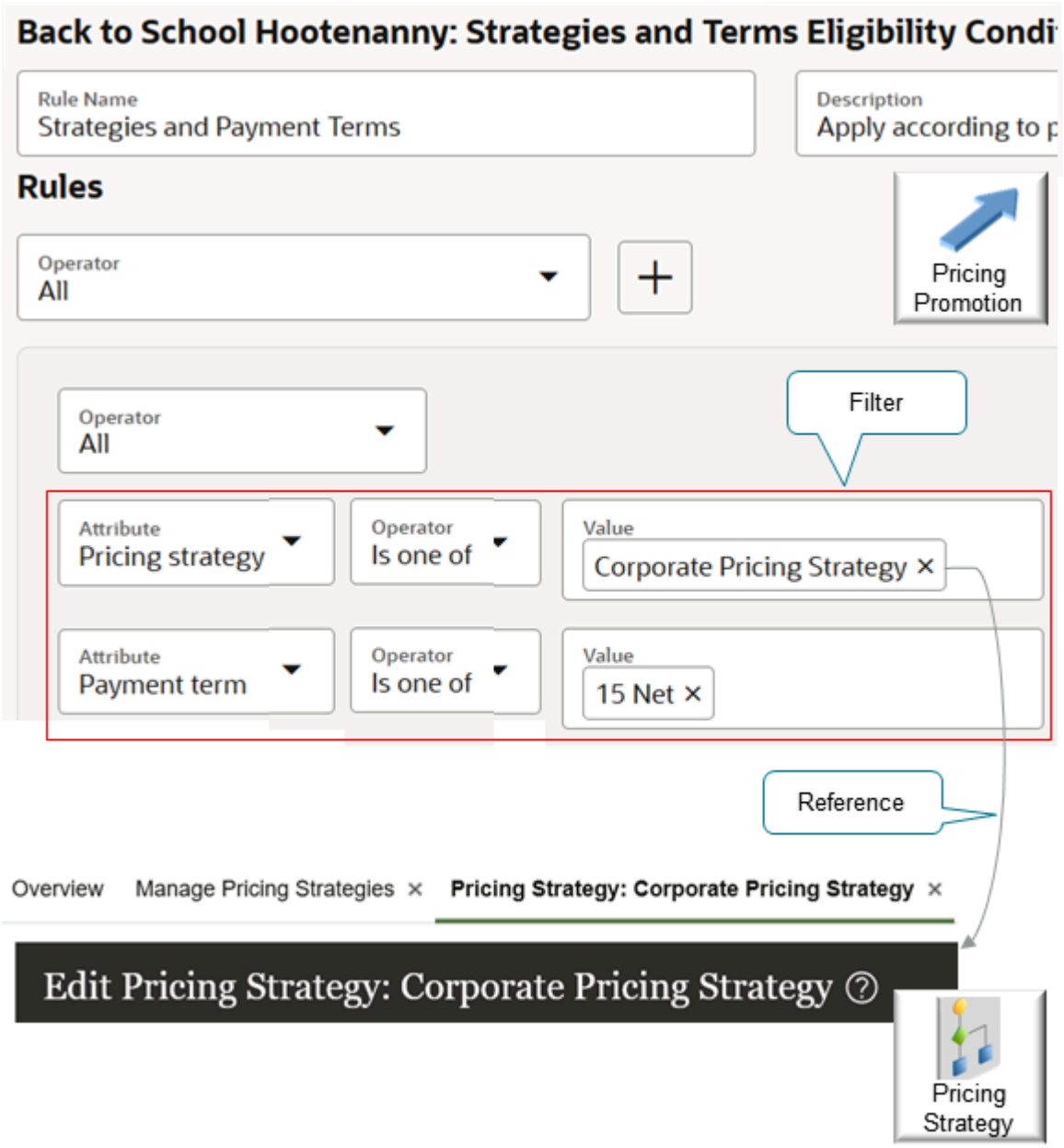
Automatically match a promotion to your pricing strategy and payment term.

Create an eligibility rule that uses a pricing strategy or payment term to determine whether the order is eligible for a pricing promotion. Use the pricing strategy to automatically match the promotion to your pricing objective. As an alternative, add a payment term to your eligibility rule so you can reward each customer who agrees to the term that meets your strategy.

Try it:

1. Create your promotion and promotion line, then add a benefit.
2. Add an eligibility rule to the benefit and use it to specify which sales orders are eligible for promotion.

For example, filter eligibility where the pricing strategy is Corporate Pricing Strategy and the payment term is 15 NET:



Note

- You can add a pricing strategy to an eligibility rule even if you haven't approved the strategy. This way you can set up and test your strategies and rules before you assign the strategy.
- Select your payment term carefully. Payment terms use a variety of formats. For example, 2/10 Net 30 is a payment term, and so is 2/10, Net 30. If you need the one that has the comma, then search for 2/10, Net 30. You might also see duplicate values for the same term. To avoid this problem, we recommend that you use unique values in the Name attribute when you use the Manage Payment Terms task in the Setup and Maintenance work area to create your payment terms.

For more, see:

- *Pricing Strategies*

- *Guidelines for Defining Payment Terms*

Use Closed Lines to Help Price Items

Allow Oracle Pricing to consider closed lines when it needs to price an item that involves a pricing promotion. See *Use Closed Lines to Help Price Items*.

Follow Guidelines

- You can have more than one promotion line on the same promotion. For example, set up different eligibility conditions for different items. Enable and disable eligibility to meet your requirements as they change over time.
- If you add more than one promotion line to a promotion, then Pricing applies them in the same sequence that you add them. For example, if you add lines x, y, and z, then Pricing applies line x first, then line y, and finally line z at run time.
- Your eligibility conditions apply to all of the rules that you set up on the line. Assume you have a 20% discount and a 30% discount on the line. Pricing will apply the eligibility conditions to the 20% discount and to the 30% discount. If you need to apply eligibility for the discounts differently, then create a new promotion line and apply the 30% discount and your different eligibility on the new line.
- You can have a subgroup in a group, but you can't have a subgroup in a subgroup.
- You can add an exclusion to a group, but not to a subgroup.

Use the Operator attribute to refine how you determine whether the sales order is eligible for the promotion. You can set the Operator on different levels:

Level	Value
Rule	• **Any**. If any group is eligible, then apply the promotion. • **All**. Apply the promotion only if all groups are eligible.
Group	• **Any**. If any condition in the group is eligible, then apply the promotion. • **All**. Apply the promotion only if all conditions in the group are eligible.
Subgroup	• **Any**. If any condition in the subgroup is eligible, then apply the promotion. • **All**. Apply the promotion only if all conditions in the subgroup are eligible.
Exclusion	• **Any**. If any condition in the exclusion is eligible, then apply the promotion. • **All**. Apply the promotion only if all conditions in the exclusion are eligible.

You can use these attributes in your rule:

Customer Attributes	Sales Order Attributes	Item Attributes from Product Management
• Customer • Customer Account • Billing Address City • Billing Address County	• Currency • Line Quantity • Line Amount • Order Total	• Item • Item Category • UOM

Customer Attributes	Sales Order Attributes	Item Attributes from Product Management
• Billing Address State • Billing Address Province • Billing Address Country • Billing Address Postal Code • Shipping Address City • Shipping Address County • Shipping Address State • Shipping Address Province • Shipping Address Country • Shipping Address Postal Code	• Business Unit • Shipment Method Pricing uses the list price for a one time charge in the: • Order line's currency to calculate the Line Amount. • Order header's currency to calculate the Order Total. Pricing considers the currency only on outbound lines. It doesn't consider the currency on return lines.	

Set Up

1. Make sure you have these privileges:
    - Manage Draft Pricing Promotions (QP_MANAGE_DRAFT_PRICING_PROMOTIONS)
    - Manage Pricing Promotions (QP_MANAGE_PRICING_PROMOTIONS)
    - View Pricing Promotions (QP_VIEW_PRICING_PROMOTIONS)
    - Activate Pricing Promotions (QP_ APPROVE_PRICING_PROMOTIONS)
    - Manage Dynamic Rules (QP_MANAGE_DYNAMIC_RULES)
2. Use the Manage Price Elements page to make sure the Active attribute is enabled for these price elements:
    - **Adjustment for Promotion**. For simple discounts.
    - **Tiered Adjustment for Promotion**. For tiered adjustments.

    For details, see *Manage Pricing Elements*.

More

See these topics to get more detail about how you can manage your promotions:

- *AI Agent: Promotion Recommendation Advisor*
- *Redwood: Use AI Services with Your Pricing Promotions*
- *Redwood: Use Generative AI to Create Your Pricing Promotion's Description*
- *Redwood: Apply Promotions According to Items on Sales Orders*
- *Redwood: Use Promotion Codes to Apply Promotions On Order Lines*
- *Redwood: Create Promotions That Add Free Items*

Shipping Charges

Manage Shipping Charge Lists

Create a shipping charge list and use it to calculate an item's freight, duty, handling, and insurance charges.

For example, select 2nd day air delivery for a desktop computer at a price of $75 for each computer, or select ground transportation at a price of $25.

Assume you have need to price shipping for the AS54888 Desktop Computer at $50 USD for Standard Delivery and $100 USD for Express Delivery.

Also, if your customer orders a quantity of:

- **5 to 10 computers.** Apply a 10% discount on the shipping charge.
- **More than 10 computers.** Apply a 25% discount on the shipping charge.

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

Summary of the Steps

1. Create shipping charge list.
2. Create pricing matrix.
3. Add shipping charge list.

Create Shipping Charge List

1. Go to **Home > Order Management > Pricing Administration > Manage Shipping Charge Lists**.
2. On page Manage Shipping Charge Lists, in the search results, click **Actions > Create**.
3. In the Create Shipping Charge List dialog, set the values for the Name attribute and the other required attributes, then click **Save and Edit**:

Attribute	Value
Name	Shipping Charge List for Standard Desktop

4. On the Edit Shipping Charge List page, on the Shipping Charges tab, click **Items**.
   You add a shipping charge for a single item. Pricing doesn't support the Flat Rates tab in the current update.
5. Click **Actions > Create > Create Item Charge**.
6. In the Create Item Charge dialog, set the values:

Attribute	Value
Shipping Method	Standard Delivery
Pricing Charge Definition	Freight

Attribute	Value

7. Click **Actions > Add Row**, then set the values:

Attribute	Value
Name	AS54888
Calculation Method	Price
Base Price	50
Allow Manual Adjustment	Doesn't contain a check mark. You can't allow users to manually adjust price in the current update of Pricing.

8. Click **OK**.
9. Repeat steps 5 through 8, except use these values:

Attribute	Value
Shipping Method	Express Delivery
Base Price	100

Create Pricing Matrix

Create a pricing matrix that adds the discounts:

1. Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
2. Add a check mark to each option, then click **OK**.
3. Add a matrix:

Minimum Extended Quantity	Maximum Extended Quantity	Adjustment Type	Adjustment Amount
5	10	Discount Percent	10
11	NA	Discount Percent	25

Minimum Extended Quantity	Maximum Extended Quantity	Adjustment Type	Adjustment Amount

To specify a maximum quantity to infinity, leave Maximum Extended Quantity empty in the row that specifies the highest value.

4. Scroll up, click **Access Sets**, then add an access set.
5. Click **Save > Approve**.

Add Shipping Charge List

Add a shipping charge list to pricing strategy Corporate Pricing Strategy. For details, see *Manage Pricing Strategies*.

Related Topics

- Pricing Rules
- Manage Pricing Strategies

Refund a Shipping Charge

Set up your pricing so you can refund a shipping charge.

I create sales order 768495 and submit it:

Charge	Value
Item x on Line 1	$12.50
Item y on Line 2	$4.50
Freight on Line 3	$6.50
Order Total	$23.50

Sometime later, I create a referenced return for 768495 but it has a total value of $17. The return doesn't include the $6.50 freight charge.

This happens because the Refundable option is disabled on the pricing charge definition that you're using for the freight charge. If you're using the predefined QP_SHIP_FREIGHT pricing charge definition for freight charges, and you want to refund a shipping charge, then you must create your own pricing charge definition. If you're already using your own pricing charge definition for freight, then enable the Refundable option for it. The same situation applies for other shipping charge definitions that come predefined, such as QP_SHIP_HANDLING.

Assume you must create a new charge definition for freight:

1. Make sure you have the privileges that you need to administer Order Management, then create a pricing charge definition:

Attribute	Value
CODE	REFUNDABLE_FREIGHT
Name	Refundable Freight
Description	Charge we use for return orders where we refund freight charges.
Applies To	Shipping
Price Type	One Time
Charge Type	Freight
Charge Subtype	Fee
Refundable	Contains a check mark
Setup Enabled	Contains a check mark
Active	Contains a check mark

   For details, see *Manage Pricing Charge Definitions*.
2. Set up the shipping charge list:
   - Make sure you have the privileges that you need to administer Pricing, then open your shipping charge list.
   - Set the value, then click **Approve**:

Attribute	Value
Pricing Charge Definition	Refundable Freight

   For details, see *Manage Shipping Charge Lists*.
3. Make sure you have the privileges that you need to manage sales orders.

4. Go to the Order Management work area, create a return for sales order 768495, and confirm that it includes the freight charge.

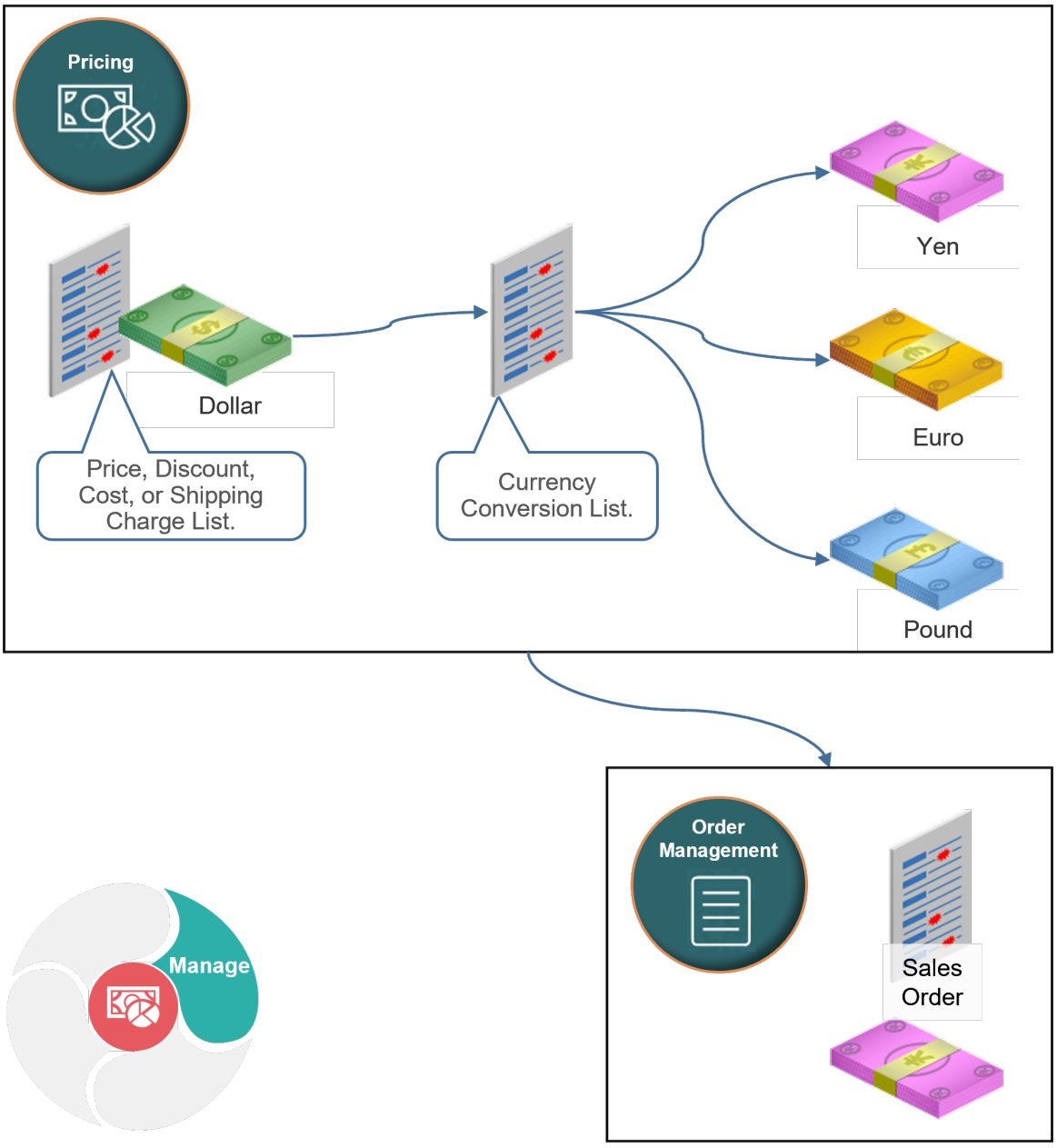
Related Topics

- Pricing Rules
- Manage Pricing Strategies
- Manage Pricing Charge Definitions
- Manage Shipping Charge Lists

Currencies

Currency Conversion Lists

Set up a currency conversion list to specify the conversion rate that you want to use between two currencies, such as between the US Dollar and the Canadian Dollar.



Realize these benefits:

- Do business in more than one currency while maintaining your pricing rules in one currency.
- Set up price values in a single currency but offer items in more than one currency.
- Adjust the conversion rate to accommodate changes that happen in company policy, currency fluctuations, international monetary policy, and so on.
- Set up the currencies and conversion rules that Pricing uses to calculate price at run time so the application that needs price details, such as Order Management, can display prices in the currency that you're familiar with.
- Create a currency conversion rule for each pricing strategy.
- Create one price with more than one conversion type.
- Create markups and markdowns for each currency or country.

- Reduce maintenance because you create only a few lists in Pricing for a pricing strategy when compared to creating a price list for each currency.

Guidelines

- If you don't specify a currency conversion list, then Pricing uses the default currency conversion list.
- You can't create a currency conversion list to convert currency in the general ledger, but you can add a conversion rate for the general ledger when you create a currency conversion list.
- If you encounter a currency conversion error, then you might need to set up a default value for the Currency Conversion Type profile option. For details, see *Use Order Profiles to Control Order Management's Behavior*.
- If you use a currency conversion on a sales order, and then must return that order, then you must set up a conversion for the return. Assume you create currency conversion that converts USD to RMB. You create sales order 12345 that uses the conversion. You now need to create a return order that returns the items you ordered in order 12345. You must create a conversion that converts RMB to USD to support the return.
- You can only round the unit price. You can't round an amount on the order header.

Order Management reprices a sales order differently depending on how you set the currency when you create the sales order:

How I Set Currency When Creating the Order	How Order Management Reprices the Order
I set the currency	Uses the currency that you set on the original order even if you change the currency on the pricing strategy in the Pricing Administration work area or directly on the revised sales order.
I left the currency empty	Uses the currency that's currently defined on the pricing strategy in the Pricing Administration work area or that you set on the revised sales order.

For details about setting currency on the sales order, see *Manage Other Behavior on Sales Orders*.

Override Currency

Use the Pricing Strategies REST API to create, get, update, or delete an override currency for a pricing strategy. Use a finder to get all the override currencies that are currently active:

- Efficiently manage how you override the currency in each of your pricing strategies.
- Process all your override currencies at the same time.

For details and example payloads, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Pricing Strategies > Allowed Override Currencies**.

Currency Conversion Matrix Class

The Currency Conversion matrix class comes predefined to use USD, by default. If you set the currency on your pricing strategy to a value that isn't USD, then you must remove the default value from the matrix class.

Assume you set the currency on your pricing strategy to EURO. Here's what you need to do.

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, in the Name column, click **Currency Conversion**.
3. On the Edit Matrix Class page, in the Domain column, click the **pencil**.

4. In the dialog that displays, change the value, then click **OK > Save and Close**.

Related Topics

- Pricing Rules
- Manage Pricing Strategies
- How Profiles, Segments, and Strategies Work Together
- Use Order Profiles to Control Order Management's Behavior

Manage Currency Conversion Lists

Set up a conversion rate of 1.01 to adjust the US Dollar down by 0.99 to the Canadian Dollar.

You will create a pricing rule. For details, see *Pricing Rules*.

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

Summary of the Set Up

1. Create currency conversion list.
2. Add conversion rate to currency conversion list.
3. Add currency conversion list to pricing strategy.

Create Currency Conversion List

1. Go to **Home > Order Management > Pricing Administration > Currency Conversion Lists**.
2. Click **Create Currency Conversion List**, then enter a value:

Attribute	Value
Name	Currency Conversion List for Standard Desktop You can enter any text.

Add Conversion Rate to Currency Conversion List

1. Click **Add**, then set these values:

Attribute	Value
From Currency	USD US Dollar.
To Currency	CAD Canadian Dollar.
Conversion Type	Fixed.

Attribute	Value
	You can specify the source that Pricing uses for the conversion rate: ○ **Fixed.** Use the fixed rate that you specify. If you select Fixed, then set a rate in the Conversion Rate attribute. ○ **GL Sourced.** Get rates from the GL (general ledger) conversion list or the General Ledger application. GL fixes rates for a time period, such as yearly, quarterly, monthly, weekly, or daily.
Conversion Rate	Enter `1.01`.
GL Conversion Type	Leave it empty. If you set Conversion Type to GL Sourced, then you must set the GL Conversion Type. For example, select Corporate, Quarterly, Monthly, Spot, EMU Fixed, User Defined, and so on. This attribute doesn't affect pricing calculations or relationships. Use it to document how you're using the currency conversion list.
Adjustment Type	Markdown Amount You can specify how to make the adjustment, such as markdown or markup.
Adjustment Amount	Enter `0.2`. You can specify the amount or percent to apply for the adjustment.

2. Add an access set, then click **Activate**.
3. Scroll to the top of the page, click **Access Sets**, then add an access set.

Add Currency Conversion List to Pricing Strategy

For this example, add this currency conversion list to the pricing strategy named Corporate Pricing Strategy. For details, see *Manage Pricing Strategies*.

Example Conversions

These examples use a 1.01 conversion rate to convert USD to CAD:

Adjustment Type	Adjustment Amount	Adjustment Calculated on Conversion Rate	Final Conversion Rate
Markdown Amount	0.2	0.2	1.01 minus 0.2 equals 0.99
Markdown Percent	20%	20% of 1.01 equals 0.202	1.01 minus 0.202 equals 0.808
Markup Amount	0.2	0.2	1.01 plus 0.2 equals 1.21

Adjustment Type	Adjustment Amount	Adjustment Calculated on Conversion Rate	Final Conversion Rate
Markup Percent	20%	20% of 1.01 equals 0.202	1.01 plus 0.202 equals 1.212

Related Topics

- Pricing Rules
- Manage Pricing Strategies
- How Profiles, Segments, and Strategies Work Together

Manage More Than One Currency

Allow your users to select from more than one currency in a single sales order.

In this example, you set up Pricing so your users can order the AS5488 desktop computer from Computer Service and Rentals in the USD currency or Yen currency.

Summary of the Steps

1. Examine predefined behavior.
2. Create pricing strategies.
3. Assign pricing strategies to pricing segment.
4. Test your set up.

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

Examine Predefined Behavior

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area, then create a new sales order:

Attribute	value
Customer	Computer Service and Rentals
Item	AS54888

3. On the order header, click **Additional Order Details > Pricing and Currency**.
4. Notice that the Order Currency includes only US Dollar.
5. Sign out of Order Management.

Create Pricing Strategies

Create two pricing strategies. One strategy will default to USD, and the other will default to Yen:

1. Go to **Home > Order Management > Pricing Administration > Manage Pricing Strategy**.

2. Create a pricing strategy:

Attribute	Value
Name	Pricing Strategy for USD
Business Unit	Computer Service and Rentals
Default Currency	USD US Dollar
Objective	Allow user to select USD
Allow Currency Override	Contains a check mark.

3. Create another pricing strategy:

Attribute	Value
Name	Pricing Strategy for Yen
Business Unit	Computer Service and Rentals
Default Currency	JPY Yen
Objective	Allow user to select Yen
Allow Currency Override	Contains a check mark.

Assign Pricing Strategies to Pricing Segment

1. On the Pricing Administration page, click **Tasks > Manage Pricing Strategy Assignments**.
2. Immediately below the page title, click **Actions > Add Row**, then set the values:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales

Attribute	Value
Transaction Type	Sales Orders
Start Date	Select today's date.

3. Click **Save**.
4. Click **Create Assignment Matrix**.
5. In the Create Assignment Matrix dialog, add a **check mark** to each option, then click **OK**.
6. In the Header-Sales-Sales area, click **Actions > Add Row**, then set the values:

Attribute	Value
Channel Method	Inside Sales
Pricing Segment	Corporate Segment
Transaction Type	Sales Order
Pricing Strategy	Pricing Strategy for USD

7. Click **Actions > Add Row**, then set the values:

Attribute	Value
Channel Method	Inside Sales
Pricing Segment	Corporate Segment
Transaction Type	Sales Order
Pricing Strategy	Pricing Strategy for Yen

8. Click **Save and Close**.

Test Your Set Up

1. Sign into Order Management, then create a sales order:

Attribute	value
Customer	Computer Service and Rentals
Item	AS54888

2. On the order header, click **Additional Order Details > Pricing and Currency**.
3. Verify that you can select US Dollar or Yen.

Related Topics

- Manage Pricing Strategies
- Assign Pricing Strategy
- How Profiles, Segments, and Strategies Work Together

Use Different Currencies for the Same Customer

Set up pricing so you can use different currencies for the same customer.

Assume you must sell the AS54888 item priced in USD (United States Dollar), and the AS54600 item priced in RMB (Renminbi), to customer Computer Service and Rentals, in the Vision Operations business unit:

1. Go to **Home > Order Management > Pricing Administration > Price Lists**..
2. Create the USD price list:
    - On the Price Lists page, click **Actions > Create**, set the values, then click **Save and Edit**:

Attribute	Value
Name	USD Price List
Type	Segment Price List
Currency	USD
Pricing Charge Definition	Sale Price
Business Unit	Vision Operations

Attribute	Value
Line Type	Buy
Calculation Method	Price

- In the Items area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	AS54888
Pricing UOM	Each
Line Type	Buy

- Click **Create Charge**, then set the values:

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	1100
Allow Manual Adjustment	Not enabled

- Click **Access Set**, click **Actions > Add Row**, then set the values:

Attribute	Value
Set Code	COMMON
Set Name	Common Set

- Click **Approve**, then click **Save and Close**.

3. Create the RMB price list:
    - Repeat step 2, but use different values:

Attribute	Value
Name	RMB Price List
Type	Segment Price List
Currency	RMB
Pricing Charge Definition	Sale Price
Business Unit	Vision Operations
Line Type	Buy
Calculation Method	Price

    - Add the item:

Attribute	Value
Name	AS54600
Pricing UOM	Each
Line Type	Buy
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	1300
Allow Manual Adjustment	Not enabled

    - Assign the Common Set access set.

4. Create the pricing strategy:
    - Click **Tasks > Manage Pricing Strategies**.
    - On the Manage Pricing Strategies page, click **Actions > Create**, set the values, then click **Save and Edit**:

Attribute	Value
Name	USD Pricing Strategy
Business Unit	Vision Operations
Default Currency	USD
Default GL Conversion Type	Leave empty
Allow Price List Override	Enabled
Allow Currency Override	Enabled

    For this example, assume Vision Operations uses USD as the default value for most sales orders. So, you name the strategy USD Pricing Strategy, then add currencies to it.
    - In the Segment Price Lists area, click **Actions > Select and Add**, search for USD Pricing Strategy, click the **row** in the search results, then click **OK**.
    - In the Segment Price Lists area, click **Actions > Select and Add**, search for RMB Pricing Strategy, click the **row** in the search results, then click **OK**.
    - Verify that the Segment Price Lists area contains your price lists, then click **Save**:

Name	Business Unit	Currency	Status
USD Price List	Vision Operations	USD	Approved
RMB Price List	Vision Operations	RMB	Approved

    - Click the **Allowed Override Currencies** tab.
    - Click **Actions > Add Row**, then set the value:

Attribute	Value
Currency	RMB

- Click **Approve**, then click **Save and Close**.

5. Test your set up:
   - Go to the Order Management work area, then create a sales order:

Attribute	Value
Customer	Computer Service and Rentals
Item	AS54888

   - Verify that the Amount attribute on the order line displays the value as USD.
   - Create another sales order for Computer Service and Rentals, but this time, in the order header, click **Additional Order Details > Pricing and Currency**, then set Order Currency to Chinese Renminbi.
   - Add the AS54600 item to an order line, then verify that the Amount attribute on the order line displays the value as RMB.

Adjustments

Adjust Price for Pricing Rules

Use tier pricing or a pricing matrix to adjust the price that a pricing rule calculates.

Tier Pricing

Adjust a pricing rule according to quantity or monetary value. For example:

- If the quantity on the sales order is 10 to 19, then reduce the price for each unit by $200.
- If the quantity on the sales order is 20 or more, then reduce the price for each unit by $400.

Pricing Matrix

Adjust a pricing rule according to the value of an attribute. For example:

- If the customer on the sales order is Computer Service and Rentals, then reduce the total price by $50.
- If the customer on the sales order is Computers Direct to U, then reduce the total price by $25.

Example

If Computer Service and Rentals orders 10 computers, then the rule does this calculation:

Description	Calculation
Base price for each item before tier discount	$2,000.
Tier	The quantity of 10 places the sales order in tier 1.

Description	Calculation
Tier adjustment	$200 discount for each unit.
Price for each unit after tier adjustment	$2,000 base price minus $200 discount equals $1,800.
Price for all units after tier adjustment	10 unit multiplied by $1,800 equals $18,000.
Adjustment in pricing matrix	$50 discount off total price.
Price after tier adjustment and matrix adjustment	$18,000 minus $50 equals $17,950.

Lists

You can add an adjustment on these lists:

List	Tier Pricing	Pricing Matrix
Price list	yes	yes
Cost list	yes	no
Currency conversion list	no	yes
Shipping charge list	no	yes

For an example tier adjustment, see *Add Tiers to Pricing Rules*.

For an example pricing matrix, see *Manage Shipping Charge Lists*.

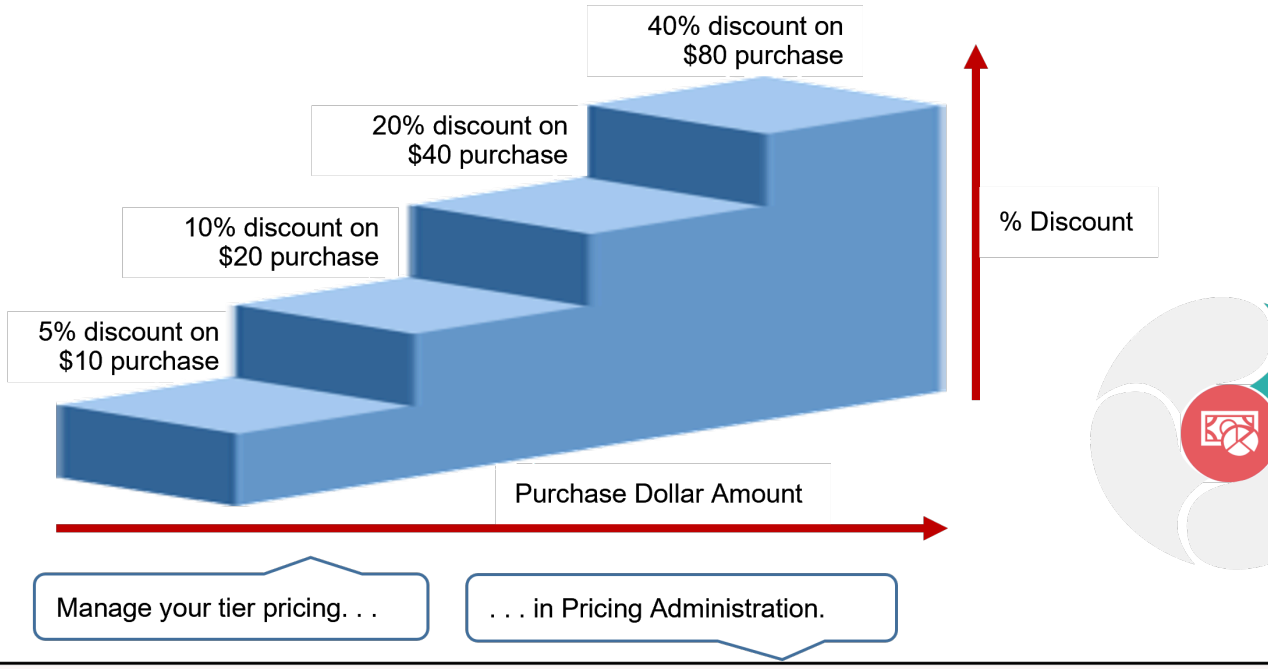
See how a matrix class determines the attributes that you can select in a pricing matrix. See *Pricing Matrix Class*.

Related Topics

- Pricing Matrices
- Tier Pricing

Tier Pricing

Use tier pricing to adjust a pricing rule according to quantity or monetary value.



	More Than	Less Than Or Equal To	Adjustment Type	Value	Adjustment
	10	19	Discount percer	5	List Price
☐	20	39	Discount percent	10	List Price
☐	40	79	Discount percent	20	List Price
☐	80		Discount percent	40	List Price

Note

- Use the price list tier to adjust the base list price.
- Use the tier adjustment on a discount list to adjust list price.
- Adjust a price list or discount list according to quantity or amount.
- Apply tiers on the order line or entire sales order.
- Apply rules to only the highest tier or to all tiers.
- Apply discount percent, discount amount, markup percent, markup amount, or price override.
- Apply pricing on each unit ordered or to a block of units.

For example, add tier pricing to a pricing rule to adjust price according to the number of items that the customer orders. If the customer buys:

- Two desktop computers, use tier 1 to apply a 10% discount.
- Four desktop computers, use tier 2 to apply a 15% discount.

This topic describes attributes you use with tier pricing. For details, see *Add Tiers to Pricing Rules*.

You can use these values when you set the Application Method attribute of a tier pricing rule:

- **Per Unit.** Apply the rule to each unit.

- **As Block.** Apply the rule to a block of units.

Set Up Tiers for Each Unit

Apply the tier pricing rule to each unit. For example, if the sales order includes a quantity of four, and if the discount is $1 for each item, then the total discount is $4.

Use these attributes when you set up a tier pricing rule for each unit:

Attribute	Description
Minimum	Set the lowest value that defines the tier.
Maximum	Set the highest value that defines the tier.
Set the Apply To attribute to All Tiers	Apply the adjustment that each tier specifies.
Set the Apply To attribute to Highest Tier	If the quantity places the sales order in the highest tier, then apply the adjustment that the highest tier specifies to all items, and ignore all other tiers.

Example of Setting Up Tiers for Each Unit

Assume you must set up:

- **Tier 1.** If the customer orders a quantity of one to 10 items, then price each item at $50.
- **Tier 2.** If the customer orders a quantity of more than 10 items, then price each item at $45.
- Override the list price with a price that you specify for each tier.

Pricing assigns items to tiers when the sales order includes a quantity of 15 items, depending on whether you set the Apply To attribute to All Tiers or Highest Tiers:

Tier	All Tiers	Highest Tier
1	10 items	Not applicable
2	5 items	15 items

Pricing calculates these values:

	Minimum	Maximum	Price Override	All Tiers	Highest Tier
Tier 1	1	10	$50	10 items multiplied by $50 equals $500	Not applicable
Tier 2	11	Not applicable	$45	5 items multiplied by $45 equals $225	15 items multiplied by $45 equals $675

	Minimum	Maximum	Price Override	All Tiers	Highest Tier
Price for All Tiers	Not applicable	Not applicable	Not applicable	$500 plus $225 equals $725	$675

Set Up Tiers for Blocks

You use the same attributes to set up tiers for blocks of units that you use when you set up tiers for each unit, plus a few more:

Concept	Description
Block	A quantity of items, such as 100 desktop computers.
Increment	Size of the block. For example, if you specify the Increment as 100, then the block size is 100.
Partial block	A block that includes only part of the full count of items that the Increment specifies for a block. For example, if the Increment is 100, then any block that includes less than 100 items is a partial block, so a block that includes a quantity of 99 is a partial block.
Satisfied block	A block that includes the full quantity that the Increment specifies for a block. For example, if the Increment is 100, then a block that includes a quantity of 100 is satisfied.

If you set Partial Block Action to:

- **Include Partial Block.** Pricing applies the discount to each partial block and to each satisfied block.
- **Include Satisfied Blocks.** Pricing applies the discount only to each satisfied block.

Examples of Setting Up Tiers for Blocks

Assume you must set up these tiers:

Tier	Quantity	Price for Each Item
1	100 to 1,000	10
2	1,001 to 2,000	5
3	More than 2,000	3

Also, you override the list price with a price that you specify for each tier.

If you set the Apply To attribute to Highest Tier Pricing, then Pricing will assign these quantities to tiers:

Minimum	Maximum	Increment	Item Price
1	1,000	100	$10
1,001	2,000	50	$5
2,001	Not applicable	Not applicable	$3

Example of Using All Tiers or Highest Tier

If the sales order includes a quantity of 2,300 items, then Pricing will assign quantities to each tier depending on how you set the Apply To attribute:

Tier	Set Apply To Attribute to All Tiers	Set Apply To Attribute to Highest Tier
1	1,000 items	Not applicable
2	1,000 items	Not applicable
3	300 items	2,300 items

Pricing calculates these values for this example:

	Minimum	Maximum	Increment	Price Override	All Tiers	Highest Tier
Tier 1	1	1,000	100	$10	1,000 items multiplied by $10 each equals $10,000	Not applicable
Tier 2	1,001	2,000	50	$5	1,000 items multiplied by $5 each equals $5,000	Not applicable
Tier 3	2,001	Not applicable	Not applicable	$3	300 items multiplied by $3 each equals $900	2,300 items multiplied by $3 each equals $6,900
Price for All Tiers	Not applicable	Not applicable	Not applicable	Not applicable	$15,900	$6,900

Example of Using Include Partial Block or Include Satisfied Blocks

If the sales order includes a quantity of 850 units, then Pricing will assign all items to tier 1, and will ignore the other tiers.

Assume the list price is $12 each.

Here are the values that Pricing will calculate depending on whether or not you select Include Partial Block or Include Satisfied Blocks:

	Minimum	Maximum	Increment	Price Override	Include Partial Block	Include Satisfied Blocks
Tier 1	1	1,000	100	$10	850 items multiplied by $10 each equals $8, 500	800 items multiplied by $10 each equals $8000
Tier 2	1,001	2,000	50	$5	Not applicable	Not applicable
Tier 3	2,001	Not applicable	Not applicable	$3	Not applicable	Not applicable
Price for All Tiers	Not applicable	Not applicable	Not applicable	Not applicable	$8,500	$8000
List Price for Items Not in Satisfied Bloc	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	50 items multiplied by $12 each equals 600
Total Price for All Items	Not applicable	Not applicable	Not applicable	Not applicable	Not applicable	$8600

Evaluate Block Tiers Across All Lines

Allow Oracle Pricing to consider the quantity from all order lines in your sales order when it prices according to a block of units in a tiered pricing rule. See *Evaluate Block Pricing Tiers Across All Lines*.

Related Topics

- Add Tiers to Pricing Rules
- Manage Shipping Charge Lists
- Pricing Matrices
- Adjust Price for Pricing Rules

Add Tiers to Pricing Rules

Use tier pricing to adjust a pricing rule.

In this example, you add tier pricing that applies discounts on a sales order:

- Apply a 10% discount on all items when the customer buys two or three desktop computers.
- Apply a 15% discount on all items when the customer buys four or more desktop computers.

For details about tier pricing and some of the attributes you set, see *Tier Pricing*.

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

Add tiers to a pricing rule:

1. Create a discount list for the AS54888 Desktop Computer.
   For details, see *Manage Discount Lists*.
2. Add a tier rule.
   The combination of these attributes constitute the tier header:

Attribute	Description
Name	Enter **`Desktop Computer Tiered Discount`**.
Tier Basis Type	For this example, select Item Quantity. Here are the values you can select: o **Item Quantity.** Adjust the tier according to the quantity of items that the customer orders. o **Extended Amount.** Adjust the tier according to the total monetary value of the sales order. You must also select a tier basis to specify the price element that Pricing uses to calculate the tier adjustment, such as One Time or Recurring. For details about tier basis and adjustment basis, see *Manage Pricing Bases*.
Apply To	Highest Tier For details, see the Setting the Apply To Attribute section later in this topic.
Application Method	Per Unit.
Aggregation Method	For this example, select On Line. Here are the values that you can select: o **On Document.** Calculate adjustment according to the total quantity or total amount for the entire sales order or document. o **On Line.** Calculate adjustment according to the quantity or amount for each order line. Pricing applies the tier rule to each order line regardless of how you set Aggregation Method.
Adjustment Type	Leave Adjustment Type empty because you specify the adjustment type in the Tiered Pricing Rules area: o You can include different adjustment types across tiers. o If you set the Adjustment Type in the header, then Pricing defaults the Adjustment Type in each line to the value that you set in the header. For details, see the Setting the Adjustment Type Attribute section in this topic.

Attribute	Description
Enforce Adjustment Calculation on Each Tier	For this example, make sure this option doesn't contain a check mark. If you add a check mark, then Pricing applies the attributes you specify in the Calculation Basis area, such as Application Method, to each of the rules that you specify in the Tiered Pricing Rules area, and you can't modify these attributes in the Tiered Pricing Rules area.

3. In the Tiered Pricing Rules area, set up the first tier. Click **Actions > Add Row**, then enter values:

Attribute	Value
Minimum	0 Pricing calculates the minimum value as Minimum plus 1. For example, if you set Minimum to 0, then Pricing uses 1 as the minimum value for the tier.
Maximum	3
Application Method	Per Unit
Adjustment Type	Discount Percent
Adjustment Amount	10
Adjustment Basis	List Price If you specify a percent discount, then you must specify the adjustment basis that Pricing uses to calculate the adjustment, such as list price or installation charge. For example: o If the list price of a desktop computer is $500, and o You set List Price as the adjustment basis, and o You set the adjustment amount to 10% o Then the discount is $500 multiplied by 0.10 equals $50.

4. In the Tiered Pricing Rules area, set up the first tier. Click **Actions > Add Row**, then enter values:

Attribute	Value
Minimum	3

Attribute	Value
Maximum	Leave empty Pricing will use this tier for any sales order that includes a quantity of four or more.
Application Method	Per Unit
Adjustment Type	Discount Percent
Adjustment Amount	15
Adjustment Basis	List Price

5. Click **OK**.
6. On the on the discount list, click **Save**, and then click **Approve**.

Setting the Apply To Attribute

You use the Apply To attribute to specify the tiers that Pricing applies.

Assume you create these tiers:

Tier	Quantity	Tier Rule
1	Zero to three	10% discount
2	Four or more	15% discount

Pricing applies these tiers differently depending on how you set the Apply To attribute:

Quantity That Customer Orders	Set the Apply To Attribute to All Tiers	Set the Apply To Attribute to Highest Tier
Two	Apply a 10% discount on two items.	Apply a 10% discount on two items.
Four	Apply a 10% discount on three items, and a 15% discount on one item.	Apply a 15% discount on four items.
Six	Apply a 10% discount on three items, and a 15% discount on three items.	Apply a 15% discount on six items.

Setting the Adjustment Type Attribute

Adjustment Type	Description
Discount Amount	Adjust price according to the value you enter. For example, if you enter 15, then decrease price by $15.
Discount Percent	Adjust price according to the discount you specify. For example, if you enter 15, then decrease price by 15%.
Markup Amount	Adjust price according to the value you enter. For example, if you enter 15, then increase price by 15$.
Markup Percent	Adjust price according to the markup you specify. For example, if you enter 15, then increase price by 15% of the tier basis.
Price Override	Adjust price according to the value you enter. For example, if you enter 15, then adjust price by $15.

Pricing applies these adjustments to each item, and it applies them to the value of the Adjustment Basis. In this example, Adjustment Basis is List Price, so it applies the adjustment to the list price.

Related Topics

- Manage Shipping Charge Lists
- Pricing Matrices
- Adjust Price for Pricing Rules
- Tier Pricing
- Manage Pricing Bases

Add Adjustment Matrix to Price List

Add an adjustment matrix to a price list and see how it affects order lines in Order Management.

For example:

- Set the unit price.
- Set the start and end dates when the price applies.
- Use an adjustment matrix. For example, apply a discount according to quantity.
- Use a tier adjustment.

For details, see *Manage Price Lists*.

Assume you must modify a predefined price list that references the AS54888 Desktop Computer:

- Reduce the base price from $2,500 to $2,000 for each computer.
- If the order line quantity is 10 or more, then discount the sale price by 20%.

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

Summary of the Steps

1. Examine the current behavior.
2. Modify the price list.
3. Verify your work.

Examine the Current Behavior

1. Go to the Order Management work area, then create a new sales order.
2. Complete attributes in the header.
3. Add an item on the order line:

Attribute	Value
Item	AS54888, Desktop Computer
Quantity	10
UOM	Each
Amount	2,500

4. Click **2,500** in the Amount column, then examine the details:

Price Components	Unit Selling Price	Amount
Base List Price Applied from Corporate Segment Price List	2,500.00	25,000.00
List Price	2,500.00	25,000.00
Your Price	2,500.00	25,000.00

You will modify the price list so it applies a unit price of $2,000, and a discount of 20%.

5. Click **Done**, and then sign out of Order Management.

Modify the Price List

1. Sign into Oracle Pricing with administrative privileges.
2. Go to **Home > Order Management > Pricing Administration > Price Lists**..
3. Enter the value, then click **Search**:

Attribute	Value
Name	Corporate Segment Price List

4. In the search results, click **Corporate Segment Price List**.
5. On the Price List, enter the value, then click **Search**:

Attribute	Value
Item	AS54888

6. Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
7. Add the Ordered Quantity column.
8. Click **Add Row**, then add one:

Ordered Quantity	Adjustment Type	Adjustment Amount
10	Discount Percent	20

9. Click **Save**, then sign out of Pricing.

Verify Your Set Up

1. Go to the Order Management work area, then create a new sales order.
2. Complete attributes in the header.
3. Add an item on the order line:

Attribute	Value
Item	AS54888, Desktop Computer
Quantity	10
UOM	Each
Amount	1,600

4. Click **1,600** in the Amount column, then verify values:

Price Components	Unit Selling Price	Amount
Base List Price Applied from Corporate Segment Price List	2,000.00	20,000.00
Price list matrix adjustment rule applied because Ordered Quantity Exceeds 10	-400.00	-4,000.00
List Price	1,600.00	16,000.00

Price Components	Unit Selling Price	Amount
Your Price	1,600.00	16,000.00

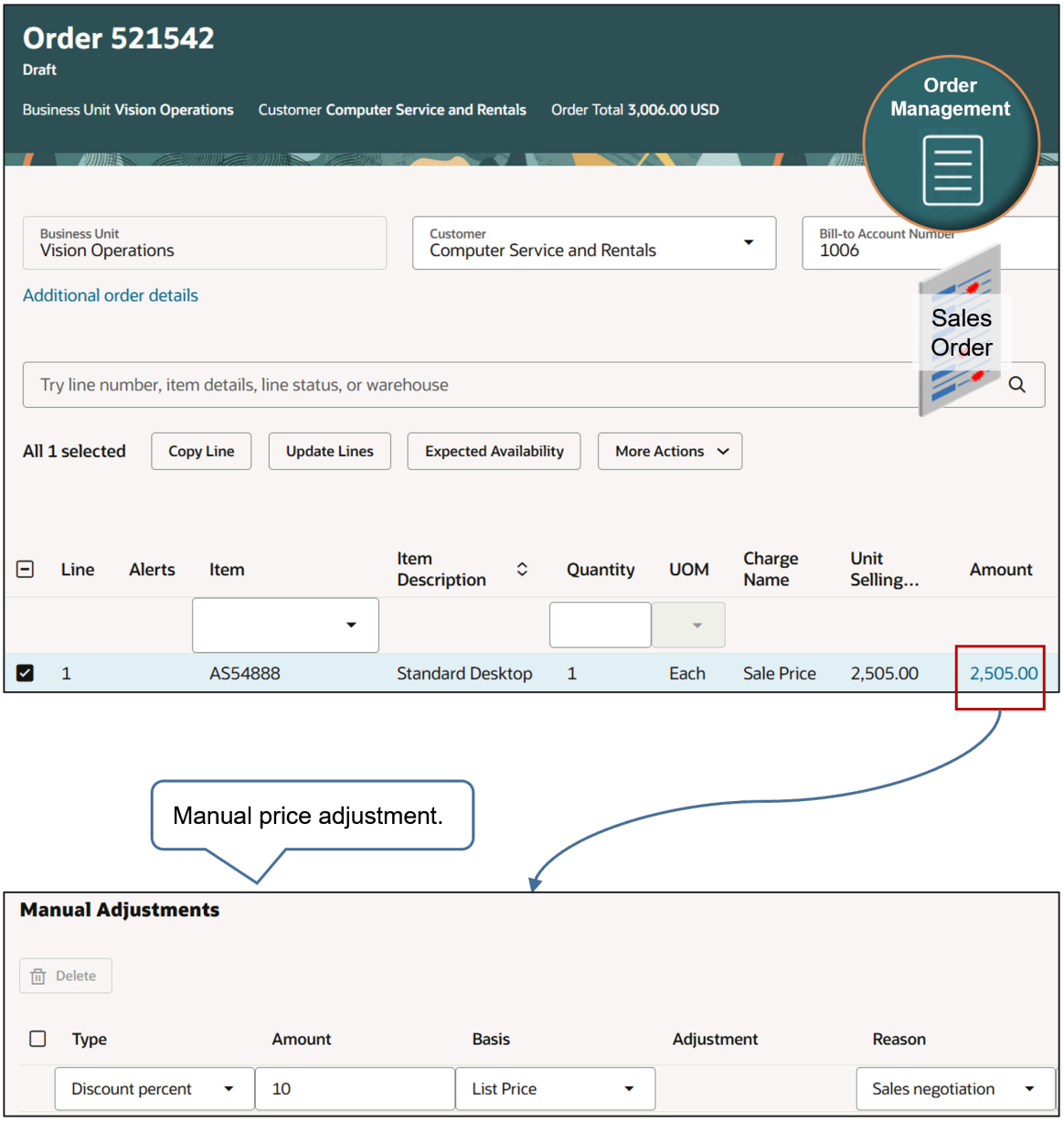
Related Topics

- Pricing Rules
- Manage Price Lists

Allow Users to Adjust Price

Allow your users to adjust price on an order line in Order Management.

For example, click the value in the Amount column on the order line, then add a 10% percent discount off the list price during a sales negotiation:



Pricing can apply a manual price adjustment on a configured item only to the price of the entire item. It can't apply an adjustment on a configurable option.

This topic assumes you already set up Pricing, including adding items to the Corporate Segment Price List. For details, see *Roadmap to Manage Oracle Pricing*.

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

Allow users to adjust the sale price for the AS54111 item:

1. Go to **Home > Order Management > Pricing Administration > Price Lists**..
2. Search for, then open Corporate Segment Price List.

3. On the Price List page, in the Search area, enter the value, click **Search**, then wait for the result to display:

Attribute	Value
Item	AS54111

4. In the AS54111 Charge area, on the Sale Price row, make sure Allow Manual Adjustment contains a check mark, then click **Save**.
5. Optional. Set up a pricing guideline to control the adjustment amount or to prevent an unprofitable discount:
    - Control adjustment to the net price according to a percent of the list price or net price. For example, prevent a manual price adjustment from exceeding 50% of the list price.
    - Control adjustment to the net price according to a value.
    - Set a new value for the net price.
    - Apply one or more adjustments to a charge for each order line.
    - Require a reason for the adjustment.

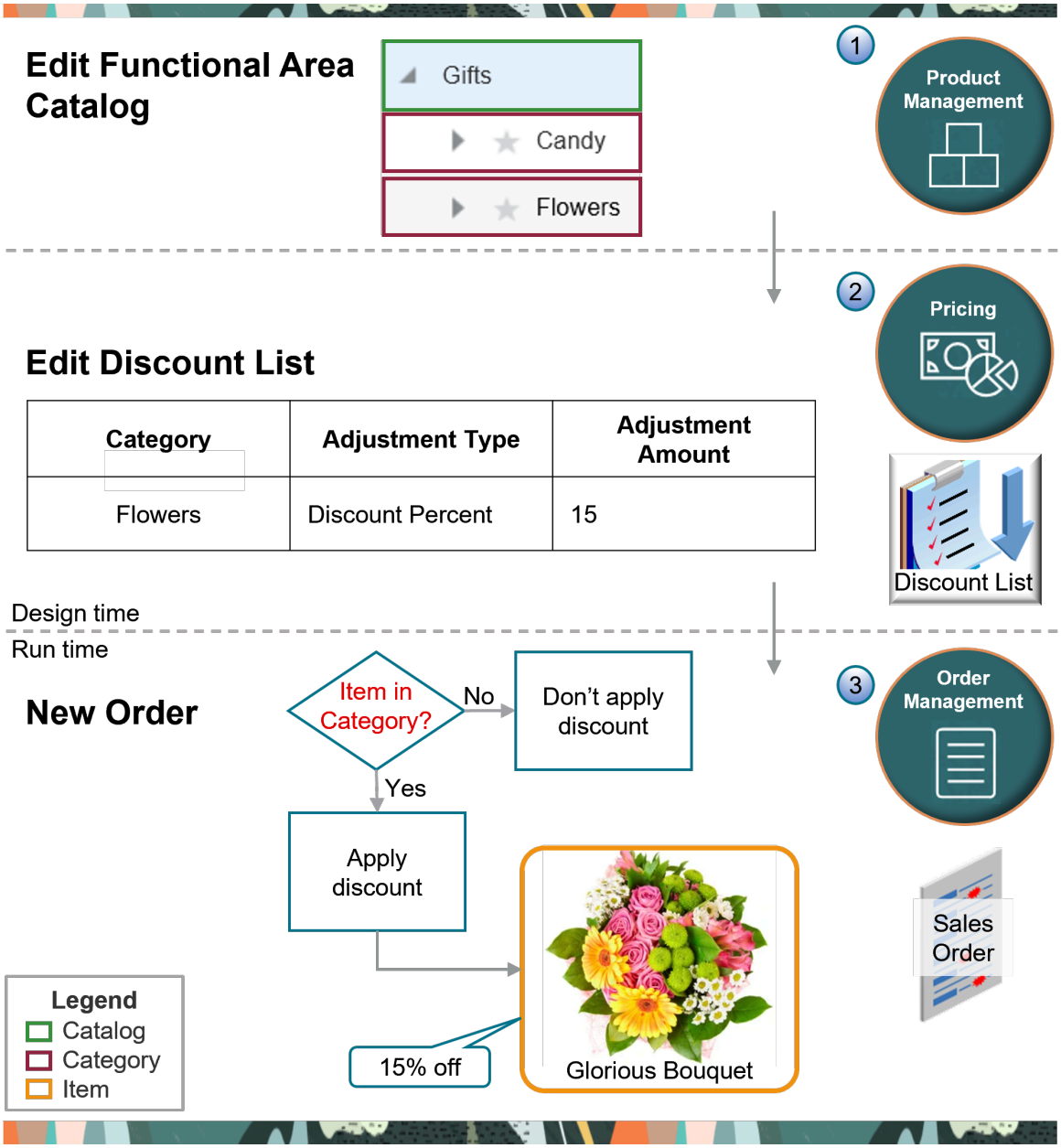
    At run time, Pricing validates each manual price adjustment against the pricing guidelines that you define.

Related Topics

- Roadmap to Manage Oracle Pricing
- Manage Price Lists
- Pricing Guidelines

Adjust Price According to Item's Category

Adjust price on a price list or a discount list according to one or more categories in a catalog.



Note

1. Assume you use the Product Management work area to create a Gifts catalog, and it contains a child Candy category and a child Flowers category. Each child contains items. For example, Flowers contains the Glorious Bouquet item.
2. You use the Pricing Administration work area to create a rule that applies a discount for each item that's in the Flowers category.
3. At run time, you create a sales order in the Order Management work area, and Pricing applies your rule depending on whether the item is in the Flowers category. The Glorious Bouquet is in the Flowers category, so Pricing applies the 15% discount.

Including and excluding provides you with more flexibility when you apply price adjustments. Assume you have 10 items that are in the Flowers category, including the Glorious Bouquet and the California Poppy. You want to apply a 15% on

all 10 items except you don't want to apply that discount on the California Poppy. You can create a rule that applies a discount on all items that are in the Flowers category. You can create another rule that excludes the California Poppy.

Realize these benefits:

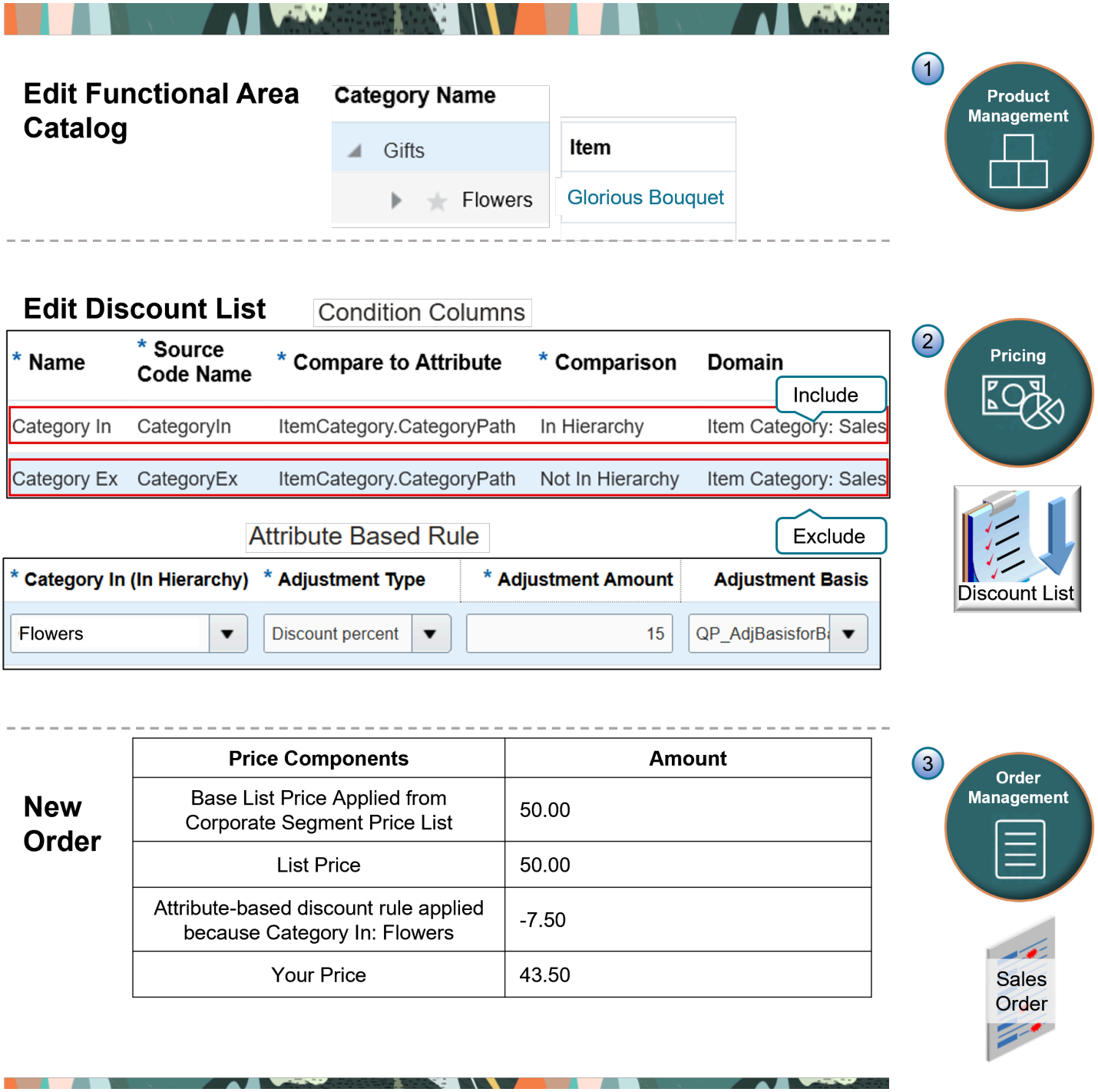
- Have more flexibility when you apply price adjustments.
- Track your category's price adjustments more efficiently.
- Improve financial and pricing analytics.
- Meet more complex pricing requirements.
- Improve decision making that involves category pricing.

Try It

Assume you need to set up this rule:

If the item is the Glorious Bouquet, and if the item is in the Flowers category of the Gifts catalog, then apply a 15% discount.

Here's a summary of your setup:



Note

1. Set up the category's hierarchy and assign the item to the category.
2. Manage the matrix class and discount list.
3. Test your work.

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

Manage Catalog

1. Sign in with the privileges that you need to manage items.
2. Go to the Product Management work area, then click **Tasks > Manage Catalogs**.
3. On the Manage Catalogs page, click **Create**.
4. Set the values:

Attribute	Value
Functional Area	Order Entry You can select any area.
Catalog Name	Gifts
Catalog Code	Gifts
Description	My gift catalog
Controlled At	Master Level You can select any level.
Category Name	Flowers
Category Code	Flowers
Description	My default category

5. Click **Save and Continue**.
6. On the Edit Functional Area Catalog page, under Category Name, expand **Gifts**, then click **Flowers**.
7. In the Details area, click **Select and Add**.
8. Search for the Glorious Bouquet item, then click **Apply > OK**.
9. Run the *Refresh Product Hub Snapshots* scheduled process.
10. Sign out, then sign in with the privileges that you need to administer Oracle Order Management.
11. Go to **Home > My Enterprise > Setup and Maintenance > Manage Pricing Parameters**.
12. On the Manage Pricing Parameters page, click the **row** that has Product Catalogs in the Name column.
13. In the Product Catalogs area, click **Add Row**, add the Gifts catalog, then click **Save and Close**.

Manage the Matrix Class

1. Sign out, then sign in with the privileges that you need to administer pricing.
2. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
3. On the Manage Matrix Classes page, click **Pricing Term Adjustment**.
4. On the Edit Matrix Class page, create the condition that you'll use to include the item in the adjustment:
    - in the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Category In

Attribute	Value
Source Code Name	CategoryIn
Comparison	In Hierarchy
Compare to Attribute	ItemCategory.CategoryPath

- Click **Save**.
- In the Domain column, click the **pencil**, set the values, then click **OK**:

Attribute	Value
Domain Type	Item Category
Catalog	Gifts

- Click **Save**.

5. While we're here, let's go ahead and add a condition that you can use to exclude the item. We'll talk about excluding later:
   - In the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Category Ex
Source Code Name	CategoryEx
Comparison	Not In Hierarchy
Compare to Attribute	ItemCategory.CategoryPath

   - In the Domain column, click the **pencil**, set the values, then click **OK**:

Attribute	Value
Domain Type	Item Category
Catalog	Gifts

   - Click **Save and Close**.

Create Your Rule

1. Click **Manage Discount Lists**, then search for and open the list that you want to use.
2. Add the Glorious Bouquet item to your discount list, then click **Save**.
3. On the Edit Discount Lists page, in the Discount Rules area, click **Actions > Create > Attribute Based Rule**.
4. In the dialog that displays, under Select Optional Condition Columns, add a check mark to Category In (In Hierarchy), then click **OK**.

5. In the Create Discount Rule dialog, in the Attribute Based Rule area, click **Actions > Add Row**, set the values, then click OK:

Attribute	Value
Category In	Flowers
Adjustment Type	**Discount Percent**
Adjustment Amount	15
Adjustment Basis	QP_AdjBasisforBaseListPrc

6. Click **Access Sets**, then set the value:

Attribute	Value
Set Code	Vision Operations For this example, assume you're selling in the Vision Operations business unit.

7. Click **Approve > Save and Close**.
8. Add your discount list to the pricing strategy that you use to price this item.

Test Your Work

1. Go to the Order Management work area.
2. Create a sales order:

Attribute	Value
Business Unit	Vision Operations
Order Type	Standard Orders

3. Search for the Glorious Bouquet item on the order line, then click the **price**.
4. Examine the charge breakdown in the Line dialog. Verify that the breakdown says:

```
Attribute-based discount rule applied because Category In: Flowers In Hierarchy Flowers
```

Match Any Part of the Catalog's Hierarchy

You can set up your rule so it applies pricing when the condition matches some other part of the catalog's hierarchy, such as a category's parent. Assume you have this hierarchy:

```
Gifts
 Candy
 Flowers
 Bouquets
 Stems
```

Assume you need this rule:

If the item is in the Bouquets category, and if the Bouquets category is a child of the parent Flowers category, then apply a 10% discount on the Glorious Bouquet item.

Use these values when you create your rule:

Attribute	Value
Category In	Flowers
Adjustment Type	Discount Amount
Adjustment Amount	10

At run time, the charge breakdown will say:

Attribute-based discount rule applied because Category In: Bouquets In Hierarchy Flowers

Reference a Different Category

You can also set up your rule so it references another category. For example, you can reference the Seasonal Promotion category from the condition that you specify on the discount rule from the Flowers category. At run time, Pricing will apply the discount from the Seasonal Promotion category to the Glorious Bouquet.

Exclude a Category

You can set up your rule so it excludes items that are in a category. Assume you have this hierarchy:

```
Gifts
 Candy
 Flowers
 Bouquets
 Glorious Bouquet
 Stems
 California Poppy
```

Assume you need this rule:

Apply a 15% discount on the Glorious Bouquet item, but don't apply a discount on the California Poppy item.

Set these values on your discount rule:

Attribute	Value
Category Ex	Stems
Category In	Bouquets
Adjustment Type	Discount Percent
Adjustment Amount	15

At run time, Pricing will apply the discount for the Glorious Bouquet, and the charge breakdown will say:

Attribute-based discount rule applied because Category In: Bouquets In Hierarchy Flowers Category Ex: Bouquets No In Hierarchy Stems

The charge breakdown won't have a discount for the California Poppy.

Guidelines

You must:

- Use the Price List Charge Adjustment matrix class for price lists.
- Use the Pricing Term Adjustment matrix class for discount lists.
- Use the Item Category domain type when you create your condition on the Manage Matrix Classes page.
- Set the Comparison attribute to In Hierarchy or to Not In Hierarchy on your rule.
- Assign the catalog to the condition name so that the Pricing Administration work area displays the relevant categories when you set up your rule.
- Not write a pricing algorithm that creates a charge according to a category.

Note

- You can use the Pricing Administration work area, file-based data import, or REST API to manage your adjustment rules.
- If more than one rule is eligible to apply the discount, then Pricing will use the rule that has deepest category in the hierarchy when it prices the item. For example, if one rule is eligible for Flowers, and another rule is eligible for Bouquets, then Pricing will use the rule for Bouquets.

Opt In

We recommend that you use the procedure described in this topic to price according to a category:

- The Pricing Administration work area will display the Create Product Category Charge action and the Product Category area only if you enable these options.
- You don't have to enable the Adjust Price According to the Item's Category opt-in feature to use the procedure described in this topic.
- See *Opt Into Features in Order Management*.

4 Charges, Elements, and Rounding

Charges and Elements

Manage Pricing Charge Definitions

Create a pricing charge definition that defines the charges Oracle Pricing combines to determine the total price for an item.

An item might include more than one charge, such as a one-time sale charge and an administration charge. You can create a pricing charge definition that adds a handling fee to this item.

You reference a pricing charge definition from a pricing rule that you create in the Pricing Administration work area. For examples, see *Manage Price Lists* and *Manage Cost Lists*.

Manage pricing charge definitions:

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 Manage Pricing Charge Definitions task.
3. On the Manage Pricing Charge Definitions page, in the Search area, set Active to **Yes**, then click **Search**.
4. Examine the pricing charge definitions that come predefined with Pricing, such as for price, shipping, freight, handling, insurance, or returns.

   To reduce maintenance, use a predefined pricing charge definition instead of creating a new one.
5. If you can't locate a pricing charge definition that meets your requirements, then click **Actions > Add Row**, then set the values:

Attribute	Description
Code	Enter an abbreviated text that describes how you plan to use this pricing charge definition. Use this format: o Use all upper case letters. o Use only alphabetic text. Don't use an alphanumeric or numeric value. o Don't use the QP prefix. o Don't include any spaces. o Include no more than 30 characters. o User the underscore (_) character to separate words. For example, VISION_RECURRING_SALE_PRICE.
Name	Enter some user friendly text that describes how you plan to use this pricing charge definition. For example, Recurring Sale Price for Vision Factory 5.
Applies To	Apply the pricing charge definition to a price charge, return charge, or a shipping charge.

Attribute	Description
Price Type	Choose One Time or Recurring to specify how Pricing uses the charge for the item. Use Recurring for a charge that recurs, such as a subscription.
Charge Type	Choose the type of charge that Pricing uses for the item: o **Sale.** For a tangible item, such as a computer monitor. o **Service.** For a service, such as performing service on a computer monitor. o **Freight.** For freight charges, such as the charges incurred to ship an item.
Charge Subtype	Provide details about the charge, such as whether the charge is a Price or a Fee. For example, a service charge might include an installation charge and a delivery fee.
Price Periodicity UOM Class	Specify the UOM class that Pricing uses when it applies a recurring charge. For example, Time.
Tax Charge Type	Specify how to use this pricing charge definition for tax purposes, such as Commercial discount, Freight charge, Insurance charge, Miscellaneous charge, or Packing charge.
Refundable	Add a check mark to allow Pricing to refund the charges that this pricing charge definition references. For example, enable this option to refund freight charges when you create a return order. If you don't enable it, you can still create a return but the order total on the return won't include freight charges from the original order.
Setup Enabled	Add a check mark to associate the pricing charge definition with pricing entities: o If the Active option contains a check mark, and if the Setup Enabled option doesn't contain a check mark, then the pricing charge definition can use the pricing entities that it currently references, but you can't associate other pricing entities with the pricing charge definition. o If you must disable a pricing charge definition but must not delete it because historical data continues to use it, then remove the check mark from Setup Enabled. Pricing will prevent you from assigning a pricing charge definition to a pricing entity in the Pricing Administration work area, but the pricing charge definition will remain active so it can support records that contain historical data.
Active	Add a check mark to make the pricing charge definition available for use. Make sure the Setup Enabled option and the Active option each contain a check mark so Pricing can use the pricing charge definition at runtime.
Calculate Margin	Add a check mark to calculate the profit margin for the pricing charge definition.

Manage Pricing Elements

Use a price element to capture different price points in a price calculation.

A price point is the price you set for a charge. Base list price, tier adjustment, list price, discount adjustment, net price, and cost of goods sold are each an example of a price point.

A price element is an object that a pricing algorithm uses to capture different types of prices, costs, adjustments, taxes, or profit margins that it requires to create a charge breakdown or pricing analytic.

Pricing calculates the value for each price element when it runs the pricing algorithm. List price is an example of a price element. It might calculate the value of the list price for the AS54888 Desktop Computer at $2,500.

Manage price elements:

1. Go to the Setup and Maintenance work area, then go to the Manage Price Elements task.
2. On the Manage Price Elements page, in the Search area, set Active to **Yes**, and then click **Search**.
3. Examine the price elements that come predefined with Pricing.

   To reduce maintenance, use a predefined price element instead of creating a new one.
4. If you can't locate a predefined price element that meets your requirements, then click **Actions > Add Row**, then set the values:

Attribute	Description
Element Code	Enter a unique code that identifies the price element. The code makes sure you don't create a duplicate price element that might result in a conflict error.
Element Name	Enter a name. Pricing displays the name in the charge breakdown and in the Pricing Administration work area. You can modify the name of a predefined price element.
Type	Choose a type, such as Price or Cost.
Used in Pricing Guidelines	Enable this option to add more control in your pricing guidelines. For example, price element Base List Price comes predefined to enable you to use it in a pricing guideline. Assume you create 10 pricing guidelines, and six of them enable the user to modify the base list price, among other elements. Two years later, your company creates a policy that prohibits users from modifying the base list price. Instead of revising the six guidelines, you can disable Used in Pricing Guidelines for price element Base List Price. You can't enable this option with an element of type Cost or type Accrual because Pricing calculates the value for these elements.
Active	Add a check mark. Pricing only uses pricing elements that are active.

Attribute	Description
	If the pricing entity: o **Doesn't reference the price element.** You can remove the check mark to remove the price element from calculations, but keep it for possible later use. o **References the price element.** You can't remove the check mark.

Related Topics

- Pricing Algorithms
- Manage Price Lists
- Manage Cost Lists
- Pricing Guidelines

Manage Pricing Bases

Set up the pricing basis that Oracle Pricing uses to calculate an adjustment.

Each pricing basis references a price element, such as List Price or Invoice Price, to calculate the discount that it applies on an item. The price element is one of the components on the pricing charge.

You can manage these pricing bases:

Type of Pricing Basis	Description
Adjustment Basis	Adjust the price according to a percent: • For example, create an adjustment basis that applies a 20% discount on the list price of a network of desktop computers. For another example, apply a 10% discount on the installation charge for a network of desktop computers. • Set up the adjustment on the list price, base price, or net price, resulting in different invoice values for the network of computers.
Tier Basis	Adjust the price according to an amount: • A tier basis aggregates price for a single pricing charge or a set of pricing charges. For example, if a one-time sale charge is greater than $2,000, then apply a 10% discount on the base price. For another example, determine the one-time sale charges for the QP LIST PRICE charge component. • The tier basis contains the details of the pricing charges and the charge components that Pricing uses to calculate the adjustment. It references the price type, charge type, charge subtype, and frequency. These details are important when more than one charge exists for an item.

Type of Pricing Basis	Description
	• Include or exclude charges. For example, assume a third-party contractor installs the item you sell, so you don't want to offer a discount on the installation. Create a pricing basis that includes only the one-time sale charge and a monthly recurring service charge, but that excludes the one-time installation charge. • Specify whether to apply the tier basis to each order line or the entire sales order. • To calculate the tier basis at runtime, Pricing aggregates the charges that match the criteria you set up.

Assume you must add a tier basis for the sale, installation, and monthly service for a network of desktop computers. Assume you must apply the tier basis on these charges:

- A one time charge on the list price of the sale
- A one time charge on the list price of the installation fee
- A recurring charge on the list price of the monthly service fee

Assume you sell to a distributor who ships the item and handles the freight and shipping charges, so you must exclude freight charges from your pricing basis.

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

Manage a pricing basis:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Bases task.
2. On the Manage Pricing Bases page, in the Search area, set Active to **Yes**, then click **Search**.
3. Examine the pricing bases that come predefined with Pricing.

   To reduce maintenance, you use a predefined pricing basis instead of creating a new one.
4. If you can't locate a predefined pricing basis that meets your requirements, then click **Actions > Create**.
5. On the Create Pricing Basis page, set the values:

Attribute	Value
Usage	Tier Basis
Price Element	List Price
Price Type	One Time
Charge Type	Sale
Charge Subtype	Price
Active	Contains a check mark

6. Click **Add Charge Criteria**.

7. In the Criteria for Included Charges area, notice that the Create Pricing Basis page already added a row that includes the criteria you specified in step 6.
8. Add the criteria for the installation fee. In the Criteria for Included Charges area, click **Actions > Add Row**, then set the values:

Attribute	Value
Price Type	One Time
Charge Type	Sale
Charge Subtype	Fee

9. Add the criteria for the monthly service fee. In the Criteria for Included Charges area, click **Actions > Add Row**, then set the values:

Attribute	Value
Price Type	Recurring
Charge Type	Service
Charge Subtype	Fee
Frequency	Month Pricing gets the values it displays for Frequency from the UOM class that you specify in the pricing charge definition. If you don' specify this UOM class, then Pricing gets these values from the Default Pricing Periodicity UOM class. For details, see *Manage Pricing Charge Definitions*.

10. Add the criteria for the excluded charges. In the Criteria for Excluded Charges area, click **Actions > Add Row**, then set the values:

Attribute	Value
Price Type	One Time
Charge Type	Freight
Charge Subtype	All

11. Click **Save**.
12. Reference your pricing basis from a pricing rule that you create in the Pricing Administration work area.

    For details, see *Add Tiers to Pricing Rules*.

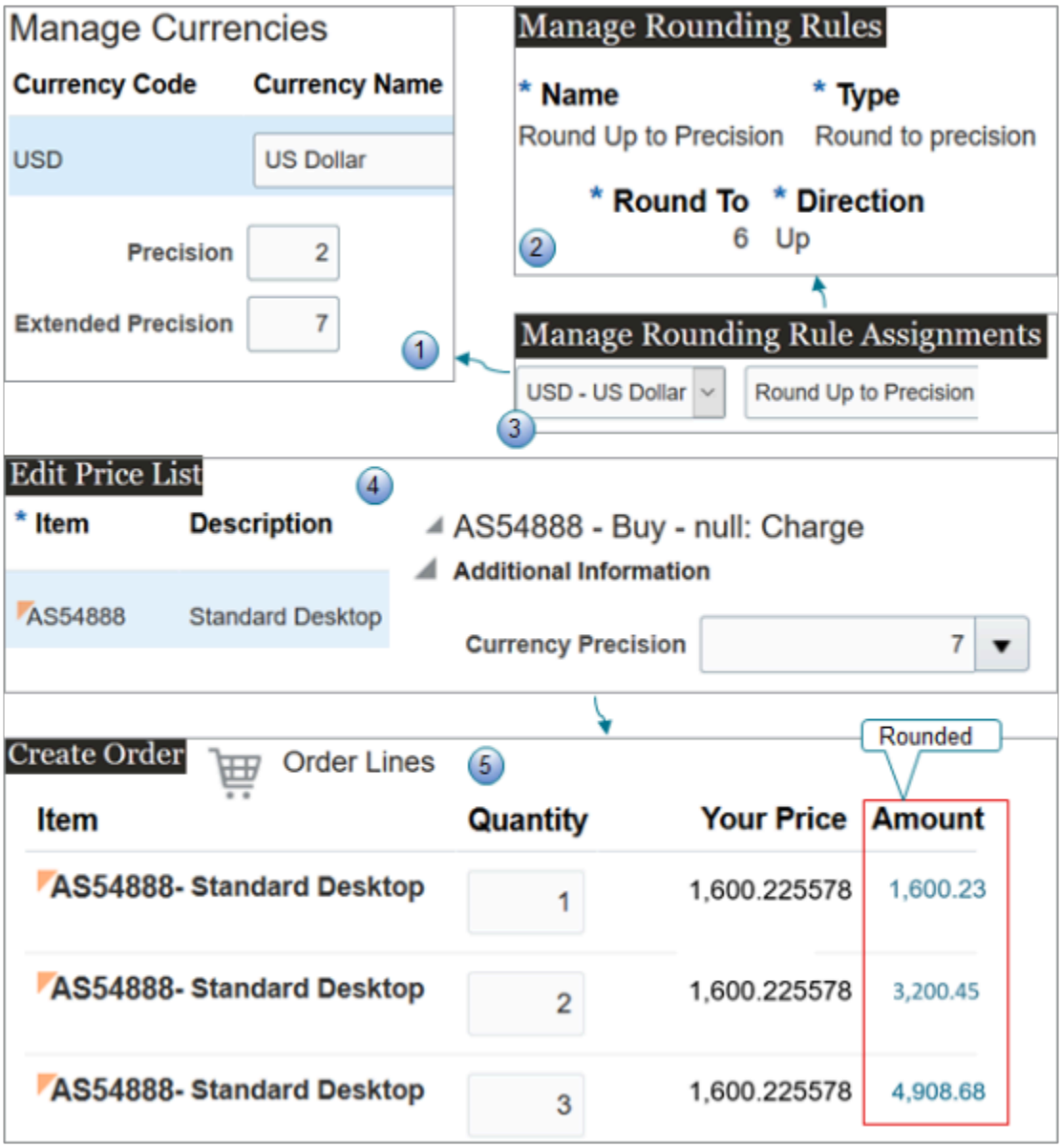
Related Topics

- Manage Pricing Elements
- Add Tiers to Pricing Rules
- Pricing Rules
- Overview of Oracle Pricing
- Manage Pricing Charge Definitions

Rounding

Round Your Pricing

See how rounding and precision affects prices on order lines.



Note

1. Use the Manage Currencies task to set precision for the currency:
    - Use the Precision attribute to specify the number of decimal places to the right of the decimal point.
    - Use the Extended Precision attribute when you need even more precision. It also specifies the number of decimal places to the right of the decimal point, but you can set Extended Precision to a higher value.
2. Use the Manage Rounding Rules task to set precision for rounding.
3. Use the Manage Rounding Rule Assignments task to assign currency to the rounding rule.
4. Use the Price List page to edit the charge for your item on the price list and to set the currency precision for the charge.

5. Create a sales order and see how Pricing applies your set up at run time:
    - The Extended Precision attribute that you set on the currency affects the Your Price attribute on the order line. In this example, it's six decimal places, rounded up from 36.2255779.
    - The Currency Precision attribute that you set on the charge affects the Amount attribute on the order line.
    - Pricing starts with the Base List Price attribute on the price list, uses your rounding rule to round the value, then applies other set ups that you have for the item, such as discounts, other charges, and so on.

Assume you sell the AS54888 Desktop Computer in the USD currency. You need to round the USD currency with a precision of 2, extended precision of 7, and round it up according to a precision of 6.

Try it:

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Currencies**.
2. On the Manage Currencies page, search for the value:

Attribute	Description
Currency	USD

3. Expand the row in the search results.
4. Set the values, then click **Save and Close**:

Attribute	Description
Precision	2
Extended Precision	7

5. Go to the Manage Rounding Rules task.
6. On the Manage Rounding Rules page, click **Actions > Add Row**.
7. Set the values, then click **Save and Close**:

Attribute	Description
Name	Round Up to Precision
Type	Round to Precision
Round To	6
Direction	Up
Setup Enabled	Contains a check mark.

Attribute	Description
Active	

8. Go to the Manage Rounding Rule Assignments task.
9. On the Manage Rounding Rule Assignments page, click **Actions > Add Row**.
10. Set the values, then click **Save and Close**:

Attribute	Description
Currency	USD
Rounding Rule	Round Up to Precision

11. Create a charge:
    - Make sure you have the privileges that you need to administer pricing.
      Go to the Pricing Administration work area.
    - Click **Price Lists**.
    - On the Price Lists page, search for and open the Corporate Segment Price List.
    - On the Price List page, in the search results, click **Actions > Add Row**, then set the values:

Attribute	Description
Item	AS54888

    - Click **Create Charge**.
    - In the AS54888 Charge area, expand **Additional Information**, then set the values:

Attribute	Description
Currency Precision	7

    - Click **Save and Close**.
12. Test your set up:
    - Go to the Order Management work area, create a sales order, then add 3 order lines:

Order Line	Item	Quantity
1	AS54888	1

Order Line	Item	Quantity
2	AS54888	2
3	AS54888	3

- Verify the price and amount for the AS54888 item. Assume that Your Price is $36.225578:

Quantity	Amount
1	$36.23 Here's the calculation: i. Multiply 36.225578 by a quantity of 1 equals 36.225578. ii. Round 36.225578 up according to a decimal precision of 2, equals 36.23.
2	$72.45 Here's the calculation: i. Multiply 36.225578 by a quantity of 2 equals 72.451156. ii. Round 72.451156 up according to a decimal precision of 2, equals 72.45.
3	$108.68 Here's the calculation: i. Multiply 36.225578 by a quantity of 3 equals 108.676734. ii. Round 108.676734 up according to a decimal precision of 2, equals 108.68.

Your Price is six decimal places, rounded up from 36.2255779.

Related Topics

- Types of Rounding Rules
- Pricing Algorithms
- Manage Rounding Rules
- Control Decimal Precision

Manage Rounding Rules

Use a rounding rule to replace a value with some other value that's shorter, simpler, or more explicit.

Apply a rounding rule to a price to make sure you apply rounding consistency across pricing algorithms.

Create a complex rounding rule. For example, create rounding rules to meet the needs of a local cultural preference, or that your company policy requires:

- Round each price to 0.97 or 0.99 for companies that reside in the United States, and round each price to 88 for companies that reside in China.
- Round each price that includes a value:
  - Between 1 and 100 to a precision of 4
  - Between 100.001 and 10000 to a precision of 2
  - Greater than 10000.01 to a precision of 0

Summary of the Setup

1. Create the rounding rule.
2. Assign the rounding rule to a currency.
3. Refresh the order promising server.

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

1. Create the Rounding Rule

1. Go to the Setup and Maintenance work area, then go to the Manage Rounding Rules task.
2. On the Manage Rounding Rules page, in the Name attribute, enter **%**, then click **Search**.

   The Manage Rounding Rules page displays the predefined rounding rules.
3. Examine the predefined rounding rules.

   To reduce maintenance, use a predefined rounding rule instead of creating a new one.
4. If you can't locate a predefined rounding rule that meets your needs, then click **Actions > Add Row**, then set the values:

Attribute	Description
Name	Enter text that Pricing can use to reference the rounding rule: - Enter alphanumeric text. - Use headline capitalization. - Use an underscore to separate each word.
Type	For details, see *Types of Rounding Rules*.
Round To	For details, see *Types of Rounding Rules*.
Direction	Set to one of these values: - **Up.** Round values up. - **Down.** Round values down.

Attribute	Description
	○ **Standard.** Round up or down to the nearest value. For example, if the value is $1.75, then round up to $2. If the value is $1.25, then round down to $1.
Incremental Value	For details, see *Types of Rounding Rules*.
Setup Enabled	Enable or disable: ○ **Add a check mark.** Make this rounding rule available at runtime. ○ **Remove the check mark.** Make this rounding rule available at run time, but prevent a user from assigning a currency to it on the Manage Rounding Rule Assignments page. This feature is useful when you must discontinue usage of a rounding rule but can't delete it because historical data still uses it.
Active	Add a check mark to make this rounding rule available at runtime.

5. Click **Save and Close**.

2. Assign the Rounding Rule to a Currency

As an option, you can assign the rounding rule to a currency. If your business uses more than one currency, then you can specify the currency that the rounding rule affects. Use this feature to create more than one rounding rule, then assign a different currency to each of them:

1. Go to the Setup and Maintenance work area, then go to the Manage Rounding Rule Assignments task.
2. On the Manage Rounding Rule Assignments page, click **Actions > Add Row**, then set the values:

Attribute	Description
Currency	Choose the currency that you must assign to this rounding rule.
Rounding Rule	Choose the rounding rule that you created earlier in this topic. At runtime, Pricing applies this rounding rule only to an item that uses the currency that you set in the Currency attribute.

3. Refresh the Order Promising Server

Refresh the Order Promising server each time you add, modify, or remove a rounding rule:

1. Go to the Scheduled Processes work area.
2. On the Pricing Administration page, click **Schedule New Process**, then search for *Refresh and Start the Order Promising Server*.

3. Set parameters, then submit:

Parameter	Value
Fulfillment Lines	Contains a check mark.
Sourcing	Contains a check mark.
Items	Contains a check mark.

4. Verify that the scheduled process finished successfully.

Related Topics

- Types of Rounding Rules
- Pricing Algorithms
- Overview of Oracle Pricing
- Collect Data for Global Order Promising
- Control Decimal Precision

Types of Rounding Rules

Create different types of rounding rules.

- Round to precision
- Round to nearest
- Round to the nearest multiple
- Round to range

The Standard direction rounds up or down to the nearest value for each of these types. For example, if the value is $1.75, then round up to $2. If the value is $1.25, then round down to $1.

Round to Precision

Here are some examples where Round To specifies to round according to a decimal place value:

Price	Round To	Direction	Rounded Value
$15.75	0	Up	$16.00
$15.75	0	Down	$15.00
$15.75	0	Standard	$16.00

Price	Round To	Direction	Rounded Value
$187.5	0	Standard	$188.00
$187.57	1	Standard	$187.60
$187.587	2	Standard	$187.59

Consider a few examples:

- The price in the first row of the table is $15.75. Round To equals 0, which means we will round to 0 decimal places. The Direction is Up, so we round the value 15.75 up to the nearest value that doesn't have any decimal values, and that value is 16.
- The price in the last row is $187.587. Round To equals 2, which means we will round to 2 decimal places. The Direction is Standard, so we round up or down to the nearest the value in the second decimal place. The value of the third decimal place in 0.587 is 7, so we round 0.587 up to 0.59.

Round to precision actually means round to scale. 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 123.45 uses a precision of 5 and a scale of 2. So if you set Round To to 2, you're actually not specifying precision. You're specifying to round the value to a scale that has two digits to the right of the decimal point.

Round to Nearest

Here are some examples that use **round to nearest**, where Round To specifies the value to use when rounding:

Price	Round To	Direction	Incremental Value	Rounded Value
$0.22	.05	Up	0.10	$0.25
$0.22	.05	Down	0.10	$0.15
$0.22	.05	Standard	0.10	$0.25
$1.87	0.05	Up	0.1	$1.95
$1.87	0.05	Down	0.1	$1.85
$1.87	0.05	Standard	0.1	$1.85
$198.67	0.05	Up	0.1	$198.75
$198.67	0.05	Down	0.1	$198.65
$198.67	0.05	Standard	0.1	$198.65

Pricing adds the Increment Value according to the value that you set for Round To. Consider the first row in the table:

- If Round To is 0.05, and if increment is 0.1, then the rounded values are 0.05, 0.15, 0.25, 0.35, 0.45 and so on.
- The price starts at 0.22.
- Round To is 0.05, which means to round the hundredth value to 5, so no matter what the direction or increment, the hundredth is going to be 0.05.
- Direction is Up, and the increment is 0.10, so we're going to round the tenth up in increments of 0.10.
- .22 rounded up to 0.05 equals 0.25. The equation doesn't add an increment because the price is already rounded to 0.25.

You can also manually modify the Increment Value. You can specify an increment of a whole, a tenth, a hundredth, or a whole multiple of a decimal, such as 0.5 or 0.25.

Round to the Nearest Multiple

Here are some examples that use **round to the nearest multiple** to round to the nearest multiple of the Round To value:

Price	Round To	Direction	Rounded Value
$15.75	5	Up	$20.00
$15.75	5	Down	$15.00
$15.75	5	Standard	$15.00

For example, if you set Round To to 5, then the multiples of Round To are 5, 10, 15, 20, and so on.

The first row in the table contains a price of 15.75 and a direction of Up. The nearest multiple of 15 when rounding up is 20.

Round to Range

Here are some examples that use **round to range** to round to a range of values that you specify:

Price	Round Value	Rounding Rule Type	Rounded Value	Direction	Increment Value
$0	100	Round to nearest	0.99	Standard	1
$100	10,000	Round to nearest	9	Standard	10
$10,000	10,500	Round to value	10,500	Standard	Not applicable

You can identify more than one range and set up a separate rounding rule for each range. Here are the round rule types you can use:

Rounding Rule Type	Description
Round to precision	Round to a precision.
Round to nearest	Round to the nearest value.
Round to the nearest multiple	Round to the nearest multiple of the value.
Round to value	Round to a fixed value.

You can set only one rounding rule for each range of values.

You must create at lease one range:

1. Set the Type attribute to Round to Range on the Manage Rounding Rules page.
2. In the Ranges area, click **Actions > Add Row**.
3. Set the values for at least the Range From, Type, and Round To attributes.
4. Click **Save**.

Related Topics

- Manage Rounding Rules

5 User Interfaces

Totals, Order Lines, Lookups, Flexfields

Manage Pricing Totals

A pricing total is the sum of more than one charge. Use it to combine and display the values of charges as a single value.

- Oracle Pricing uses the Calculate Sales Order Totals pricing algorithm to calculate the totals that Order Management displays in the charge breakdown when the order entry specialist creates a sales order.
- Use the Manage Pricing Totals page to set up the pricing totals to include in the dialog.
- Each pricing total references a pricing algorithm that calculates the total. For example, the predefined Total List Price pricing total references a pricing algorithm that adds up the extended amounts of each one-time price charge for every Buy line for the charge components where the price element equals LIST_PRICE. For details, see *Pricing Algorithms*.

Assume Order Management displays these values in the charge breakdown:

Attribute	Value
Total List Price	2,500.00
Discount	-10.00
Total Net Price	2,490.00
Shipping	0.00
Total Tax	0.00
Total Credit	0.00
Pay Now	2,490.00

Assume you must modify pricing:

- Calculate total credit before you calculate total tax.
- Move the Total Tax line to immediately below Total Credit.
- Change the Pay Now text to Your Total Price.

You modify the pricing algorithm to change the calculation sequence and the sequence of lines in the dialog, and you use the Manage Pricing Totals page to modify the text.

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

Summary of the Steps

1. Examine the current behavior.
2. Modify the pricing algorithm.
3. Modify the pricing totals setup.
4. Test your set up.

Examine the Current Behavior

1. Sign into Order Management with administrative privileges.
2. Go to the Order Management work area, then create a new sales order.
3. Complete attributes in the header.
4. Add an item on the order line:

Item	Your Price
AS54888, Desktop Computer	2,490

5. Click **2,490** next to Sale Price, then examine the price details.
6. Click **2,490** next to Add, examine price details in the Amount Sale Price dialog, then click **Add**.

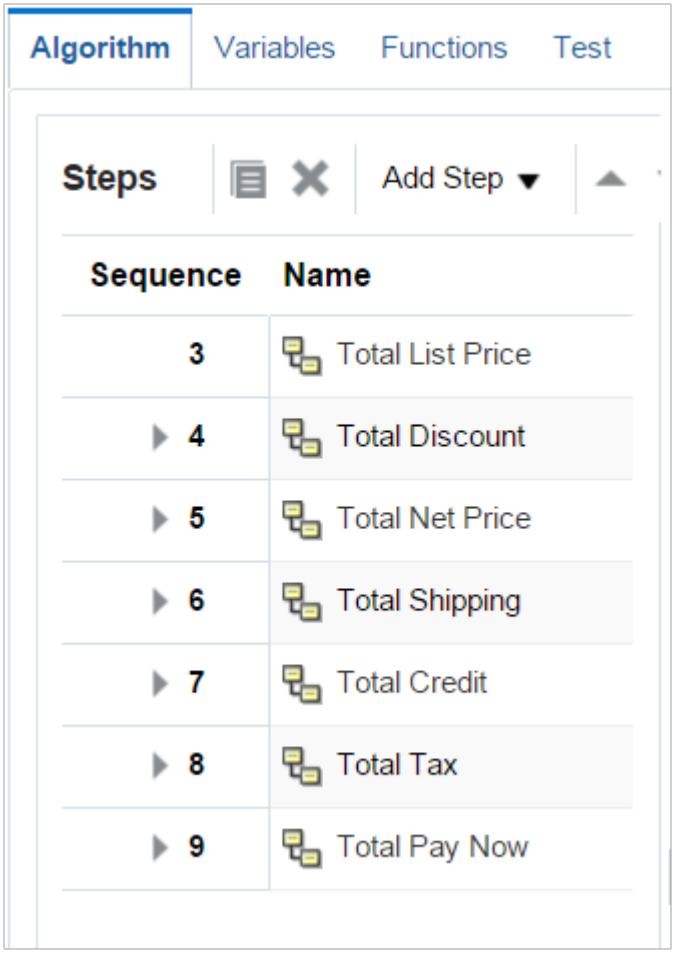
   Learn how to modify the dialog, including how to manage the pricing results presentation. For details, see *Manage Order Line's Price Details*.
7. Click **2,490** next to Total at the top of the page, then examine the price details in the charge breakdown.

   For this example, assume Order Management displays the charge breakdown included earlier in this topic.

Modify the Pricing Algorithm

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click the Calculate Sales Order Totals **row**, then click **Actions > Create Version**.
4. in the row that includes In Progress in the Status column, click **Calculate Sales Order Totals**.

5. On the Edit Algorithm page, click the Total Credit **row**, click **Move Up**, then verify your modification resembles this sequence:



6. Click **Save and Close**.
7. On the Manage Algorithms page, click the Calculate Sales Order Totals **row** that includes In Progress in the Status column, then click **Actions > Publish**.

Modify the Pricing Totals Setup

1. Sign into Order Management with administrative privileges.
2. Go to the Setup and Maintenance work area, then go to the Manage Pricing Totals task.
3. On the Manage Pricing Totals page, in the Search area, set Transaction Enabled to **Yes**, then click **Search**.
4. Notice that the values in the Name column match the values you observed in the charge breakdown.
5. In the Name column, change the value, then click **Save and Close**

Old Value	New Value
Pay Now	Your Total Price

Test Your Set Up

1. Go to the Order Management work area, then create a new sales order.
2. Add the same item you added earlier, click **Add**, click **2,490** next to Total at the top of the page, then verify the charge breakdown displays Total Credit immediately above Total Tax, and that it displays the Your Total Price text.

Related Topics

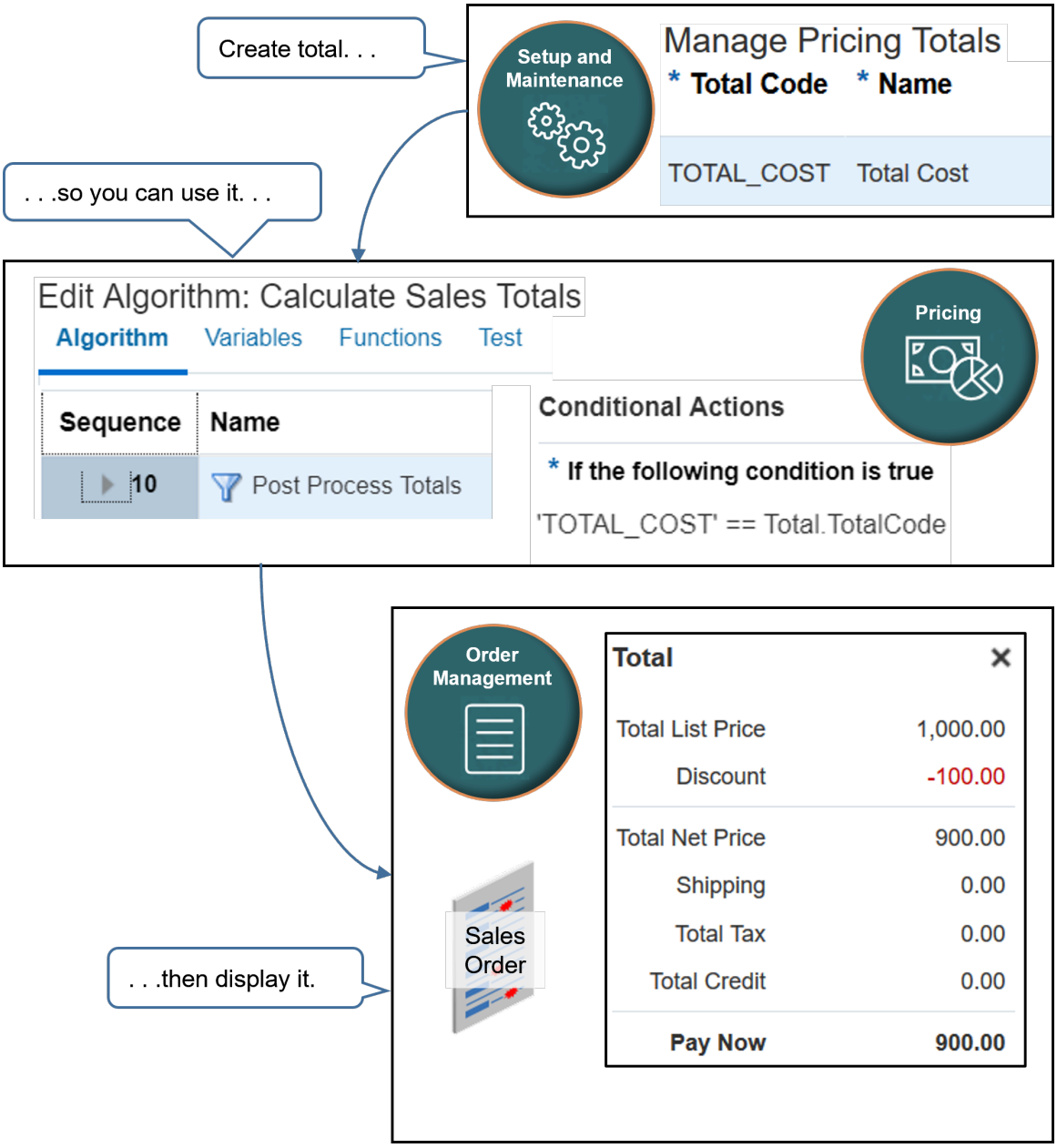
- Manage Order Line's Price Details
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Pricing Algorithms
- Overview of Oracle Pricing
- Manage Pricing Elements

Modify Order Management's Pricing Totals

Modify the pricing totals that the Order Management work area displays.

Assume you need to add total cost and total margin to the charge breakdown, and include the charge amount of each return line in the tax total.

Here's an example that includes pricing for a sales order that includes only return lines:



Summary of the Setup

1. Set up your new totals.
2. Set up the calculate sales totals pricing algorithm.
3. Set up the identify usage totals pricing algorithm.
4. Set up the compute simple totals pricing algorithm.
5. Test your setup.

Set Up Your New Totals

1. Sign into Oracle Order Management with administrative privileges.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Totals**.
3. On the Manage Pricing Totals page, use **Actions > Add Row** to create your new totals:

Total	Name	Description
TOTAL_COST	Total Cost	Total cost for price and shipping charges on the order line.
TOTAL_MARGIN	Total Margin	Total of the margin for price and shipping charges on the order line.

Make sure you enable these options when you add the totals:

- Transaction Enabled
- Return Not Null

Leave the Process Name attribute empty.

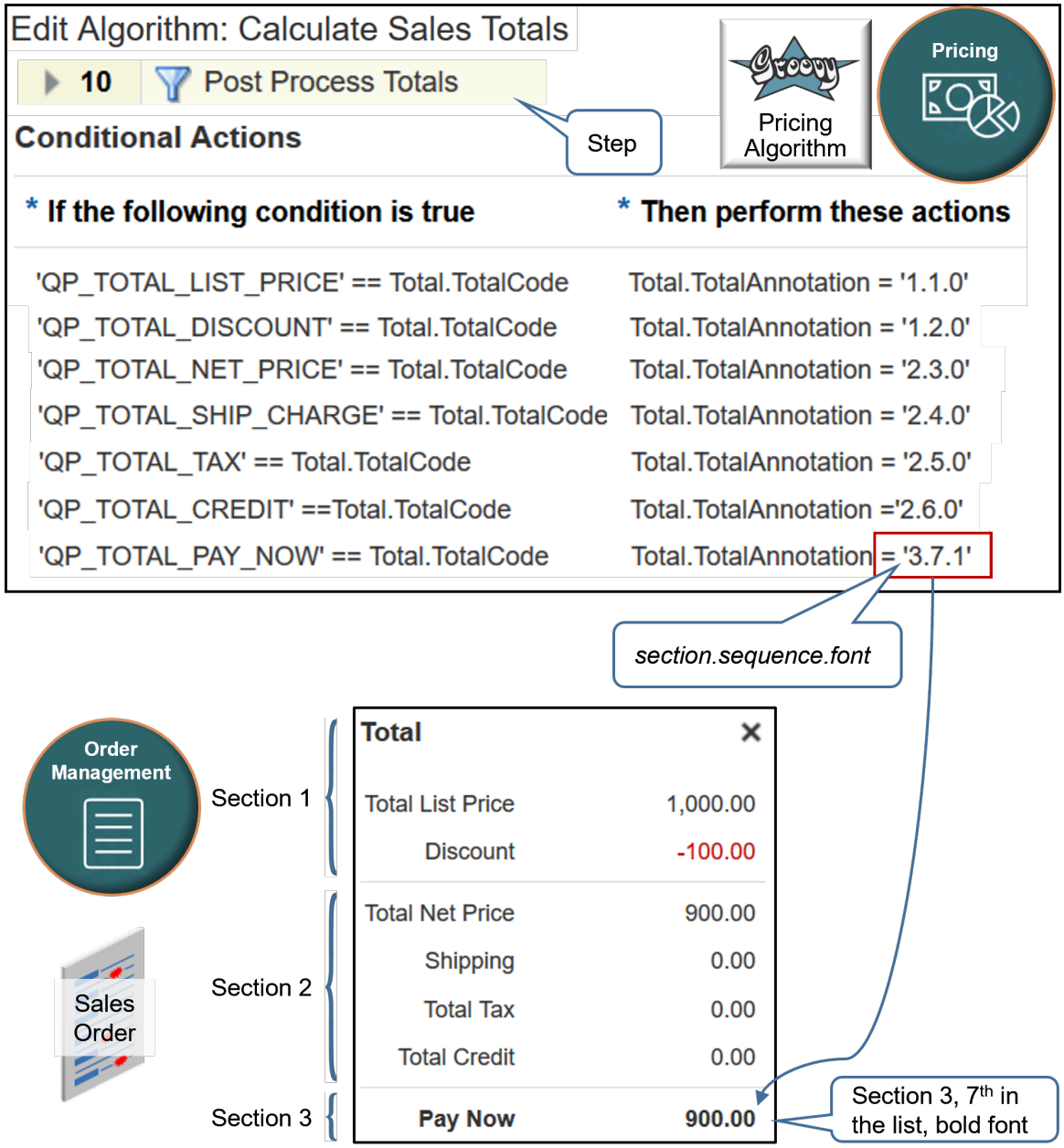
4. Click **Save**.

Set Up the Calculate Sales Totals Pricing Algorithm

Set up the algorithm that calculates sales totals and specifies how to display them in the charge breakdown:

1. Sign out, then sign into Oracle Applications with the privileges that you need to administer pricing.
2. Go to **Home >Order Management > Pricing Administration> Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Query By Example**, then search the Name column for Calculate Sales Totals.
4. Click the **row** that contains the highest value in the Version column, then click **Actions > Create Version**.
5. On the Edit Algorithm page, click the **Post Process Totals** step.

6. In the Conditional Actions area, examine the TotalAnnotation property for each total:



The annotation uses dot notation to specify how each total displays in the charge breakdown.

`section.sequence.font`

where

- `section` specifies a section in the charge breakdown.
- `sequence` specifies where to display the total in the top to bottom sequence of lines in the charge breakdown
- `font` specifies the font style. Use 0 (zero) for font that isn't bold and 1 for bold.

For example:

If This Condition Is True	Then Do This Action
'QP_TOTAL_PAY_NOW' == Total.TotalCode	Total.TotalAnnotation = '3.7.1'

This condition states to place the Pay Now total in the third section, 7th in the sequence of totals, and to use bold font.

The screen print doesn't include PrimaryFlag, for brevity. If you add a row, make sure you add PrimaryFlag under Total.TotalAnnotation. Set it to false for all rows except Pay Now. Set it true for Pay Now.

7. In the Conditional Actions area, click **Add Row**, set the values, then click **Save and Close**:

If This Condition Is True	Then Do These Actions
'TOTAL_COST' == Total.TotalCode	Total.TotalAnnotation = '3.8.0' Total.PrimaryFlag =false
'TOTAL_MARGIN' == Total.TotalCode	Total.TotalAnnotation = '4.9.0' Total.PrimaryFlag =false

8. On the Manage Algorithms page, click the **row** that contains the algorithm you just edited, then click **Actions > Publish**.
9. Verify that the status on the row changes to Published.

Set Up the Identify Usage Totals Pricing Algorithm

Set up the algorithm that identifies the usage totals you want to display in the charge breakdown for the sales order:

1. On the Manage Algorithms page, click **Query By Example**, then search the Name column for Identify Usage Totals.
2. Click the **row** that contains the highest value in the Version column, then click **Actions > Create Version**.
3. On the Edit Algorithm page, click the **Create Usage Total Entities** step.
4. In the Conditional Actions area, add actions for your new totals. Add them to the row that contains the `'ORA_SALES_ORDER' == Header.TotalUsageCode` condition.

   Add the bold code:

If This Condition Is True	Then Do These Actions
'ORA_SALES_ORDER' == Header.TotalUsageCode	UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_LIST_PRICE', HeaderId:Header.HeaderId]) UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_DISCOUNT', HeaderId:Header.HeaderId])

If This Condition Is True	Then Do These Actions
	UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_NET_PRICE', HeaderId:Header.HeaderId]) UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_SHIP_CHARGE', HeaderId:Header.HeaderId]) UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_TAX', HeaderId:Header.HeaderId]) UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_CREDIT', HeaderId:Header.HeaderId]) UsageTotal.insert([TotalCodeForUsage:'QP_TOTAL_PAY_NOW', HeaderId:Header.HeaderId]) **UsageTotal.insert([TotalCodeForUsage: 'TOTAL_COST', HeaderId:Header.HeaderId])** **UsageTotal.insert([TotalCodeForUsage:'TOTAL_MARGIN', HeaderId:Header.HeaderId])**

5. Click **Save and Close**.
6. On the Manage Algorithms page, click the **row** that contains the algorithm you just edited, then click **Actions > Publish**.

Set Up the Compute Simple Totals Pricing Algorithm

Set up the algorithm that identifies the usage totals you want to display in the charge breakdown for the sales order:

1. On the Manage Algorithms page, click **Query By Example**, then search the Name column for Compute Simple Totals.
2. Click the row that contains the highest value in the Version column, then click **Actions > Create Version**.

   This step iterates the ChargeComponents for each order header.
3. Declare your variables:
   - Scroll down to the Local Variables area.
   - In the Local Variables area, add variables. Click **Add Row** to add each variable:

Variable Name	Default
MarginTotalCode	'TOTAL_MARGIN' You must include the single quotation marks.
RunningMarginSum	0.0
MarginTotal	Leave empty

4. Add code to calculate the total:
    - In the Steps area, click the **Compute Total** step.
    - Add code in the Group Each Row Actions area.

```
//TotalMargin
totalUsage = TotalUsage.locate([TotalCodeForUsage: MarginTotalCode])
if ('ORDER' == Line.LineCategoryCode && Charge.ChargeAppliesTo in ['PRICE', 'SHIPPING'] &&
 ChargeComp.PriceElementCode == 'QP_MARGIN' && null != totalUsage) {
 if (null == MarginTotal) {
 MarginTotal = Total.locate([TotalCode: MarginTotalCode, HeaderId: Header.HeaderId]) TotalDef =
 getTotalDefinitionFromInternal(TotalDefinition, MarginTotalCode)
 if (TotalDef ? .TransactionEnabledFlag) {
 TotalId = getNextId()
 MarginTotal = Total.insert([TotalId: TotalId, TotalCode: MarginTotalCode, EstimatedFlag:
 IsEstimated, CurrencyCode: Header.AppliedCurrencyCode, HeaderId: Header.HeaderId, ParentEntityId:
 Header.HeaderId, ParentEntityCode: 'ORA_HEADER'])

 TotalAmount = MarginTotal.createDataObject('TotalAmount')
 TotalAmount.CurrencyCode = MarginTotal.CurrencyCode
 TotalAmount.Value = 0.0
 }
 }
 if (null != MarginTotal) {
 RunningMarginSum += ChargeComp.HeaderCurrencyExtendedAmount.Value
 TotalComp.insert([TotalComponentId: getNextId(), TotalId: MarginTotal.TotalId, ChargeComponentId:
 ChargeComp.ChargeComponentId])
 }
```

```
}
```

Here's an explanation:

```
//TotalMargin
  totalUsage = TotalUsage.locate([TotalCodeForUsage:MarginTotalCode])
  if ('ORDER' == Line.LineCategoryCode && Charge.ChargeAppliesTo in
       ['PRICE', 'SHIPPING'] && ChargeComp.PriceElementCode ==
       'QP_MARGIN' && null != totalUsage ) {
    if (null == MarginTotal) {MarginTotal = Total.locate([TotalCode:
       MarginTotalCode, HeaderId: Header.HeaderId])TotalDef =
       getTotalDefinitionFromInternal( TotalDefinition,
       MarginTotalCode)
      if (TotalDef?.TransactionEnabledFlag) {TotalId = getNextId()
            MarginTotal = Total.insert([TotalId: TotalId, TotalCode:
            MarginTotalCode,  EstimatedFlag: IsEstimated,
            CurrencyCode: Header.AppliedCurrencyCode, HeaderId:
            Header.HeaderId, ParentEntityId: Header.HeaderId,
            ParentEntityCode:'ORA_HEADER'])

            TotalAmount = MarginTotal.createDataObject('TotalAmount')
            TotalAmount.CurrencyCode = MarginTotal.CurrencyCode
            TotalAmount.Value = 0.0
      }
    }
    if (null != MarginTotal) {
      RunningMarginSum +=
      ChargeComp.HeaderCurrencyExtendedAmount.Value
      TotalComp.insert([TotalComponentId: getNextId(), TotalId:
      MarginTotal.TotalId, ChargeComponentId:
      ChargeComp.ChargeComponentId])
    }
  }
```





Legend

Condition that sets up calculation.

Calculates total. Stores result in variable with suffix *Sum*.

- Add code in the Group Last Row Actions area. This code locates the new Total and assigns the calculated Sum value.

```
if (null == MarginTotal) MarginTotal = Total.locate([TotalCode: MarginTotalCode, HeaderId:
 Header.HeaderId])

if (MarginTotal != null) {
 MarginTotal.TotalAmount.Value = RunningMarginSum
}
```

5. Add code to handle a situation where the total is empty:
    - In the Steps area, click the **Write Empty Totals** step.
    - Add code in the Default Action area.

```
//Empty TotalPayNow
TotalCode = 'TOTAL_MARGIN'
CheckTotal = Total.locate([TotalCode: TotalCode])
totalUsage = TotalUsage.locate([TotalCodeForUsage: TotalCode])
TotalDef = getTotalDefinitionFromInternal(TotalDefinition, TotalCode)
if (null != totalUsage && null == CheckTotal &&
 TotalDef ? .ReturnNullTotalFlag && TotalDef ? .TransactionEnabledFlag) {
 CheckTotal = Total.insert([TotalCode: TotalCode, TotalId: getNextId(),
 CurrencyCode: Header.AppliedCurrencyCode, EstimatedFlag: IsEstimated
 ])
 TotalAmount = CheckTotal.createDataObject('TotalAmount')
 TotalAmount.CurrencyCode = CheckTotal.CurrencyCode
 TotalAmount.Value = 0.0
}
```

6. Click **Save and Close**.
7. On the Manage Algorithms page, click the **row** that contains the algorithm you just edited, then click **Actions > Publish**.

Test Your Setup

1. Open another browser application and sign into Order Management.
2. Create a sales order and add an order line.
3. Click **Total** at the top of the order, then verify the charge breakdown includes your new totals and correctly calculates the values.

Related Topics

- Manage Order Line's Price Details
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Pricing Algorithms
- Manage Pricing Elements

Include Return Amounts in Pricing Totals

See how you can include amounts from a return line in the pricing totals for each sales order.

A sales order in Order Management includes one-time charges for various totals, such as Total List Price, Total Net Price, Total Tax, and Pay Now.

For example:

Attribute	Value
Total List Price	237,000.00

Attribute	Value
Discount	0.00
Total Net Price	237,000.00
Shipping	0.00
Total Tax	0.00
Total Credit	0.00
Pay Now	237,000.00

A return order has the Total Credit and Pay Now for the return lines. It doesn't have Total List Price, Total Net Price, or Total Tax because a sales order can be mixed.

A mixed sales order has outbound lines and return lines. Including totals for a mixed order can be confusing, so the shipping document for a sales order that has only return lines doesn't have totals for taxes and other amounts.

For example, here are the totals for a sales order that only has outbound lines:

Attribute	Value
Total List Price	1,850.00
Discount	0.00
Total Net Price	1,850.00
Shipping	0.00
Total Tax	222.00
Total Credit	0.00
Pay Now	2,072.00

here are the totals for a sales order that only has return lines:

Attribute	Value
Total List Price	0.00

Attribute	Value
Discount	0.00
Total Net Price	0.00
Shipping	0.00
Total Tax	0.00
Total Credit	2,072.00
Pay Now	2,072.00

This example will show you how to set up pricing so your documents include totals for return lines, such as total tax:

1. Go to the Pricing Administration work area.
2. Click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Query by Example**, then query for the value:

Attribute	Value
Name	Compute Simple Totals

   Pricing uses this algorithm to determine what totals to calculate for the sales order.
4. Click **Actions > Create Version**.
   Assume it creates version 2.
5. In the row that has version 2, click the **link** in the Name column.
6. On the Edit Algorithm page, set the value:

Attribute	Value
Description	Extend this algorithm so we can include tax totals for return orders.

7. In the Steps area, click the **step** that has the value:

Attribute	Value
Name	Compute Total

   It is usually step 2.

8. In the Step Details area, scroll down, then, in the Group Each Row Action area, locate this chunk of code.

```
//TotalTax
totalUsage = TotalUsage.locate([TotalCodeForUsage:TaxTotalCode])
if ('ORDER' == Line.LineCategoryCode && Charge.ChargeAppliesTo in ['PRICE', 'SHIPPING'] &&
 ChargeComp.PriceElementUsageCode in ['EXCLUSIVE_TAX', 'INCLUSIVE_TAX'] && null != totalUsage ) {
 if (null == TaxTotal) {
```

9. Modify that code.

```
//TotalTax
totalUsage = TotalUsage.locate([TotalCodeForUsage:TaxTotalCode])
if (Charge.ChargeAppliesTo in ['PRICE', 'SHIPPING'] && ChargeComp.PriceElementUsageCode in
 ['EXCLUSIVE_TAX', 'INCLUSIVE_TAX'] && null != totalUsage ) {
 if (null == TaxTotal) {
```

   Note

   - You remove `'ORDER' == Line.LineCategoryCode &&`. The LineCategoryCode attribute specifies whether the line is an outbound line or a return line. ORDER means it's outbound, and RETURN means it's a return.
   - `'ORDER' == Line.LineCategoryCode` specifies to process only outbound lines. So, removing this phrase from the condition means the algorithm will process outbound lines and return lines.
   - This statement uses the PriceElementUsageCode charge component to get the exclusive tax and the inclusive tax. It specifies to add charges for the return lines to the total tax.

10. Click **Save and Close**.
11. Publish version 2 of the algorithm.
12. Open another browser, sign into Oracle Applications with the privileges that you need to manage sales orders, go to the Order Management work area, create a return order that only has return lines, then click **Save**.
13. Verify that the totals include amounts for the return charge.

    For example, notice the value for the tax isn't zero:

Attribute	Value
Total List Price	0.00
Discount	0.00
Total Net Price	0.00
Shipping	0.00
Total Tax	222.00
Total Credit	2,072.00
Pay Now	2,072.00

Manage Order Line's Price Details

Use the pricing results presentation to determine how Order Management displays each price element in the charge breakdown on the order line.

- You can also set up the Sales service mapping to specify some aspects of the display.
- Oracle Pricing provides the details that Order Management displays in different charge breakdowns, such as the breakdown on the order line or order header.
- Order Management supports only one recurring charge for each order line. You can't add more than one recurring charge on the same line.

Summary of the Set Up

1. Examine the current behavior.
2. Manage the pricing results presentation.
3. Create a sandbox.
4. Set up the service mapping.

Assume you must display the Cost of Goods Sold price component in the charge breakdown that Order Management displays in the Amount Sale Price dialog on the order line.

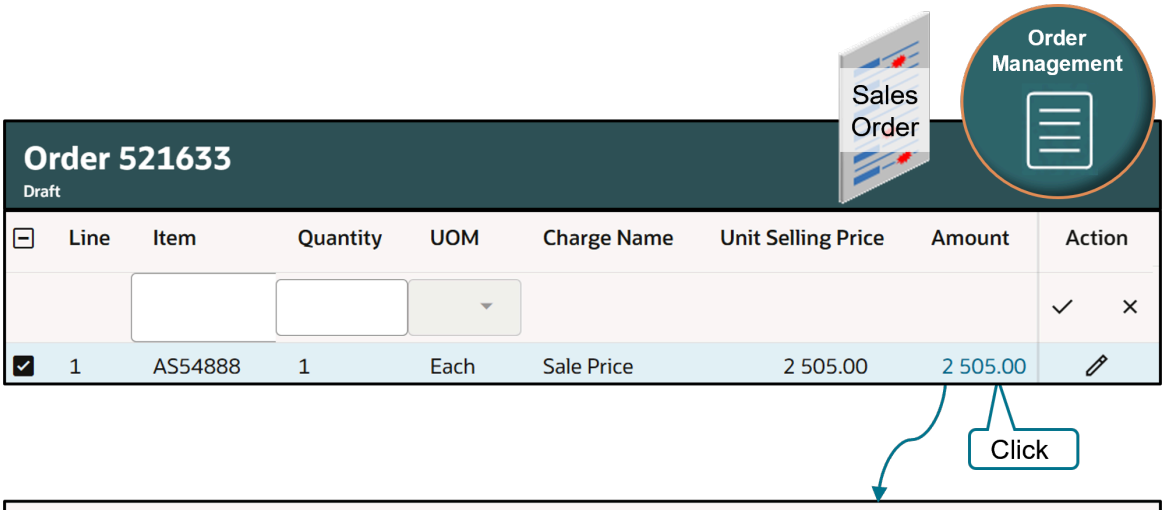
This topic uses example values. You might need different values, depending on your business requirements.

Examine the Current Behavior

1. Make sure you have the privileges that you need to manage sales orders.
2.
3. Go to **Home > Order Management**, create a sales order, then add an item on the order line:

Item	Your Price
AS54888, Desktop Computer	2,505

4. Click **2,505** on the order line.
   Order Management displays the charge breakdown. Assume you must add Cost of Goods Sold to the breakdown:



Charge Breakdown

Price Components	Unit Selling Price	Amount
Base List Price Applied from Corporate Segment Price List	2 505.00	2 505.00
List Price	2 505.00	2 505.00
Your Price	2 505.00	2 505.00
Exclusive Tax (VAT20 20%)	501.00	501.00
Net Price Plus Tax	3 006.00	3 006.00

5. Notice the 3,006 total at the top of the page. Click **Additional Order Details > Order Detail**. You can modify this layout. See *Manage Pricing Totals*.

Manage the Pricing Results Presentation

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Results Presentations task.
2. On the Manage Pricing Results Presentations page, in the Name attribute, enter **%**, then click **Search**.
   The Manage Pricing Results Presentations page displays the predefined pricing results presentations.
3. Examine the pricing results presentations.
   To reduce maintenance, use a predefined pricing results presentation instead of creating a new one.
4. Click the **row** that includes QP_SALES_PRICE_BREAKDOWN in the Name column.
   Order Management comes predefined to use this pricing results presentation to display the Amount Sale Price dialog.

5. In the Price Elements area, choose Selected Price Elements.
6. Click **Actions > Select and Add**.
7. In the Select and Add dialog, in the Element Name attribute, click %, then click **Search**.

   The results display all the elements. Use this dialog to specify the elements that Order Management displays in the Amount Sale Price dialog.
8. Set the values, then click **Search**:

Attribute	Value
Element Type	Cost
Element Name	Cost of Goods Sold

   I the results, click the **row** that includes Cost of Goods Sold in the Element Name column, then click **Apply > OK**.

   The Line dialog in Order Management only displays elements that you add, and only elements that apply for the order line. For example, if the amount on the order line doesn't require rounding, then the dialog won't display a line for the rounding adjustment. To avoid an empty dialog that doesn't contain any price lines, add at least one element that will display in most situations, such as Your Price.
9. Click **Save and Close**.
10. Test your set up:
    - Sign into Order Management, create a sales order, specify a customer, then add an order line.
    - On the order line, click the **link** in the Amount column.

      The charge breakdown displays in the Line dialog.
    - Verify that the Line dialog displays the price elements that you specified on the Manage Pricing Results Presentations page.

Create a Sandbox

You must create a sandbox that you can use to edit the service mapping:

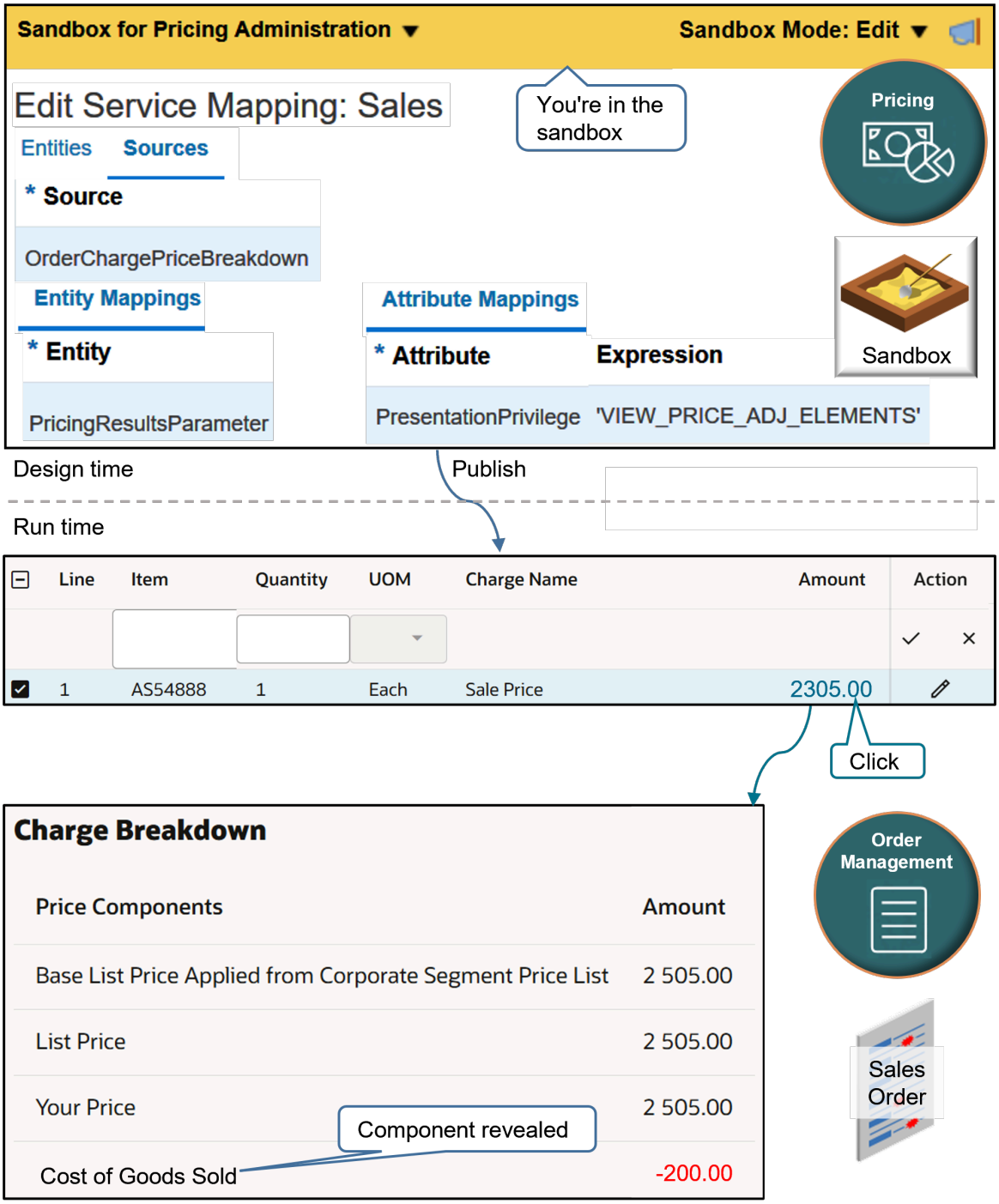
1. Open another browser and sign into Oracle Pricing with administrative privileges.
2. Go to the Sandboxes work area.
3. On the Sandbox page, click **Create Sandbox**.
4. On the Create Sandbox page, set the value:

Attribute	Value
Name	Enter any text. For this example, enter Sandbox for Pricing Administration.

5. Add a check mark to the Manage Service Mappings tool, then click **Create and Enter**.

Edit the Service Mapping

Edit the Sales service mapping so Order Management displays cost and margin in the charge breakdown on the order line:



Try it:

1. Make sure you have the privileges that you need to administer pricing.

2. Go to the Pricing Administration work area.
3. On the Pricing Administration page, click **Tasks**, then, in the Pricing Configuration area, click **Manage Service Mappings**.
4. On the Manage Service Mappings page, click **Sales**.
5. On the Edit Service Mappings page, click **Sources**, then click the **row** that includes OrderChargePriceBreakdown in the Source column.
6. On the Entity Mappings tab, click the **row** that includes PricingResultsParameter in the Entity column.
7. In the Attribute Mappings area, notice the predefined mappings:

Attribute	Expression	Description
PresentationPrivilege	'VIEW_PRICE_ADJ_ELEMENTS'	Specifies to display only price adjustment elements when the Order Entry Specialist views the charge breakdown.
ResultPresentationCode	'QP_SALES_PRICE_BREAKDOWN'	Specifies to use the QP_SALES_PRICE_ BREAKDOWN pricing results presentation when displaying the charge breakdown.

8. Set the value:

Attribute	Expression	Description
PresentationPrivilege	'VIEW_ALL_PRICE_ELEMENTS'	Specifies to display all price adjustment elements, including cost and margin details, when the Order Entry Specialist views the charge breakdown.

9. At the top of the page, click **Sandbox for Pricing Administration > Publish**.

Related Topics

- Manage Pricing Totals
- Overview of Oracle Pricing
- Manage Pricing Elements
- Create and Activate Sandboxes

Manage Pricing Lookups

Oracle Pricing comes predefined with the lookups that it displays in the Pricing Administration work area and in some of the Pricing pages of the Setup and Maintenance work area. You can't delete a predefined lookup, but you can modify some of their attributes.

Summary of the Setup

1. Examine the current behavior.

2. Modify the pricing lookup.
3. Test your set up.

Assume you must add a new value that you can use to specify the pricing strategy for a loss leader. You add this value to the Pricing Strategy Objectives lookup.

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

> **CAUTION:** Modify a lookup only when the modification is critical to meet your business requirements. Most predefined lookups make sense in their predefined context and you shouldn't need to modify them, but if you do modify a predefined lookup, then you must make sure your modification doesn't affect Pricing logic. The example in this topic doesn't modify Pricing logic. It affects only your requirement to document how you use a pricing strategy. Some lookups are required. For example, the Customer Size attribute of a pricing profile is required. You can modify some of these lookups. If you aren't certain whether modifying a lookup will affect Pricing logic, then you must consult with Oracle before you make the modification.

Examine the Current Behavior

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Pricing Strategies**.
3. On the Manage Pricing Strategies page, click **Actions**, and then click **Create**.
4. In the Create Pricing Strategies dialog, click the **Objectives** attribute, and notice that it displays these values:
    - Competitive pricing
    - Profit maximization
    - Revenue maximization
5. Click **Cancel**.

Modify a Pricing Lookup

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Lookups task.
2. On the Manage Pricing Lookups page, in the Search area, set one or more attributes to filter the search results, then click **Search**.

   For this example, set this value:

Attribute	Value
Module	Pricing Administration

3. In the search results, scan the Meaning column for a lookup that describes the attribute you must modify.

   For this example, you modify the Pricing Strategy Objectives lookup.
4. Click the **row** that includes Pricing Strategy Objectives in the Meaning column.
5. In the Lookup Codes area, click **Actions > New**, then set the values:

Attribute	Value
Lookup Code	LOSS_LEADER

Attribute	Value
Display Sequence	4
Meaning	Loss leader
Description	Business objective to stimulate sales for other, more profitable items

6. Modify the values in the Display Sequence column so it uses this sequence:

Lookup Code	Display Sequence
LOSS_LEADER	1
ORA_COMPETITIVE_PRICING	2
ORA_MAXIMIZE_PROFIT	3
ORA_MAXIMIZE_REVENUE	4

7. Click **Save and Close > Done**.

Test Your Set Up

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Pricing Strategies**.
3. On the Manage Pricing Strategies page, click **Actions > Create**.
4. On the Create Pricing Strategies dialog, click the **Objectives** attribute, then verify that it displays these values:
    - Loss leader
    - Competitive pricing
    - Profit maximization
    - Revenue maximization
5. Click **Cancel**.

Related Topics

- Manage Pricing Strategies
- Overview of Oracle Pricing
- Overview of Lookups

Manage Descriptive Flexfields for Pricing

Use the descriptive flexfields that come predefined with Oracle Pricing to store details.

Summary of the Steps

1. Examine the current behavior.
2. Modify the descriptive flexfield.
3. Test your set up.

Assume you must add a new field you can use to describe details about the pricing strategy objective.

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

For details, see *Use Descriptive Flexfields in Your Integration*.

Examine the Current Behavior

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Pricing Strategies**.
3. On the Manage Pricing Strategies page, click **Actions > Create**.
4. On the Create Pricing Strategies dialog, click the **Objectives** attribute, then notice the values it displays.
5. Expand and contract the Additional Information area.

   Notice that this area is empty. You will add a text box you can use to describe the pricing strategy objective.
6. Click **Cancel**.

Modify a Flexfield

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Descriptive Flexfields task.
2. On the Manage Pricing Descriptive Flexfields page, in the Search area, set one or more attributes to filter the search results, then click **Search**.

   For this example, set this value:

Attribute	Value
Module	Pricing Administration

3. In the search results, scan the Name column for a flexfield that describes the pricing entity you must modify.

   For this example, you modify flexfield Pricing Strategy Header DFF.
4. Click the **row** that includes Pricing Strategy Header DFF in the Name column, then click **Actions > Edit**.
5. In the Edit Descriptive Flexfield dialog, click **Actions > Create**.

6. In the Create Segment dialog, set values:

Attribute	Value
Name	LOSS_LEADER
Data Type	Character
Value Set	150 Characters Optional
Prompt	Description of the business objective
Display Type	Text Box
Display Size	150
Display Height	4
Enabled	Contains a check mark
Range Type, Required	Doesn't contain a check mark
Instruction Help Text	Enter details that describe how you intend to meet the objective for this strategy

7. Click **Save and Close > Save and Close**.
8. On the Manage Pricing Descriptive Flexfields page, click the **row** that includes Pricing Strategy Header DFF in the Name column, then click **Deploy Flexfield**.
9. In the dialog that displays, wait until the indicator reaches 100%, then click **OK**.
10. On the Manage Pricing Descriptive Flexfields page, click **Done**.

Test Your Set Up

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Pricing Strategies**.
3. On the Manage Pricing Strategies page, click **Actions > Create**.
4. Expand and contract the Additional Information area.

   Verify that this area includes the flexfield you added, and that you can enter text.
5. Click **Cancel**.

Related Topics

- Manage Pricing Strategies
- Overview of Flexfields
- Overview of Descriptive Flexfields

Pricing Guidelines

Pricing Guidelines

Pricing Guidelines

Create a pricing guideline to control modifications that your users can make to price, net price, margin, and so on. You apply it on an item, user role, customer detail, or time period.

- Set a maximum discount amount, or set a maximum discount percent.
- Set a minimum margin percent.
- Control pricing for a nonconfigured or configured item.
- Make sure the Order Entry Specialist sets values that remain within your company profitability policy while negotiating a price with the customer.
- Reference a pricing matrix so you can leverage the conditions, constraining values, and violation type that the matrix specifies.

Pricing can validate each sales order against a pricing guideline at one of these predefined events:

- Save
- Customer Update
- Reprice
- Validate
- Submit

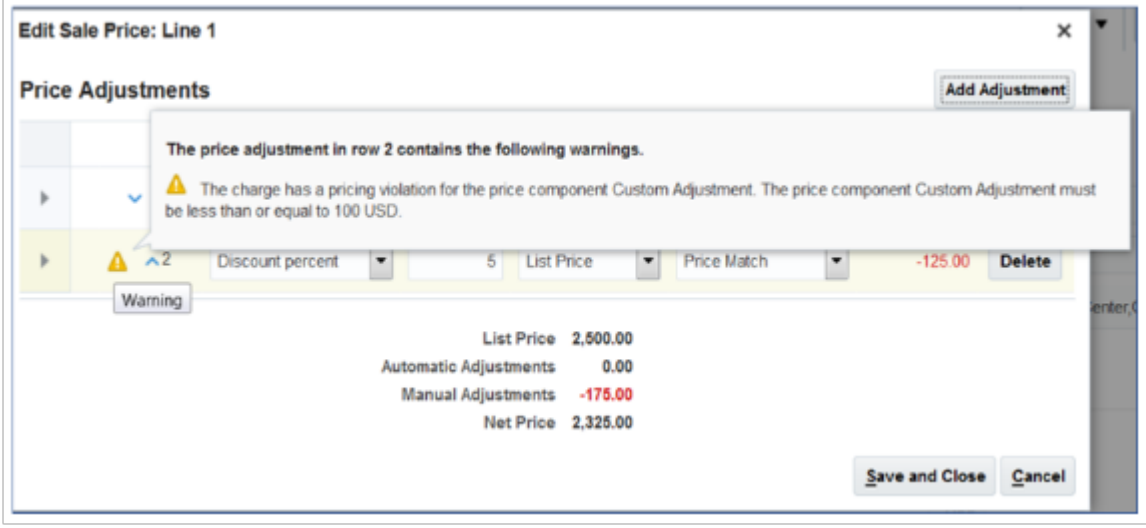
Note

- Pricing validates the entire transaction.
- Pricing validates only the order line for a line pricing event or for a manual price adjustment.
- Pricing validates against each charge component. For example, assume you create a guideline that disallows a manual price adjustment that exceeds 10%. Assume the Order Entry Specialist applies adjustments:
    - One adjustment for 9%
    - One adjustment for 9%
    - One adjustment for 8%

  Pricing individually evaluates each adjustment. It doesn't evaluate the sum of adjustments, such as the entire 27%. None of the adjustments exceeds 10%, so Pricing accepts them.
- If Pricing determines that a sales order exceeds a guideline, then it displays a warning. The Order Entry Specialist can submit a sales order that contains a warning.

How it Works

1. An Order Entry Specialist does an action in Order Management that requires pricing, such as editing the sale price.
2. Order Management sends a request to Pricing to price the transaction and validate the sales price.
3. Pricing runs a pricing process to validate the sales price. It runs the pricing process according to the assignments that the Manage Pricing Process Assignments page specifies.
4. Pricing runs a pricing algorithm that evaluates the pricing guidelines that you created for the pricing strategy when it prices the transaction.
5. Pricing sends violations it finds for the order line charges for each charge component to Order Management.
6. Order Management displays each violation as a warning or error, depending on how you set up the pricing guideline:



You can also set up a rule or constraint to control the sales order when a pricing violation happens:

- Approval rule. Require the Order Entry Specialist to get approval before submitting a sales order that contains a pricing violation. For details, see *Use Visual Information Builder*.
- Processing constraint. Prevent the Order Entry Specialist from submitting a sales order that contains a pricing violation. For details, see *Manage Processing Constraints*.

Related Topics

- Manage Pricing Guidelines
- Manage Pricing Strategies
- How Profiles, Segments, and Strategies Work Together
- Use Visual Information Builder
- Manage Processing Constraints

Manage Pricing Guidelines

In this example, create a pricing guideline that controls the pricing adjustment the Order Entry Specialist can make on the AS54111 Standard Desktop item.

- If the manual price adjustment exceeds 20% of the list price, then display a warning, and allow the Order Entry Specialist to submit the sales order.
- If the manual price adjustment exceeds 40% of the list price, then display an error.
- Make sure the net price is 80% or more of the list price.

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

Summary of the Steps

1. Add the control for the manual price adjustment.
2. Add the control for Your Price.
3. Add the pricing guideline to a pricing strategy.
4. Test your set up.

Add the Control for the Manual Price Adjustment

1. Go to **Home > Order Management > Pricing Administration > Manage Guidelines**.
2. Click **Actions > Create**.
3. In the Create Guideline dialog, set the values, then click **Save and Edit**:

Attribute	Value
Name	My guideline
Business Unit	Vision Operations

4. Add the control for the manual price adjustment. In the Charge Guideline Components area, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Component	Custom Adjustment Specify the component where Pricing must apply the pricing guideline. In this example, you're creating a pricing guideline against the Custom Adjustment charge component that affects the adjustment the Order Entry Specialist can make. The Used in Guidelines attribute of the Manage Price Elements page determines the price element to use. At runtime, the evaluation happens against the charge component because Pricing returns charge components in the runtime environment.

Attribute	Value
	You can also apply a pricing guideline that controls the changes that you can make on other components, such as Your Price, List Price, and so on. If the list is empty or it doesn't have the component you need, do this: a. Go to the Setup and Maintenance work area, then go to the Manage Price Elements task. b. On the Manage Price Elements page, click **Search**. c. In the search results, click the **row** that contains the element you need. d. In the row you clicked, add a check mark to the **Used in Pricing Guidelines** option.
Calculation Type	Percent of
Operator	Less than or equal to
Calculation Component	List Price

5. In the search results, click **Actions > Add Row**, then set the values. Use this area to specify where to apply the pricing guideline at a detailed level:

Attribute	Value
Item Level	Item Select All Items to apply the guideline to all items in the pricing strategy that this guideline references.
Name	AS54111 In this example, you apply the pricing guideline on the AS54111 Standard Desktop item.
Line Type	Buy In some implementations, you can apply the pricing guideline only to Buy order lines.
Applies To	Price Note that Applies To can also apply to shipping charges.
Pricing UOM	Each
Price Type	All

Attribute	Value
Charge Type	Sale
Charge Subtype	Price

6. Click **Create Charge Guideline Rule Matrix**.
7. In the Charge Guideline Rules area, click **Actions > Add Row**, then set the values. This area sets the rule to apply for the pricing guideline:

Attribute	Value
Constraining Value	20
Violation Type	Warning

To add more conditions, you add more rows to the matrix.

8. At the top of the page, click **Save**.

Notice that the Statement attribute now contains a value. If the Order Entry Specialist makes an adjustment that violates the guideline, then Order Management displays a similar message in the Price Adjustments dialog.

Add the Control for Your Price

1. Create the pricing guideline for the manual price adjustment. In the Charge Guideline Components area, click **Actions > Add Row**, then set the values:

Attribute	Value
Component	Your Price
Calculation Type	Percent of
Operator	Greater than or equal to
Calculation Component	List Price

2. In the search results, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Item Level	Item
Name	AS54111
Line Type	Buy
Applies To	Price
Pricing UOM	Each
Price Type	All
Charge Type	Sale
Charge Subtype	Price

3. Click **Create Charge Guideline Rule Matrix**.
4. In the Charge Guideline Rules area, click **Actions > Add Row**, set values, then click **Save**:

Attribute	Value
Constraining Value	80
Violation Type	Warning

5. Click **Access Set**.
6. Click **Actions > Add Row**, set the value, then click **Save**:

Attribute	Value
Set Code	COMMON

7. Click **Approve**.

Add Your Pricing Guideline to a Pricing Strategy

1. Click **Tasks > Manage Pricing Strategies**.
2. On the Manage Pricing Strategies page, search for, then open Corporate Pricing Strategy for editing.
3. On the Edit Pricing Strategies page, click **Guidelines**.
4. In the Charge Guidelines area, click **Actions > Select and Add**.
5. In the Select and Add dialog, set the value, then click **Search**:

Attribute	Value
Name	Corporate Pricing Strategy

6. In the search results, click the **row**, click **Apply**, then click **OK**.
7. In the Charge Guidelines area, set the Start Date for your guideline.
8. At the top of the page, 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, then create a sales order:

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

3. Add an order line:

Attribute	Value
Item	AS54111 Standard Desktop
Quantity	1
UOM	Each

Notice the order line displays a Sale Price of 2,000.00 and a Recycling Fee of 10.00.

4. In the AS54111 Standard Desktop row, click the **value** in the Amount attribute.

5. Set these values:

Attribute	Value
Type	Discount Amount
Amount	500
Reason	Price Match

6. Notice that the page displays these values:

Attribute	Value
List Price	2,000
Automatic Adjustments	0
Manual Adjustments	500
Net Price	1,500

Recall that you set up these controls:

- Make sure the manual price adjustment doesn't exceed 20% of the list price.
- Make sure the net price is 80% or more of the list price.

Your manual adjustment violates each of the controls:

- The $500 manual adjustment is 25% of the $2,000 list price.
- The net price is less than 80% of the list price. $1,600 is 80% of the list price.

7. At the top of the page, click **Save**.

Order Management calls Pricing to validate the sales price. Pricing evaluates the pricing guidelines that reference the pricing strategy for the transaction, then applies the guidelines. In this example, it finds two violations, then sends the results to Order Management.

8. Notice that Order Management displays a Warning icon on the far left side of the order line.
9. Click the **warning** on the order line.
10. Verify that Order Management displays a warning.

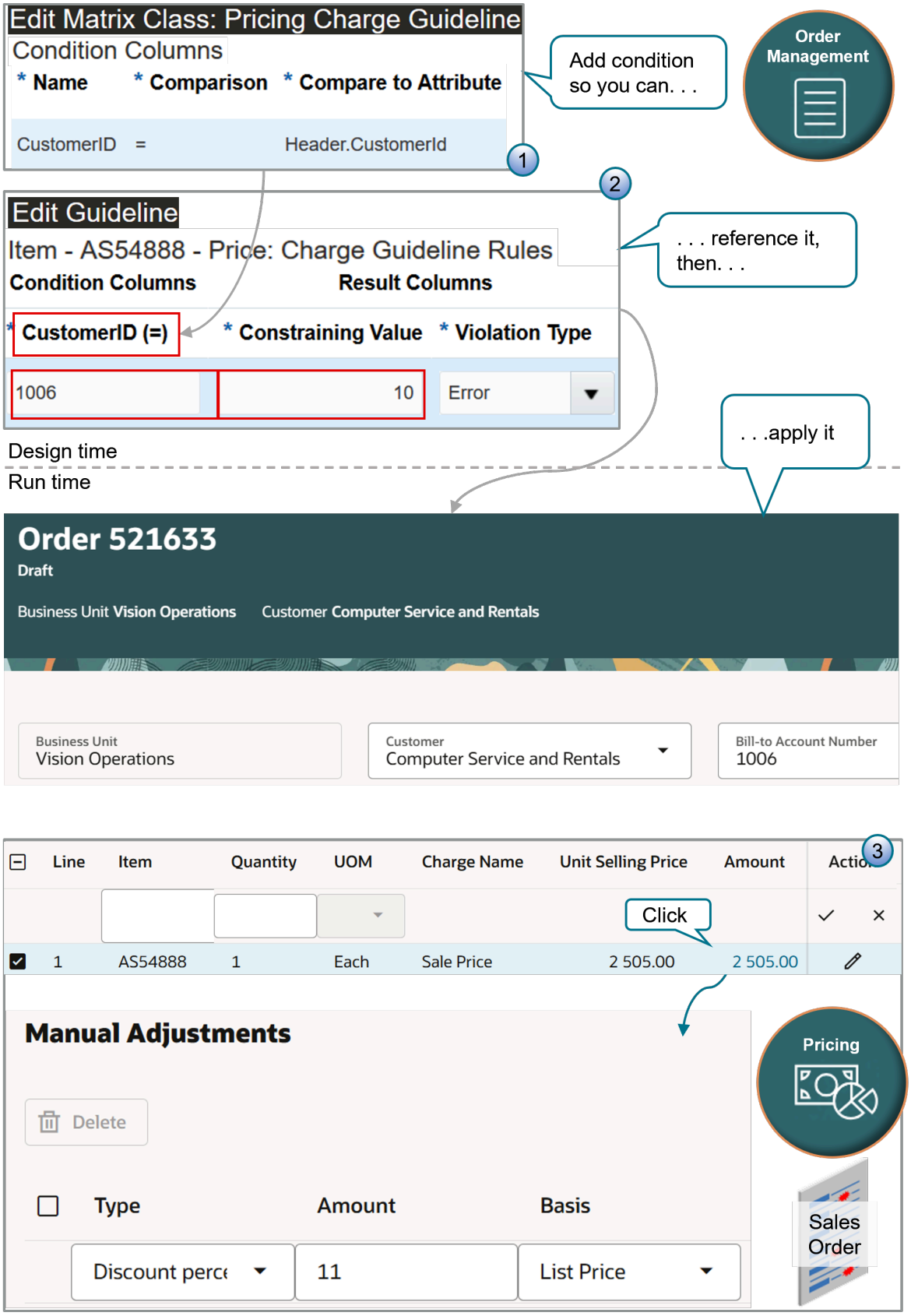
```
The charge has a pricing violation for the charge component Your Price. The charge component Your Price
must be greater than or equal to 1,600 USD (value 80% of the charge component List Price).
```

Related Topics

- Pricing Guidelines
- Manage Pricing Strategies
- How Profiles, Segments, and Strategies Work Together

Create Your Own Conditions for Pricing Guidelines

Modify a matrix class so you can add your own condition to a pricing guideline.


Edit Matrix Class: Pricing Charge Guideline
Condition Columns
* Name
* Comparison
* Compare to Attribute
CustomerID
=
Header.CustomerId
1
Add condition so you can. . .
Order Management
2
Edit Guideline
Item - AS54888 - Price: Charge Guideline Rules
Condition Columns
Result Columns
* CustomerID (=)
* Constraining Value
* Violation Type
1006
10
Error
. . . reference it, then. . .
. . .apply it
Design time
Run time
Order 521633
Draft
Business Unit Vision Operations
Customer Computer Service and Rentals
Business Unit
Vision Operations
Customer
Computer Service and Rentals
Bill-to Account Number
1006
3
Line
Item
Quantity
UOM
Charge Name
Unit Selling Price
Amount
Click
1
AS54888
1
Each
Sale Price
2 505.00
2 505.00
Manual Adjustments
Delete
Type
Amount
Basis
Discount perce
11
List Price
Pricing
Sales Order

Note:

1. Edit the matrix class to specify the condition that you use in the pricing charge guideline. You can specify the attribute that you will reference in the guideline. For example, specify the CustomerId attribute on the order header.
2. Create a pricing guideline that makes sure you can't apply a manual price adjustment that exceeds 10% of the list price. You specify the item, customer, constraining value, and type of violation that Pricing will apply. For example, 1006 is the CustomerID for Computer Service and Rentals.
3. At run time, Pricing applies the pricing guideline when you make a manual price adjustment. If the adjustment exceeds the guideline, then Order Management prevents you from submitting the order and displays a message instead. In this example, the adjustment amount on the order line is 11, it exceeds the constraining value of 10 on the pricing guideline, so Order Management will prevent the submit.

Assume you need a pricing guideline that limits the percent discount that the Order Entry Specialist can set for the AS54888 Desktop Computer to no more 10% for your Computer Services and Rentals customer. You will create this rule:

- If the discount percent is less than or equal to 10% of list price, then allow the Order Entry Specialist to submit the sales order.
- If the discount percent is more than 10% of list price, then don't allow Order Entry Specialist to submit the sales order. Display an error message instead.

Summary of the Set Up

1. Modify the matrix class.
2. Create your pricing guideline.
3. Test your setup.

Modify the Matrix Class

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click the value:

Attribute	Value
Name	Pricing Charge Guideline

3. On the Edit Matrix Class page, in the Condition Columns area, click **Actions > Add Row**, set the values, then click **Save and Close**:

Attribute	Value
Name	CustomerID
Comparison	=
Compare to Attribute	Header. CustomerId
Source Code Name	CustomerID

Attribute	Value
Allow Null	Doesn't contain a check mark
Domain	Leave empty

Create Your Pricing Guideline

1. Click **Tasks > Manage Guidelines**.
2. On the Manage Guidelines page, click **Actions > Create**, set the values in the dialog that displays, then click **Save and Edit**:

Attribute	Value
Name	Pricing Guideline for Customer Discount
Business Unit	Vision Operations

3. On the Edit Guidelines page, in the Charge Guideline Components area, click **Actions > Add Row**, then set the values:

Attribute	Value
Component	Manual Adjustment
Calculation Type	Discount Percent
Operator	Less Than or Equal To This guideline specifies to make sure the manual price adjustment is less than or equal to the list price. If it isn't, then display an error.
Calculation Component	List Price
Currency	Leave empty

4. In the Manual Adjustment Discount Percent area, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Item Level	Item You can also set this value to All Items and Pricing will apply the guideline to any item that your customer orders.
Name	AS54888
Description	Standard Desktop
Line Type	Buy
Applies To	Price
Pricing UOM	Each
Price Type Charge Type Charge Subtype	All

5. Click **Create Charge Guideline Rule Matrix**.
6. In the dialog that displays, add a check mark to the **CustomerID** option, then click **OK**.
7. In the Charge Guideline Rules area, click **Actions > Add Row**, set values, then click **Save**:

Attribute	Value
Customer ID	1006 Here's an easy way to get the Customer ID. a. Go to the Order Management work area, create a sales order, then select the Customer attribute. b. Search the Name attribute for your customer. c. In the search results, notice the value in the Registry ID attribute, such as 1006.
Constraining Value	10
Violation Type	Error

8. Click **Access Sets**, click **Actions > Add Row**, set the values, then click **Save and Close**:

Attribute	Value
Set Code	Common
Set Name	Common Set

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
Bill-to Account	1006

2. Search for and add the AS54888 item on the order line.
3. On the order line, click the **value** in the Amount column.
4. In the dialog that displays, set the values, click **Save and Close**, then submit:

Attribute	Value
Type	Discount Percent
Amount	11
Basis	List Price
Reason	Select any value

5. Verify that a warning dialog displays a warning.

```
The charge Sale Price has a pricing violation for the price component Your Price. The price component
 Your Price must be less than or equal to 250 USD (the price component List Price with discount %)
```

6. Click **OK**, go back to the order line and change the amount that you set earlier from 11 to 10, then resubmit the order.

7. Verify that you can now successfully submit the sales order.

Related Topics

- Pricing Guidelines
- Manage Pricing Strategies
- How Profiles, Segments, and Strategies Work Together

Messages

Manage Pricing Messages

Use a pricing message to describe a price element.

- Describe a price element, such as Total List Price, Discount, Shipping, Total Tax, and so on, that Order Management displays in the charge breakdown on the New Sales Order page.
- Describe the reason for a sale price violation. For example, Pricing displays the QP_PDP_PRC_TERM_ERR message when a pricing term is missing data.
- Describe how Oracle Pricing identifies the pricing segment or pricing strategy that it uses for a sales order.
- Describe other details, such as the name of an adjustment and the reason why Pricing applied it.

Summary of the Steps

1. Identify predefined tokens.
2. Create a pricing message.

You will create a pricing message that Pricing displays for the Standard Desktop item in the Corporate price list. Pricing then uses the pricing rule to determine the token value, then displays the message at runtime.

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

Identify Predefined Tokens

You can use a predefined token. You must identify the token name so you can reference it in you message. Consider alternatives when you can't find a predefined token that meets your requirements. For details, see *Manage Pricing Message Tokens*.

Identify predefined tokens:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Message Token Attribute Definitions task.
2. On the Manage Pricing Message Token Attribute Definitions page, choose the pricing entity where Pricing must display the message, then click **Search**. For this example, use this value:

Attribute	Value
Pricing Entity	Price List Items

3. In the search results, click **Price List Message**.
4. On the Edit Pricing Message Token Attribute Definition page, click **Refresh Token Attributes**.
5. Scan the search results for the tokens that look like might meet your requirements, and then note the values in the Token Name column.
   For this example, you must display the name of the price list and the name of the item, so note these token names:
   - PRICE_LIST_NAME
   - ITEM_NAME
6. Click **Cancel > Done > Done > Save and Close**.

Create the Pricing Message

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Messages task.
2. On the Manage Pricing Messages page, choose the pricing entity where Pricing must display the message, then click **Search**. For this example, set this value:

Attribute	Value
Pricing Entity	Price List Items

3. Examine the predefined pricing messages.
   To reduce maintenance, use a predefined pricing message instead of creating a new one.
4. If you can't locate a pricing message that meets your requirements, then click **Actions > Add Row**, then set the values:

Attribute	Description
Name	Enter text that Pricing can use to reference this pricing message: - Enter only alphabetic text. - Use all capital letters. - Use an underscore to separate each word.
Description	Enter text that describes the purpose of the pricing message. The description is intended only to help you administer and manage pricing messages. Pricing doesn't display it elsewhere.
Message Text	Enter the text that Pricing will display. Enclose each token that you use with curly brackets. For this example, enter `Item {ITEM_NAME} in price list {PRICE_LIST_NAME}is discontinued.`
Pricing Entity	Choose the pricing entity where Pricing must display the message.

Attribute	Description

5. Click **Save**.
6. Modify the pricing rule so it can determine the token value.

Related Topics

- Manage Pricing Message Tokens
- Overview of Oracle Pricing

Manage Pricing Message Tokens

Create a token that specifies the attributes that Oracle Pricing uses to determine the token value for a pricing entity. Use the token in a pricing message.

Pricing uses the token to get and display the token value at runtime.

A token is placeholder for variable content, such as text or a number. An Oracle application replaces the token with a value at runtime. Consider this QP_CL_CHRG_BATCH_ERR message.

```
You can't import the charge for the cost list because dates overlap across rows for the item that has the
{CHARGE_DEF} charge definition.
```

where

- CHARGE_DEF is the name of the token.
- Consultation is an example value that CHARGE_DEF might contain.

So, the runtime value might be:

```
You can't import the charge for the cost list because dates overlap across rows for the item that has the
Consultation charge definition.
```

For another example, assume you select the CurrencyCode attribute and the Price List pricing entity, and you use this attribute as a token in a pricing message. At runtime, Pricing returns the value of the CurrencyCode attribute in the pricing message, then displays it in the charge breakdown.

You will create a token that Pricing uses when it displays a message for the Standard Desktop item in the Corporate price list. You specify Corporate and Standard Desktop each as a token value.

Manage pricing message tokens:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Message Token Attribute Definitions task.
2. On the Manage Pricing Message Token Attribute Definitions page, in the Active attribute, click **Yes**, then click **Search**.

   The Manage Pricing Messages page displays the predefined tokens.
3. Click **Detach**, then examine the pricing message tokens.

   To reduce maintenance, use a predefined pricing message token instead of creating a new one.

4. If you can't locate a pricing message token that meets your requirements, then click **Actions > Create**, then set the values:

Attribute	Description
Name	Enter text that Pricing can use to reference this pricing message token: o Enter only alphabetic text. o Use headline capitalization. For this example, enter **`Custom Price List Message`**.
Description	Enter text that describes the purpose of this pricing message. This description is intended only to help you administer and manage pricing messages. Pricing doesn't display it elsewhere. For this example, enter **`This definition contains custom tokens for prices`**.
Pricing Entity	For this example, choose Price List Items.
Active	Add a check mark.
Package Name	Enter **`oracle.apps.scm.pricing.setup.pricingMessages.publicModel.applicationModule`**. You must use this value for all tokens.
Application Module	Enter **`PricingMessageTokenAttrDefnAM`**. You must use this value for all tokens
Application Module Configuration	Enter **`PricingMessageTokenAttrDefnAMShared`**. You must use this value for all tokens.
View Object	Enter the name of the view object where Pricing must display this message. Use this format. **`<entityName>MessageVO`** where o **entityName**. Identifies the name of pricing entity you set in the Pricing Entity attribute of this token. For this example, Pricing must display this message on a price list, so enter **`PriceListChargeMessageVO`**. Here are the values you can use, depending on how you set the Pricing Entity attribute: o CostListChargeMessageVO o ItemChargeGuidelineMessageVO

Attribute	Description
	○ ManualChargeAdjustmentMessageVO
View Criteria	For this example, choose **`PriceListChargeMessageVC`**.

5. Click **Save and Edit**.
6. On the Edit Pricing Message Token Attribute Definition page, in the Pricing Message Token Attributes area, click **Refresh Token Attributes**.

Related Topics

- Manage Pricing Messages
- Overview of Oracle Pricing

6 Administer

Various Set Ups

Select a Technology to Manage Your Pricing Data

Evaluate the different technologies you can use to manage and maintain your pricing data.

Use these technologies to create, update, export, search, or delete data in your price lists:



Entity	C	U	E	D	C	U	S	D	C	U	S	D
Headers	■	■	■						■	■	■	■
Items	■	■	■		■	■	■		■	■	■	■
Charges	■	■	■		■	■	■		■	■	■	■
Tiers	■		■		■	■	■	■	■	■	■	■
Matrices	■	■	■		■	■	■	■	■	■	■	■
Flexfields	■	■	■		■	■	■	■	■	■	■	■

Legend
C. Create
U. Update
E. Export
S. Search
D. Delete

Red Cell. Supported
White Cell. Not Supported

You can also use these technologies to manage your discount lists:

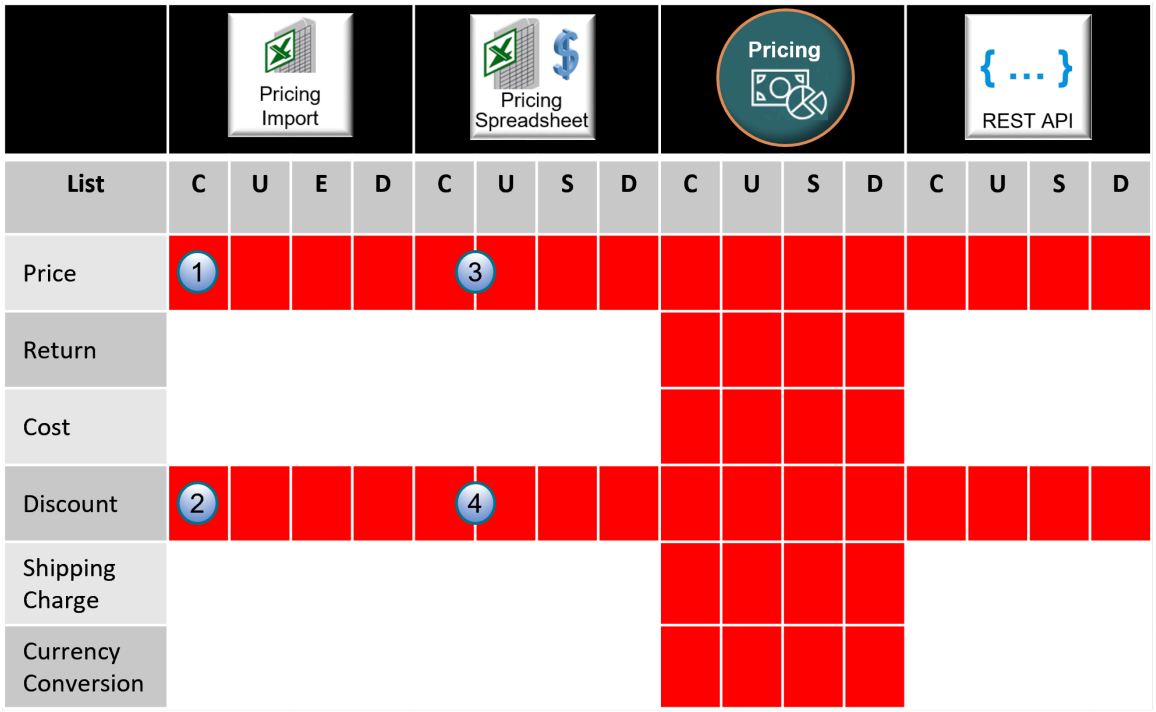


Entity	C	U	E	D	C	U	S	D	C	U	S	D
Headers	■	■							■	■	■	■
Items	■	■			■	■	■		■	■	■	■
Simple Rules	■	■			■	■	■	■	■	■	■	■
Tiered Rules					■	■	■	■	■	■	■	■
Matrix Rules	■	■			■	■	■	■	■	■	■	■
Flexfields	■	■							■	■	■	■

Legend
C. Create
U. Update
E. Export
S. Search
D. Delete

Red Cell. Supported
White Cell. Not Supported

Use these technologies to manage different types of lists:



List	C	U	E	D	C	U	S	D	C	U	S	D	C	U	S	D
Price	1				3											
Return																
Cost																
Discount	2				4											
Shipping Charge																
Currency Conversion																

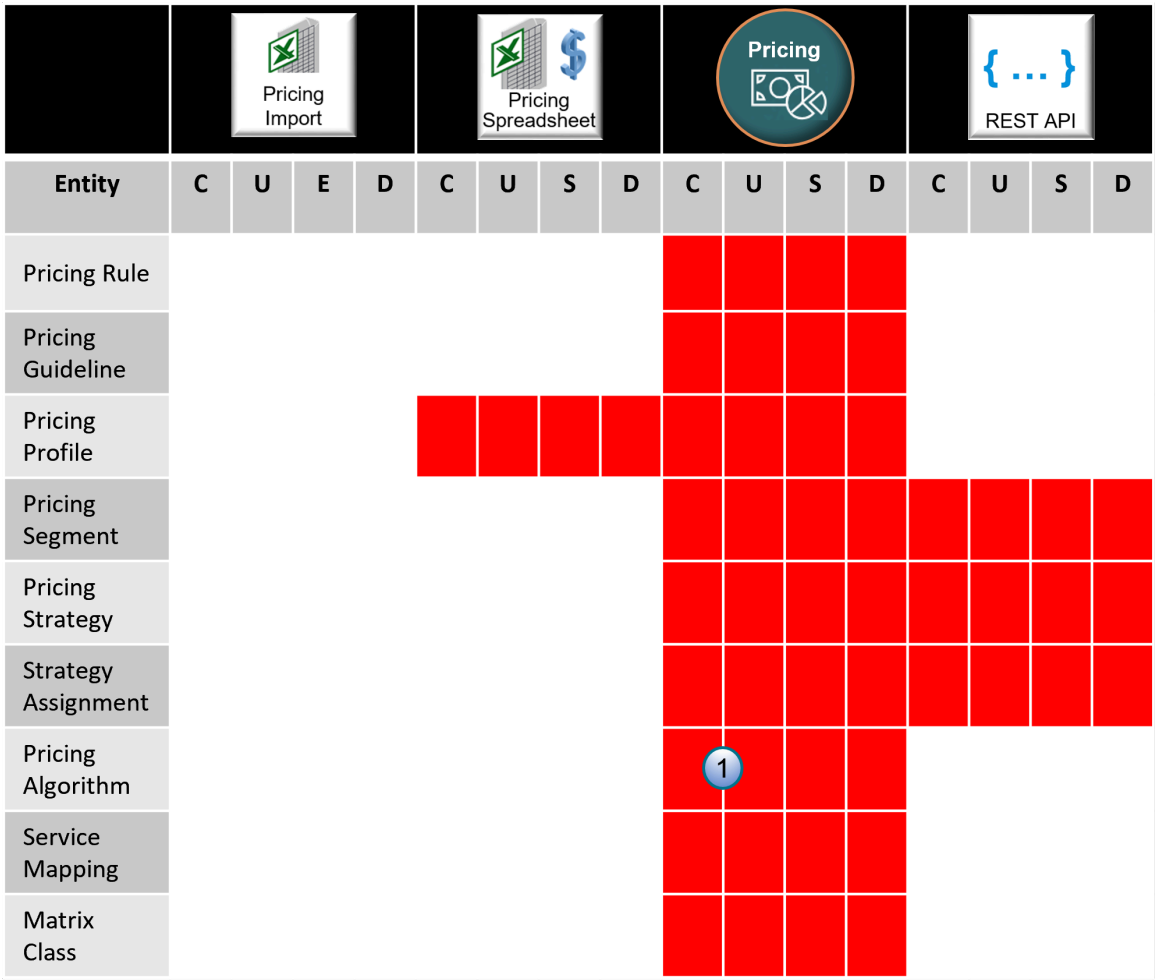
Legend
C. Create
U. Update
E. Export
S. Search
D. Delete

Red Cell. Supported
White Cell. Not Supported

Here's what the numbers mean. You can't use:

1. Pricing import to create the header for a price list.
2. Pricing import to create the header for a discount list.
3. Pricing spreadsheet to create or update a tiered price list, price component, or covered item.
4. Pricing spreadsheet to create or update a tiered discount list.

Use these technologies to manage different types of entities:



Legend

C. Create
U. Update
E. Export
S. Search
D. Delete

Red Cell. Supported
White Cell. Not Supported

Here's what the number means:

1. You can use the Pricing Administration work area to import and export a pricing algorithm.

Consider your requirements, then select the technology that's optimized to meet them:

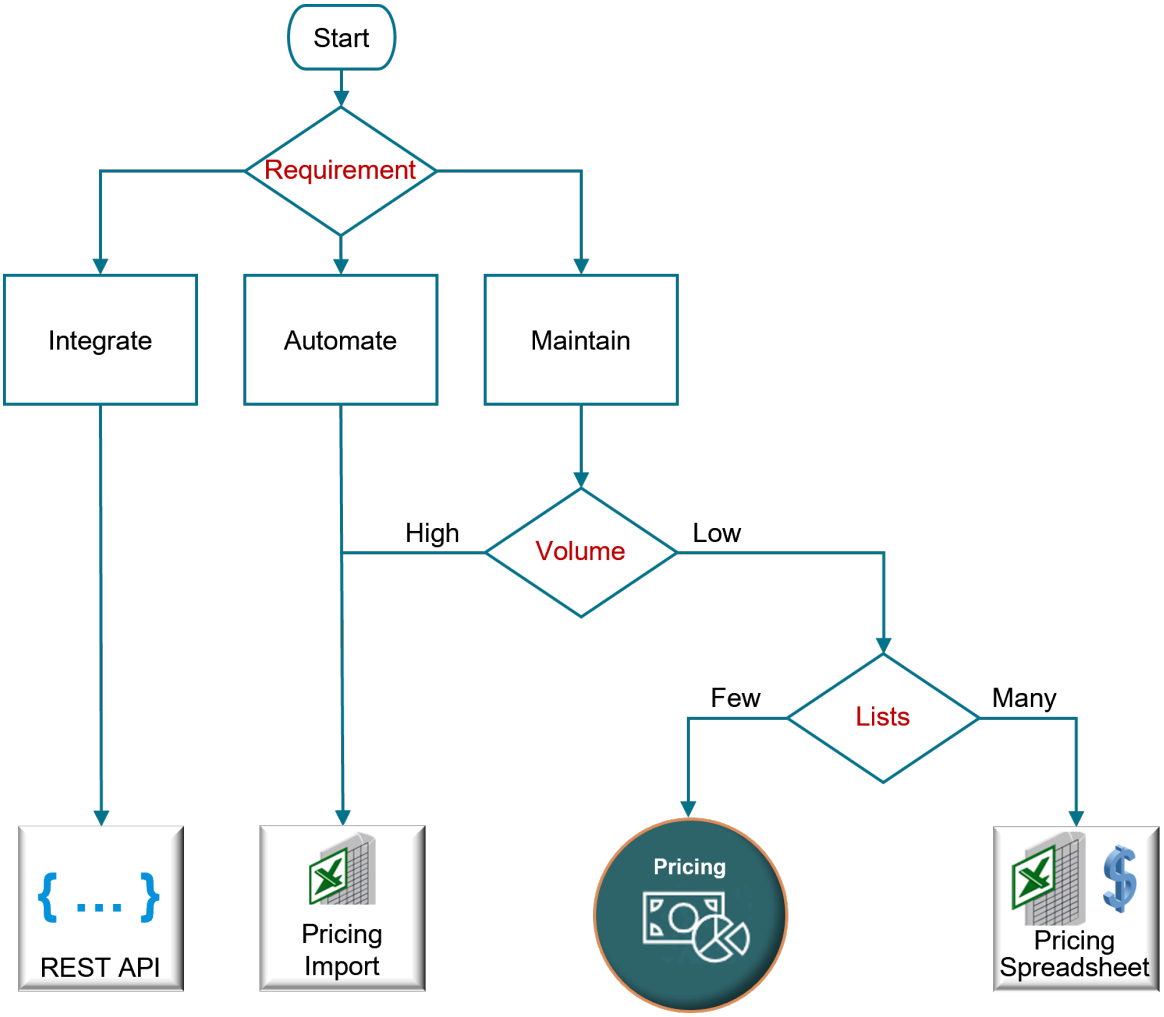
Requirement	Pricing Import	Pricing Spreadsheet	Pricing	REST API
Migrate Data	✓			
Convert Data	✓			
Import a Large Volume of Data	✓			
Import a Small Volume of Data		✓		
Periodically Update Data		✓		
Automate Data Import	✓			✓
Integrate Data				✓
Use a Simple User Interface			✓	

Here are some advantages and disadvantages for each technology:

Technology	Description
Pricing Import Spreadsheet with File-Based Data Import	Use pricing import spreadsheets to import, migrate, or convert your data. They provide superior performance, even when importing a large amount of data, but you can't modify the template. If you must do a lot of modification, consider using REST API. For details, see *Import Price Lists*.
Pricing Spreadsheet with ADF Desktop Integration	Use interactive spreadsheets to maintain and update your data. They're interactive, so you can search and download data, and, if you encounter an error, you can fix it and resubmit, but they don't perform as well as File-Based Data Import when you must manage a large amount of data. If you need to automatically import data from your source system into Pricing, then use File-Based Data Import or REST API instead. For details, see *Overview of Using Spreadsheets to Manage Pricing* .
REST API	Use this industry standard technology to integrate with your systems, such as in a Platform as a Service (PaaS) environment, but it doesn't perform as well as File-Based Data Import when must manage a large amount of data. If you create a sales order, submit it, and modify the price list that you use to price the sales order, then you can't use REST API to update pricing on the sales order with the modified price list. For more, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Document Prices**, then click **Price Sales Transaction**.

Technology	Description
Pricing Administration work area	Use this graphical, easy to use interface, and there's almost no set up you can't do, but you must do all your data management manually. If you prefer to automate, consider using File-Based Data Import. For details, see *Administering Pricing*.

Here's a decision tree to help you decide:



Note

- If the requirement is to:
    - Integrate, then use REST API.
    - Automate, then use the Pricing Import spreadsheet.
    - Maintain, and if the volume of data to manage is:
        - High, then use the Pricing Import spreadsheet.
        - Low, and if you have:
            - Only a few lists to manage, use the Pricing Administration work area.
            - Lots of lists to manage, use the Pricing Spreadsheet.

For details about the privileges that you need to administer Pricing, see these sections in *Security Reference for Order Management*:

- Pricing Administrator (Job Role)
- Pricing Analyst (Job Role)
- Pricing Manager (Job Role)

Set Up Roles and Privileges for Pricing Administrators

Set up user roles and privileges to manage the authentication and authorization that Pricing Administration uses to secure processing for pricing, including web service usage.

Oracle Pricing implements security:

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

Here's where you can get background details:

Reference	Details
Securing SCM	Privileges and how to set up security, including values you set for each user.
Security Reference for Order Management	Privileges for Pricing Administration.

Get the privileges that you need to do the set up:

Role	Description
Pricing Administrator- All Business Units, which is QP_PRICING_ADMINISTRATOR_ALL_BUSIN	Use the privileges in this role to administer pricing.

Summary of the Set Up

Assume you need to set up two administrators, Diane Cho and Yu Li:

1. Create administrator and get privileges.
2. Manage data access.
3. Add administrator to business unit.
4. Get your privileges.

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

Create Administrator and Get Privileges

Create two administrators. One user can administer pricing set up. The other user can view pricing set up:

1. Identify the privileges that provides access to Pricing Administration:
    - See the Pricing Administrator chapter in *Security Reference for Order Management*.
    - Examine the privileges until you locate one that meets your needs. For this example, the Pricing Administrator job role has the privileges that you need.

      Don't assign a predefined role. Instead, make a copy of it, rename the copy, such as MY_ROLE, remove the privileges that your users don't need from MY_ROLE, then assign MY_ROLE to your users. For details, see *Guidance for Assigning Predefined Roles*.
2. Make sure you have the privileges that you need to manage job roles.

   If you don't have these privileges, then various actions will be grayed out when you do the Create Implementation Users task, and you won't be able to add privileges to a job role.
3. Go to the Scheduled Processes work area and run the Import User and Role Application Security Data scheduled process.

   This process updates the Create Implementation Users task with the latest user data. For details, see *Configure the Security Console*.
4. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Create Implementation Users**.
5. 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`

Attribute	Value
Password	Your user must use the password the first time the user signs in. Instruct your user to change the password immediately after sign in.

6. In the Add Role Membership dialog, enter `Pricing Administrator`, then click **Search**.
7. In the search results, click the **row** that contains the values:

Attribute	Value
Name	Pricing Administrator
Code	ORA_QP_PRICING_ADMINISTRATOR_JOB

8. click **Add Role Membership**, then click **Done**
9. Repeat the above steps for your next user:

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

10. On the Add User Account page, click **Save and Close**.

Manage Data Access

Manage data access for Yu.

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Data Access for Users**. For details about this task, 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 that you set up for 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:
   - Make sure you have the privileges that are in the IT Security Manager job role (ora_fnd_it_security_manager_job).

     In Microsoft Excel, in the Connect dialog, click **Yes**, then sign in.
   - In the Authorize Data Access for Users template that displays, verify that the template includes the security contexts that Yu needs for view access.
   - 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.

Add Administrator to Business Unit

You specify the business unit when you set up a price list in Pricing Administration. You must make sure the person who uses the Pricing Administration work area or imports a price list is in this business unit.

Assume you have a price list that's in the Vision Operations business unit. Yu Li is a pricing administrator who creates and updates price lists. You must add Yu to the Vision Operations business unit.

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Data Access Set Data Access for Users**.
2. On the Manage Data Access for Users page, click **Actions > Create**.
3. In the dialog that displays, set the values:

Attribute	Value
User Name	Yu Li
Role	Pricing Administrator
Security Context	Business Unit
Security Context Value	Vision Operations

4. Click **Save and Close > Done**.

Create Job Role

You can create a job role to meet your specific needs. In this example, you create a job role that allows Yu to view set ups in Pricing Administration but not edit them:

1. On the User Accounts page, click **Roles**.
2. On the Roles page, in the Search window, enter `Pricing Administrator`, then click **Search**.
3. In the search results, click **Actions > Copy Role**.
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	Pricing Administrator View Only
Role Code	QP_PRICING_ADMINISTRATOR_JOB_VIEW_ONLY
Description	Search for and view setups for pricing administration, including pricing strategies, price lists, service mappings, pricing algorithms, and so on.

6. On the Function Security Policies page, click **Load Inherited Policies**.
7. Delete all rows except rows that contain these privileges:
    - Manage Pricing Administration Work Area

- Manage Pricing Rules
- View Pricing Algorithms

If you must add a privilege, then click **Add Function Security Policy**, and add it.

For example, in the Add Function Security Policy dialog, enter `View Pricing`, examine the list of privileges that displays, then add the ones you need.

Here are some examples of privileges you can add:

- View Pricing Algorithms
- View Pricing Bases
- View Pricing Charge Definitions
- View Pricing Guidelines
- View Pricing Matrix Types
- View Pricing Messages
- View Pricing Parameter Values

8. Click **Next**.
9. On the Data Security Policies page, delete rows as necessary, then click **Next**.

   To delete a row, select it, then click **Remove Data Security Policy**.
10. Click **Next**.
11. On the Role Hierarchy page, delete all role hierarchies except for:

Role Name	Role Code
Item Inquiry	ora_egp_item_inquiry_duty_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 details about the role hierarchy that each predefined job role uses, see *Security Reference for Order Management*.
12. Click **Next**.
13. On the Users page, click **Add User**.
14. In the Add User dialog, search for Yu Li, wait for the results to display, click **Add User to Role**, then click **Cancel**.
15. Click **Next > Submit and Close**.

Examine Role Usage in Your Implementation Project

Your implementation project specifies the roles that can do 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.
6. Expand the Order Management row.
7. In the Pricing row, add a check mark in the Include column, then click **Save and Open Project**.
8. In the Task list, expand **Order Management > Define Pricing**, then notice the list of tasks you can access, such as Manage Price Elements.
9. In the Manage Price Elements row, in the Authorized Roles column, click **Details**, then examine the roles that can access the Manage Price Elements task.

Manage Pricing Parameters

Use a pricing parameter to manage behavior that applies across Oracle Pricing.

Each pricing parameter comes predefined with Oracle Pricing. You can't create or delete a pricing parameter, but you can modify a parameter value.

Try it:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Parameters task.
2. On the Manage Pricing Parameters page, modify the parameters, as necessary:

Pricing Parameter	Description
Algorithm Exception Details	If a pricing algorithm fails because an exception happens at runtime, then an error message describes the error. For example, an error message in the Order Management work area. You can use this parameter to control how much detail you want to display in the message. Set it to: o **Summary**. Display only a summary of the error in the error message. o **Full**. Display all details about the error in the error message. All details can include a lot of information that you don't typically need, so we recommend that you set it to Full only when you need to troubleshoot a problem.
Default Price Periodicity UOM Class	Set the UOM class that Pricing uses to display the default value for frequency attributes in the Pricing Administration work area. Pricing uses this default value for each recurring charge or usage charge for frequency attributes. For example, set the Default Price Periodicity UOM Class parameter to Time to use time values, such as Weekly, Monthly, or Per Quarter. For details about frequency, see *Manage Pricing Charge Definitions*.
Item Validation Organization	See the section later in this topic.

Pricing Parameter	Description
Maximum Number of Child Processes	Specify the maximum number of child scheduled processes to run when you import pricing data. See *Import Price Lists*.
Primary Catalog	Select the catalog that contains the categories you want to use to price an item. You can set the Primary Catalog parameter only to a catalog that's in the Product Catalogs parameter. If the list of values for the Primary Catalog doesn't have the catalog you need, then add that catalog to the Product Catalogs parameter, then set the Primary Catalog.
Product Catalogs	Select the catalog that you want Pricing to use when it calculates the price, adjustments, and cost for each item.

3. Click **Save**.
4. Consider the Order Management parameters that affect pricing. See *Manage Order Management Parameters*.

Item Validation Organization

Set the item validation organization that Pricing uses for each business unit when it validates an item:

- Pricing filters the items that it displays according to the item validation organization that you select. For example, if you select Vision Operations as the business unit, and V1 as the Organization, then Pricing only displays items that are part of the V1 organization when you create a pricing entity in the Pricing Administration work area.
- Product Management associates each inventory organization with a business unit, and it determines the values that you can select for the organization. For details about how to set up Product Management, see *Implementing Common Features for Oracle SCM*.
- If you don't specify at least one item validation organization, then you can't view items in the Pricing Administration work area.

You must specify All Business Units in the Business Unit attribute for at least one organization. For example:

Business Unit	Organization
All Business Units	Vision Operations

If you need a different item validation organization for a specific business unit, you can add it. For example:

Business Unit	Organization
All Business Units	Vision Operations
Singapore Distribution Center	Singapore Distribution Center

Related Topics

- Manage Pricing Charge Definitions
- Pricing Rules
- Overview of Oracle Pricing
- Manage Price Lists

Times, Dates, and Values

Set the Time Zone for Pricing Administration

Specify the time zone to use for time attributes that display in the Pricing Administration work area, such as the Start Date and End Date.

Time attributes store date and time values in the server time zone, by default. You can specify your time zone to display in the Pricing Administration work area.

Assume you must set the time zone to Pacific Time:

1. Go to your Home page, then click **Set Preferences**.
2. On the Preferences page, click **Regional**.
3. On the General Preferences page, set the Time Zone preference to your preferred time zone.

   The Pricing Administration work area will use this preference to determine the time zone to use when it displays time attributes.

   For this example, set it to (UTC-8:00) Los Angeles - Pacific Time (PT).
4. Create a new price list, then notice the Start Date and End Date on the Edit Price Page displays in the Pacific time zone.

Note

- Pricing converts your time zone to the server time zone when it saves the value of a time attribute to the database. For example, it converts the value in Start Date to the server time zone. If you query the price list, then Pricing converts Start Date from server time to Pacific Time and displays it.
- This feature applies only to values in the Pricing Administration work area. It doesn't apply to file-based data import, pricing spreadsheets you use through Application Development Framework Desktop Integration (ADFDI), or pricing through REST API. These technologies use the server time zone. For details and examples about REST API, go to *REST API for Oracle Supply Chain Management Cloud*, then expand Order Management.

Set Up Units of Measure for Pricing

Set up units of measure so Pricing can convert the unit of measure.

For example, assume Pricing receives order lines x and y, the item for each order line is AS54888, but the UOM for order line x is Each and the UOM for order line y is Box of 5:

- Pricing doesn't convert Box of 5 to a quantity of 5 and a UOM of Each.
- Pricing doesn't calculate the price for Box of 5 as five multiplied by the price for Each.

So, you must add a separate line for the item in the Price List Line tab for each UOM that you sell for AS54888. You can then click Create Charge to specify how to calculate price for Box of 5. For details, see *Manage Price Lists*.

Note

- Primary Pricing UOM doesn't affect pricing calculations or relationships in the current update. Pricing includes it to support converting a unit of measure in a future update.
- You can add only one unique combination of item, unit of measure, and line type for the primary pricing UOM.

Here's an example that sets up two items when you must support different units of measure for the same item:

Item	Description	Pricing UOM	Primary Pricing UOM
AS54888	Desktop Computer	Each	Contains a check mark
AS54888	Desktop Computer	Box of 5	Doesn't contain a check mark
CM15140	Computer Monitor	Each	Contains a check mark
CM15140	Computer Monitor	Box of 5	Doesn't contain a check mark

Related Topics

- Manage Price Lists
- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Strategies
- Adjust Price for Pricing Rules
- Troubleshoot Your Pricing Setups

Manage Pricing's Effectivity Dates

Make sure your effectivity dates work correctly when you price an item in Order Management or when you use the PriceSalesTransaction REST API.

The start date and the end date on a pricing entity determines when the entity is in effect. We refer to a set of these start and end dates as an effectivity date. The effectivity dates that you specify when you set up pricing affect runtime behavior. Here are some example error messages that you might get if you don't set up your dates correctly.

```
A matching price list cannot be found for this transaction for the pricing strategy

Pricing strategy was not determined for the transaction

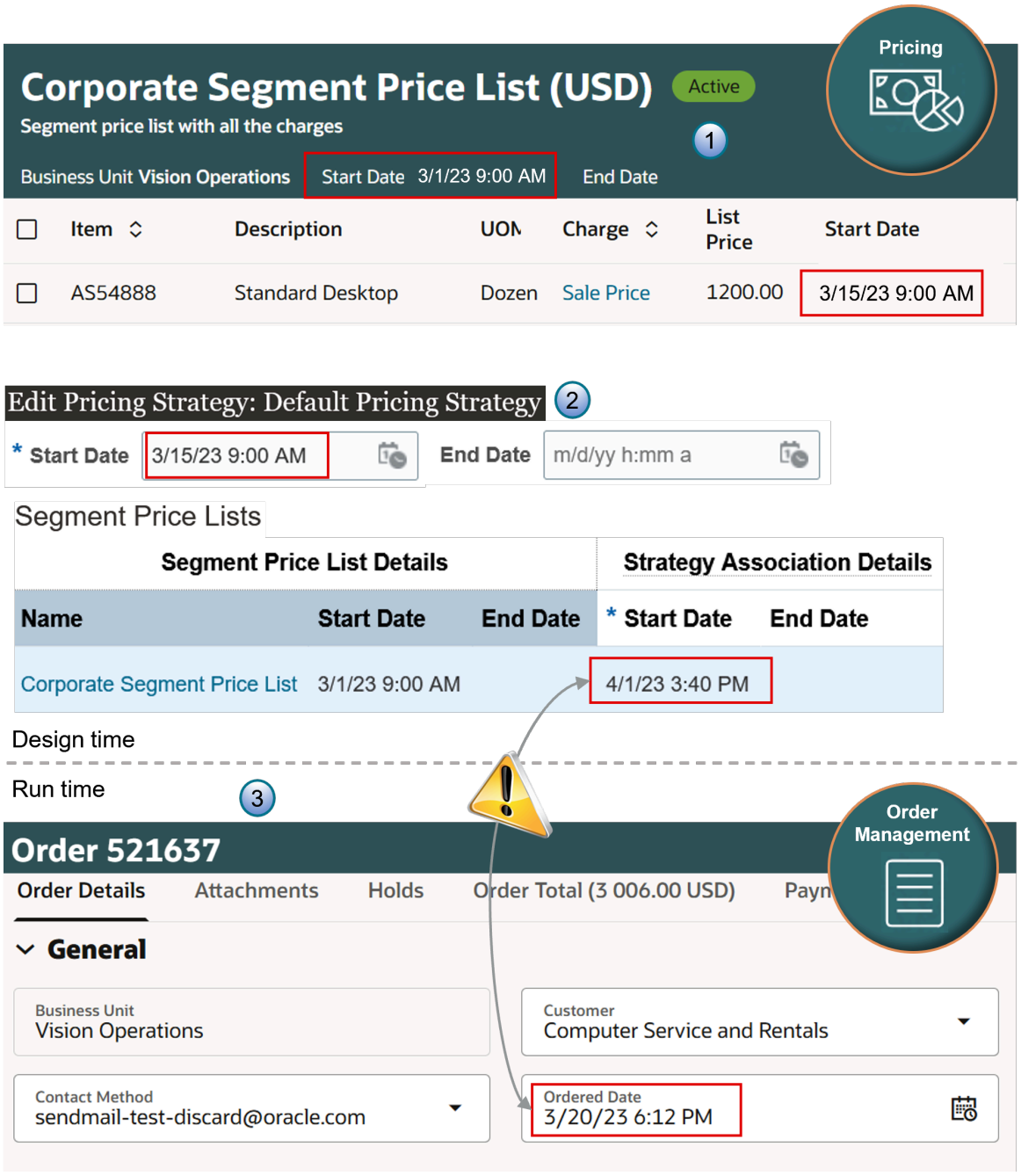
No matching pricing segment found for the transaction
```

```
No matching pricing segment was found for the customer
```

For more, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Document Prices**, then click **Price Sales Transaction**.

Example 1: Problem with the Price List's Dates

Assume you do this setup.



Assume you:

1. Go to the Pricing Administration work area, then use the Price List page to:
    - Set the start date on the Corporate Segment Price List to March 1, and leave the end date for it empty.
    - In the Price List Lines tab, you add the AS54888 Desktop Computer to the Corporate Segment Price List, add a charge to the item, set the start date for the charge to March 15, and leave the charge's end date empty.
2. Use the Edit pricing Strategy page to:
    - Set the start date on the Default Pricing Strategy to March 15, and leave the end date for it empty.
    - Add the Corporate Segment Price List to the Default Pricing Strategy, use the Strategy Association Details area to set the start date to April 1 when you add the price list to the strategy, and leave the list's end date empty.

      You use the Strategy Association Details area to specify the dates when you want to use this price list with the pricing strategy. The Segment Price List Details area displays the start and end date that you set for the price list. You can't edit these dates on the Edit Pricing Strategy page, but you can edit them on the Price List page.
3. Go to the Order Management work area, create a sales order, set the Ordered Date attribute on the order header to March 20, add the AS54888 to the order, then click Actions > Reprice order.

   Pricing successfully determines the pricing strategy to use according to your pricing setup, but then Order Management displays an error message:

```
A matching price list cannot be
 found for this transaction for the pricing strategy because the order date is outside the
 range of the start and end dates of the Strategy Detail for the price list in the
 strategy.
```

This happens because the Corporate Segment Price List doesn't start until April 1 on the Default Pricing Strategy, which is after the March 20 ordered date.

To fix this problem, set the start date for the Corporate Segment Price List on the Default Pricing Strategy to a date that happens before March 20, or wait until after the price list's April 1 start date, then reprice your order.

Example 2: Problem with the Charge's Dates

Assume you have the same setup that you use in example 1, but you also set up a $200 charge for the AS54888, set the charge's start date to May 15, and leave the charge's end date empty.

On May 10, you create a sales order with an order date of May 15 and add the AS54888 to the order. Pricing successfully determines the pricing strategy to use according to your pricing setup, but then Order Management displays an error message.

```
A matching price list cannot be found for this transaction for the pricing strategy because the charge start
 and end dates do not include the calendar date.
```

To fix this problem, change the charge's start date to a date that happens before May 10, or wait until after the charge's May 15 start date, and then reprice the order.

Apply the Theory Behind the Examples

Now that you can see how date problems happen at run time, you can use this section to consider how your effectivity dates might cause runtime problems. Consider these facts when you do your set up, or when you need to troubleshoot runtime problems.

Here are some important attributes that Pricing examines when it calculates price:

- PriceAsOf
- PricingDate
- PricedOn

Pricing uses a variety of source attributes to determine PriceAsOf, PricingDate, and PricedOn, and these sources vary depending on whether Pricing is pricing the order header or the order line.

Header Entity Stores Details for the Order Header

Date Attribute	Source Attribute	Default Value	Description
PriceAsOf	OrderedDate	CurrentPricingDate	If OrderedDate is empty, the Pricing uses CurrentPricingDate.
PricingDate	TransactionOn	CurrentPricingDate	If OrderedDate is empty, the Pricing uses CurrentPricingDate.
PricedOn	-	CurrentPricingDate	PricedOn stores the date and time when Pricing prices the sales order. Pricing sets the value in PricedOn.

Note that CurrentPricingDate is the system date.

Line Entity Stores Details for the Order Line

Date Attribute	Source Attribute	Default Value	Description
PriceAsOf	OrderedDate	Header.PriceAsOf	If OrderedDate is empty, then Pricing uses PriceAsOf from the Header entity.
PricingDate	TransactionOn	Header.PricingDate	If TransactionOn is empty, then Pricing uses PricingDate from the Header entity.
PricedOn	-	CurrentPricingDate	PricedOn stores the date and time when Pricing prices the order line. Pricing sets the value in PricedOn.

Consider How Your Service Mapping Affects Dates

If you set up your own service mapping, and if your mapping modifies the OrderedDate source attribute or the TransactionOn source attribute, then you'll want to consider this section when you set up pricing, or when you troubleshoot your setup.

Pricing uses the runtime value in the Header.PriceAsOf date attribute to determine whether strategies, strategy assignments, profiles, matrices, and rules are in effect when it prices your order.

Here are some tips that you can use to make sure these entities are in effect when you need them:

Entity in Pricing Administration	Description
Pricing Strategy	Make sure the start date and end date of your pricing strategy happens within the time frame that Header.PriceAsOf contains. For example, if Header.PriceAsOf contains May 15, and if your pricing strategy's start date happens on May 14 and it's end date is empty, then you're good to go. You set these dates on the Manage Pricing Strategies page.
Pricing Strategy Assignment	Make sure the start date and end date for the assignment level, pricing context, and transaction type in the assignment for your pricing strategy happens within the time frame that Header.PriceAsOf contains. You set these dates on the Manage Pricing Strategy Assignments page.
Pricing Profile	Make sure the start date and end date of your pricing profile happens within the time frame that Header.PriceAsOf contains. You set these dates on the Manage Customer Pricing Profiles page.
Row That You Add to a Pricing Segment Matrix	If you have a Pricing Segment matrix type, and if you enable the Date Effectivity option on that matrix, then make sure the start date and the end date of your matrix happens within the time frame that Header.PriceAsOf contains. You set these dates on the Manage Matrix Types page.
Rule or Row That You Add to an Assignment Matrix	If you enable the Date Effectivity option for the Pricing Strategy assignment matrix type, then make sure the start date and the end date of the Pricing Strategy assignment matrix type happens within the time frame that Header.PriceAsOf contains. You set these dates on the Manage Matrix Types page.

Various Kinds of Lists

List in Pricing Administration	Date Attribute	Description
Any of these lists that you add to a pricing strategy: • Price List • Cost List • Discount List • Currency Conversion List	Line.PriceAsOf	Make sure the start date and the end date that you specify in the header area of your pricing strategy and the start date and the end date that you specify in the Strategy Association Details area when you add the list to the strategy happen within the time frame that Line.PriceAsOf contains.
Any of these lists that you create or modify: • Price List • Cost List • Discount List • Currency Conversion List	Line.PriceAsOf	Make sure the start date and the end date that you specify in the header area of your list happens within the time frame that Line.PriceAsOf contains. For example, if Line.PriceAsOf contains May 15, then make sure the start date that you set when you create or modify your list happens before May 15, and the end date that you set happens after May 15.

List in Pricing Administration	Date Attribute	Description

Price List for Configured Items

Entity in Pricing Administration	Date Attribute	Description
Price List That You Add to a Pricing Strategy	Line.PriceAsOf	Same as for a standard item or coverage item.
Child Item of a Configured Item That You Add to a Price List	-	Pricing doesn't use any date attribute.

Override a Price List According to a Pricing Term

Entity in Pricing Administration	Date Attribute	Description
Pricing Term That You Add to a Price List	Line.PriceAsOf	Make sure the start date and the end date that you specify for the pricing term happen within the time frame that Line.PriceAsOf contains.
Item That You Add to a Price List	-	Pricing doesn't use any date attribute.

Currency Conversion Rate

Entity in Pricing Administration	Date Attribute	Description
Conversion Rate	Line.PricingDate	Make sure the start date and the end date of the conversion rate happen within the time frame that Line.PricingDate contains. To set the conversion rate's date, use the Manage Currency Conversion Lists task to open your list for editing. In the Details area, click **Conversion Rates**, click **Actions > Edit**, then set the Start Date attribute and the End Date attribute.

If you use the conversion rate with a coverage, then Pricing uses the start date and the end date of the coverage's duration when it determines whether the conversion is in effect. If these coverage dates are empty, then Pricing uses the coverage's start date and end date.

Charges and Shipping Charges for Items, Configured Items, Configurable Options, and Coverages

Entity in Pricing Administration	Date Attribute	Description
Charge on Any List, Such as a Price List	Line.PricingDate	Make sure the start date and the end date of the charge that you add to the list happen within the time frame that Line.PricingDate contains.

Entity in Pricing Administration	Date Attribute	Description
Charge on the Adjustment Matrix of Any List or Pricing Term	Line.PricingDate	If you enable the Date Effectivity Enabled option on the matrix, then make sure the start date and the end date of the adjustment that you add to the list happen within the time frame that Line.PricingDate contains. For details, see *Manage Pricing Matrices*.
Rule on Any List	Line.PricingDate	Make sure the rule's start date and end date happen within the time frame that Line.PricingDate contains. For details, see *Pricing Rules*.

More Entities

Entity in Pricing Administration	Date Attribute	Description
Pricing Term That You Import	Line.PricingDate	Make sure the start date and the end date of the pricing term that you import happens within the time frame that Line.PricingDate contains. This requirement applies only with pricing terms that you import through some integrations, such as Channel Revenue Management.
Override Currency on a Pricing Strategy	Header.PricingDate Header.PriceAsOf	Make sure the start date and the end date that you set when you use the Allowed Override Currencies tab of the pricing strategy happens within the time frame that Header.PricingDate contains, and that Header.PriceAsOf contains.

Filter Dates on Pricing Algorithms

You can use PriceAsOf, PricingDate, and PricedOn on these pricing algorithms to filter dates:

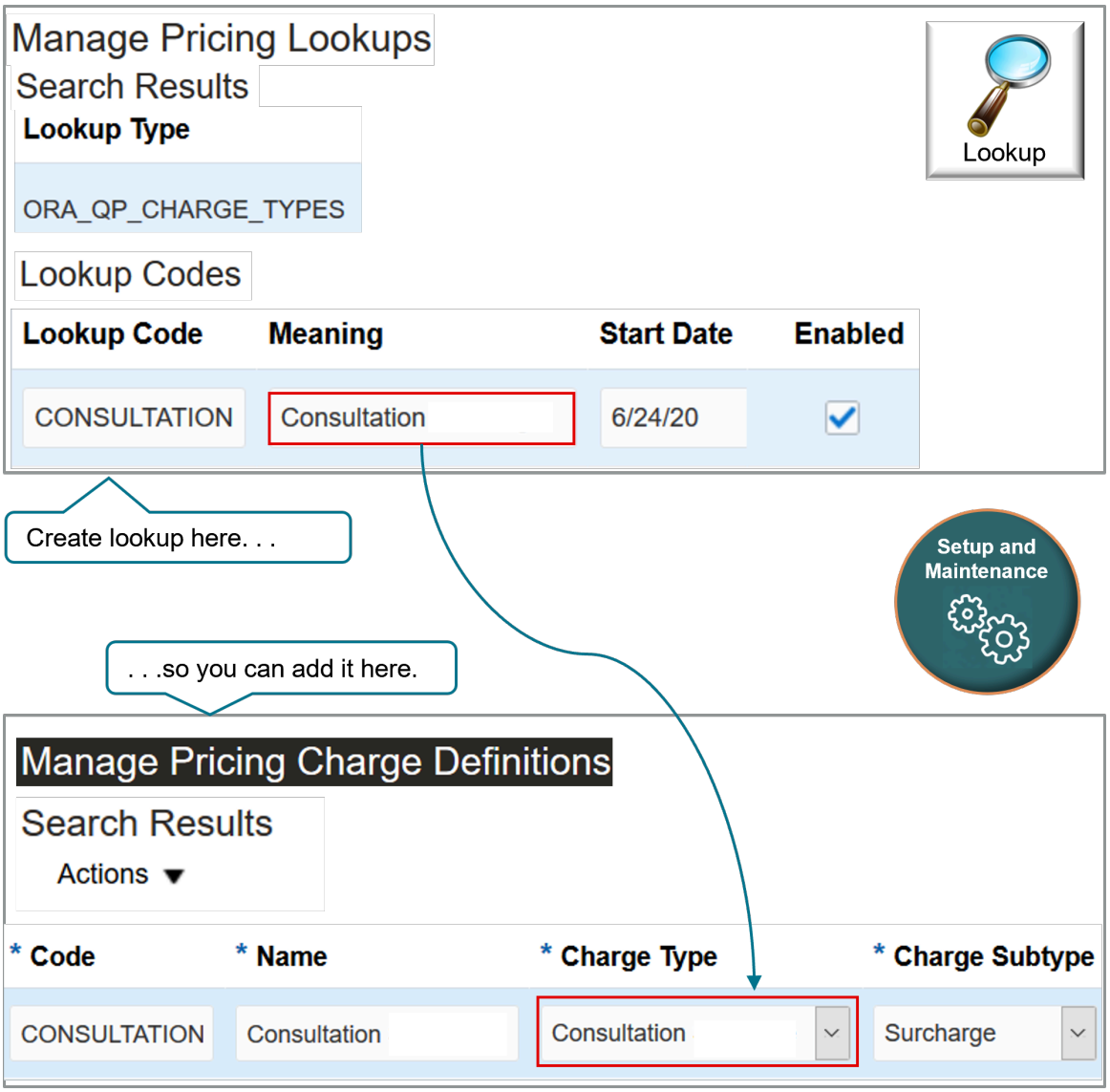
- Set Initial Values
- Get Sales Pricing Strategy
- Validate Pricing Terms
- Validate Override Currencies
- Get Base List Price for Goods and Services
- Get Currency Conversion Rates
- Derive Price List
- Apply Discounts
- Apply Pricing Terms
- Apply Matrices

Price According to Pricing Date

Your item's price might change over time but you need to honor the price commitment that you provide to your customer. You can use the new Pricing Date attribute on the order header to specify the date that you want Oracle Pricing to use when it prices the order. For details, see *Specify Pricing Dates on Sales Orders*.

Add Your Own Lookup to Charge Types

Use the Manage Pricing Lookups page to add your lookup to a charge type, then select that lookup when you set up a charge definition on the Manage Pricing Charge Definitions page.



Note

- Add your lookup to ORA_QP_CHARGE_TYPES for a charge type.

- Add your lookup to ORA_QP_CHARGE_SUBTYPES for a charge subtype.

Assume you provide a site consultation for some customers. Your policy is to add a surcharge to the purchase price to cover the costs you incur to do the consultation. So you need to create a new charge definition for the consultation surcharge, a new charge type for the consultation, and a new charge subtype for the surcharge.

Create the charge type and charge subtype:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Lookups task.
2. On the Manage Pricing Lookups page, search for the value:

Attribute	Value
Lookup Type	ORA_QP_CHARGE_TYPES

3. In the Lookup Codes area, click **Actions > New**, then set the values:

Attribute	Value
Lookup Code	CONSULTATION
Meaning	Consultation
Display Sequence	1
Enabled	Contains a check mark
Start Date	Set to today or a date that happens before today.

4. Click **Save**.
5. In the Search area, search for the value:

Attribute	Value
Lookup Type	ORA_QP_CHARGE_SUBTYPES

6. In the Lookup Codes area, click **Actions > New**, then set the values:

Attribute	Value
Lookup Code	SURCHARGE
Meaning	Surcharge

Attribute	Value
Display Sequence	1
Enabled	Contains a check mark
Start Date	Set to today or a date that happens before today.

7. Click **Save and Close**.

Create your charge definition:

1. On the setup page, go to the Manage Pricing Charge Definitions task.
2. On the Manage Pricing Charge Definitions page, click **Actions > Add Row**, then set the values:

Attribute	Value
Code	CONSULTATION
Name	Consultation
Applies To	Price
Price Type	One Time
Charge Type	Consultation This is the charge that you created on the Manage Pricing Lookups page.
Charge Subtype	Surcharge This is the charge subtype that you created on the Manage Pricing Lookups page.

3. Click **Save and Close**.

Related Topics

- Manage Pricing Lookups
- Manage Pricing Charge Definitions

Use Empty Values as Filters

Use an empty value in your pricing setup as a way to filter the pricing segments that Pricing considers when it prices an item.

Assume you set up pricing segments:

Revenue Potential	Customer Rating	Pricing Segment	Precedence
Low	-	Corporate Segment	98
Low	-	International Segment	100
Low	Low	Domestic Segment	100

You create a sales order, and Pricing uses the pricing strategy you assigned to Corporate Segment, but you want it to use the pricing strategy you assigned to Domestic Segment.

Pricing uses Corporate Segment because its precedence is 98, and Pricing uses the segment with the lowest precedence.

Next, assume you have three customers:

Customer	Revenue Potential	Customer Rating
A	Low	-
B	Low	-
C	Low	Low

Note

- You create a sales order for customer C and find that Pricing uses Corporate Segment but you want to use Domestic Segment, so you adjust precedence for Corporate Segment from 98 to 110.
- You create another sales order for customer C and find that Pricing now uses International Segment.
- If you enable Null Is Wildcard for Customer Rating, then Pricing considers all three matrix rows. So, Pricing will pick International Segment or Domestic Segment, depending on which row the query returns first.
- You want to use Domestic Segment but you don't want to adjust precedence because you have 20 other segments and you don't want a precedence adjustment to impact them.

If you already set up a customer pricing profile for each customer, then do this:

1. In the Pricing Administration work area, click **Tasks > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click the **value**:

Attribute	Value
Name	Pricing Segment

3. On the Edit Matrix Class page, disable null for the customer rating:

Name	Null Is Wildcard
Customer Rating	Doesn't contain a check mark.

Now, Pricing won't consider Corporate Segment or International Segment because their Customer Rating on the pricing segment doesn't contain a value. Instead, it will use Domestic because its Customer Rating does contain a value. Its value is Low.

Pricing Matrixes

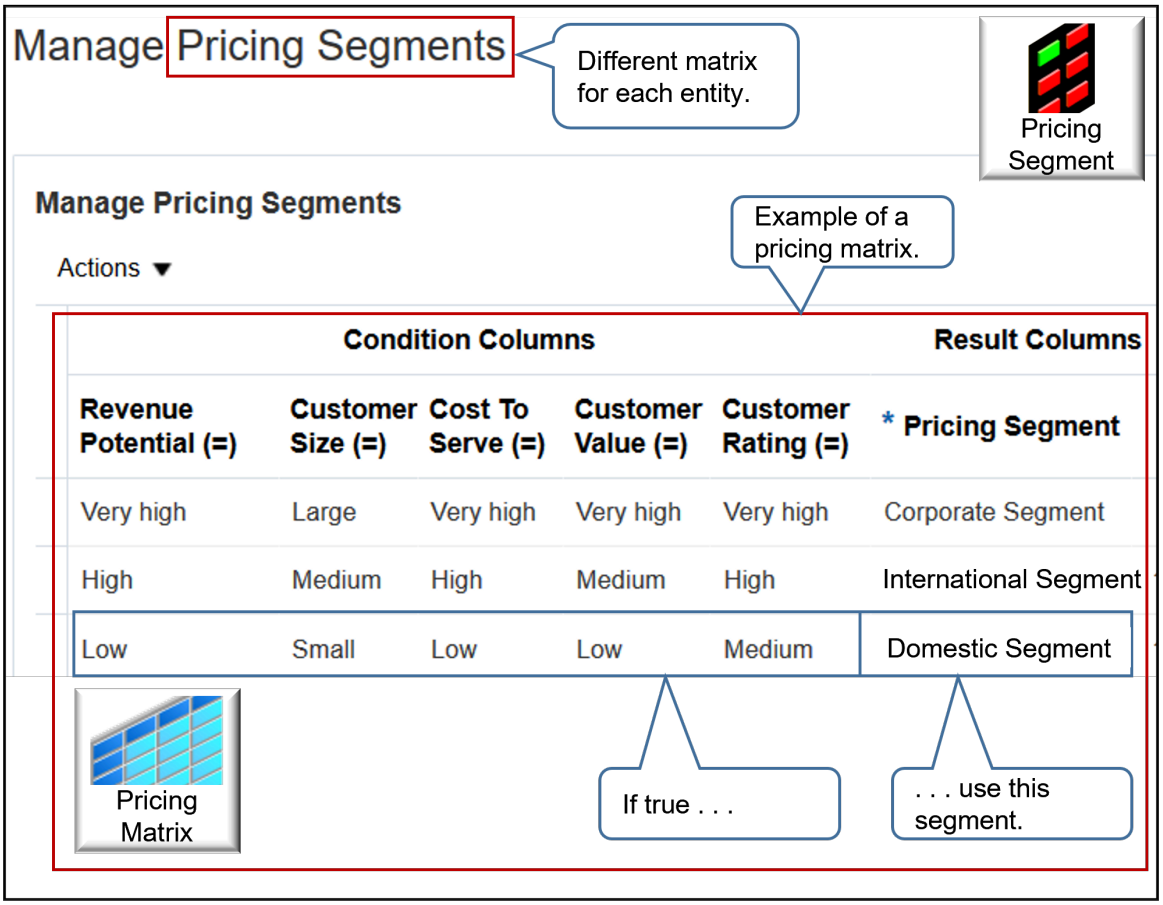
Pricing Matrices

Use a pricing matrix to apply conditional logic on a pricing entity according to the value of an attribute that you specify.

A pricing matrix is an array that specifies conditions and results:

- The array contains rows and columns.
- Condition columns specify the condition to test, such as Customer Size equals Small.
- Result columns specify what to do when the condition is true, such as use the Corporate Segment.
- Each row specifies a separate condition and result.

This pricing matrix assigns a pricing segment:



This matrix includes three rows. Each row specifies a rule that contains a condition and a result:

Row	Condition	Result
1	If Revenue Potential is Very high, and: • Customer Size is Large, and • Cost to Serve is Very High, and • Customer Value is Very High, and • Customer Rating is Very High	Assign the Corporate Segment.
2	If Revenue Potential is High, and: • Customer Size is Medium, and • Cost to Serve is High, and	Assign the International Segment.

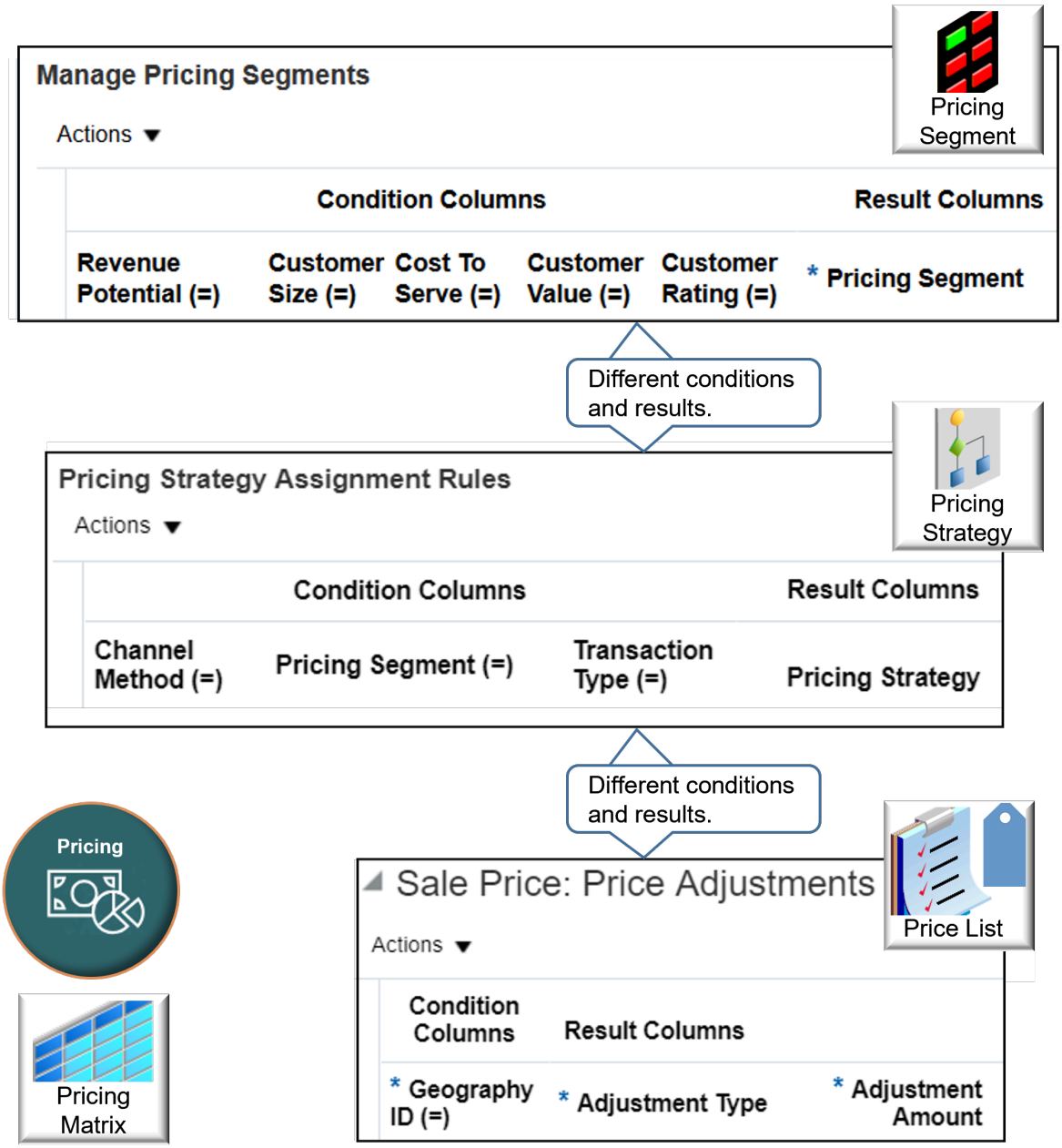
Row	Condition	Result
	• Customer Value is Medium, and • Customer Rating is High	
3	If Revenue Potential is Low, and: • Customer Size is Small, and • Cost to Serve is Low, and • Customer Value is Low, and • Customer Rating is Medium	Assign the Domestic Segment.

At runtime, Pricing evaluates each row until it finds a match. If it doesn't find a match, then it sets a default value.

Here are the pricing entities where you can add a pricing matrix:

- Pricing strategy assignment
- Pricing segment
- Price list
- Cost list
- Currency conversion list
- Shipping charge list

The conditions and results are different for each type of pricing entity:



For example:

Pricing Entity	Conditions	Results
Pricing segment	Depending on the values of: • Revenue Potential • Customer Size • Cost to Serve • Customer Value	Set the: • Pricing Segment

Pricing Entity	Conditions	Results
	• Customer Rating	
Pricing strategy assignment	Depending on the values of: • Channel Method • Pricing Segment • Transaction Type	Set the: • Pricing Strategy
Price list	Depending on the value of: • Geography	Set the: • Adjustment Type • Adjustment Amount

Add Conditions and Results in Pricing Matrixes

You can add up to 25 conditions and results in a pricing matrix when you create a pricing rule. You can use this capability to include more attributes from your customer, item hierarchy, or transaction when you set up your pricing rules. It also provides a wider variety of attributes that can you use to help analyze pricing in your reports:

- Use the Pricing Administration work area, file-based data import, or REST API to manage your matrix when you need to adjust attribute values on price lists or discounts lists.
- Use the Pricing Administration work area or REST API to assign the pricing strategies and manage the pricing segments that you use with these rules.
- Use the Pricing Administration work area to manage adjustment values for attributes on a price list or discount list, to manage pricing segments, assign pricing strategies, and create pricing guidelines.
- You can add up to only 10 conditions and results through Application Development Framework Desktop Integration (ADFDI). For details, see *Use Spreadsheets to Manage Pricing*.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Matrices
- Manage Pricing Segments
- Pricing Rules

Pricing Matrix Class

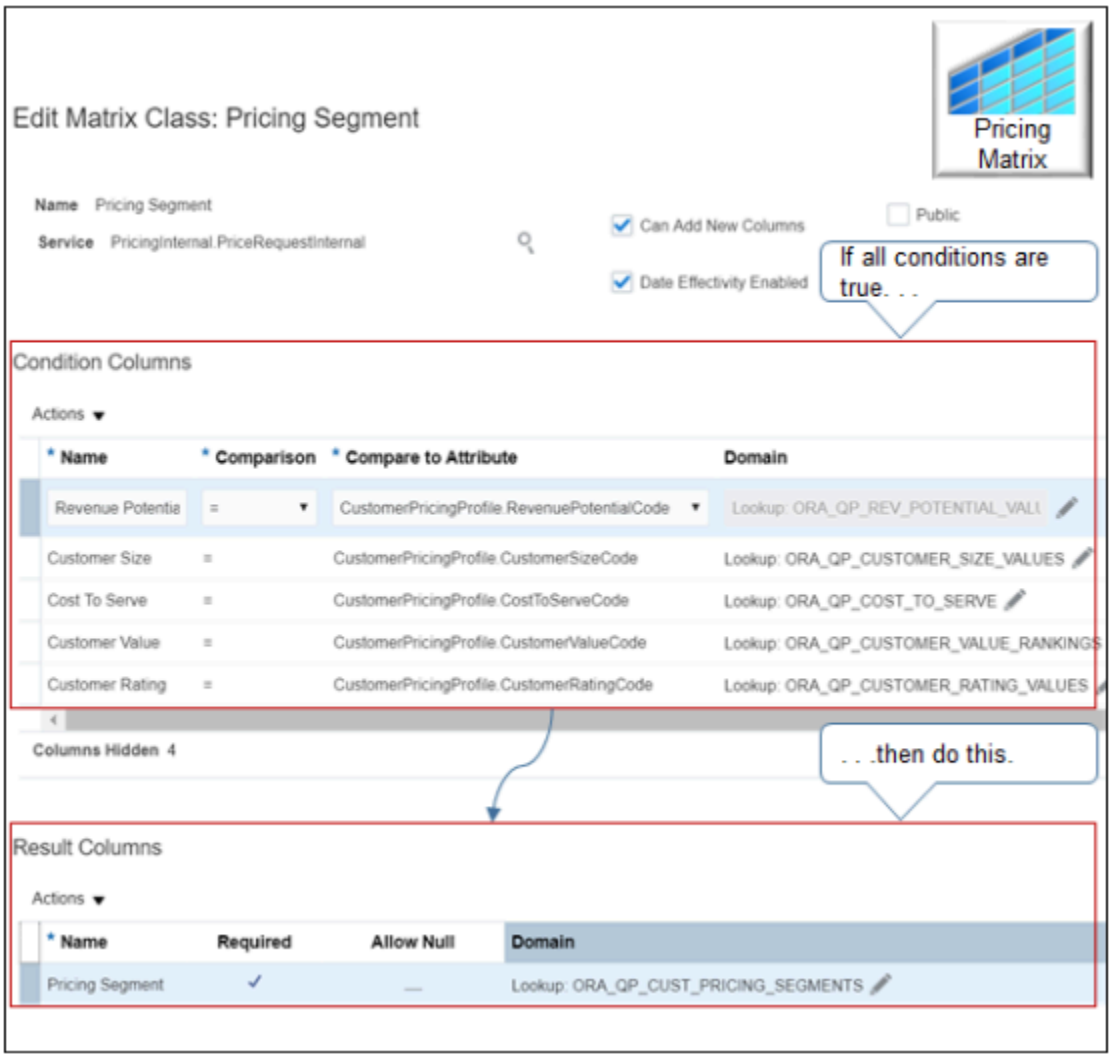
A pricing matrix class is a template that you can use to specify the structure of the pricing matrix. Use it to add condition columns and result columns to a pricing matrix.

Each pricing matrix class includes columns:

- Condition columns specify the conditions that determine whether to apply the result.
- Result columns specify what to do when all the conditions are true.

Example

Here's the predefined Pricing Segment matrix class:



The predefined Pricing Segment matrix class includes condition columns:

Name	Comparison	Compare-to Attribute	Domain
Revenue Potential	=	CustomerPricingProfile.RevenuePot	Lookup:ORA_QP_REV_POTENTIAL_VALUES
Customer Size	=	CustomerPricingProfile.CustomerSiz	Lookup:ORA_QP_CUSTOMER_SIZE_VALUES
Cost to Serve	=	CustomerPricingProfile.CosttoServe	Lookup:ORA_QP_COST_TO_SERVE
Customer Value	=	CustomerPricingProfile.CustomerVa	Lookup:ORA_QP_CUSTOMER_VALUE_RANKINGS

Name	Comparison	Compare-to Attribute	Domain
Customer Rating	=	CustomerPricingProfile.CustomerRa	Lookup:ORA_QP_CUSTOMER_RATING_VALUES

And this result column:

Name	Domain
Pricing Segment	Lookup:ORA_QP_CUST_PRICING_SEGMENTS

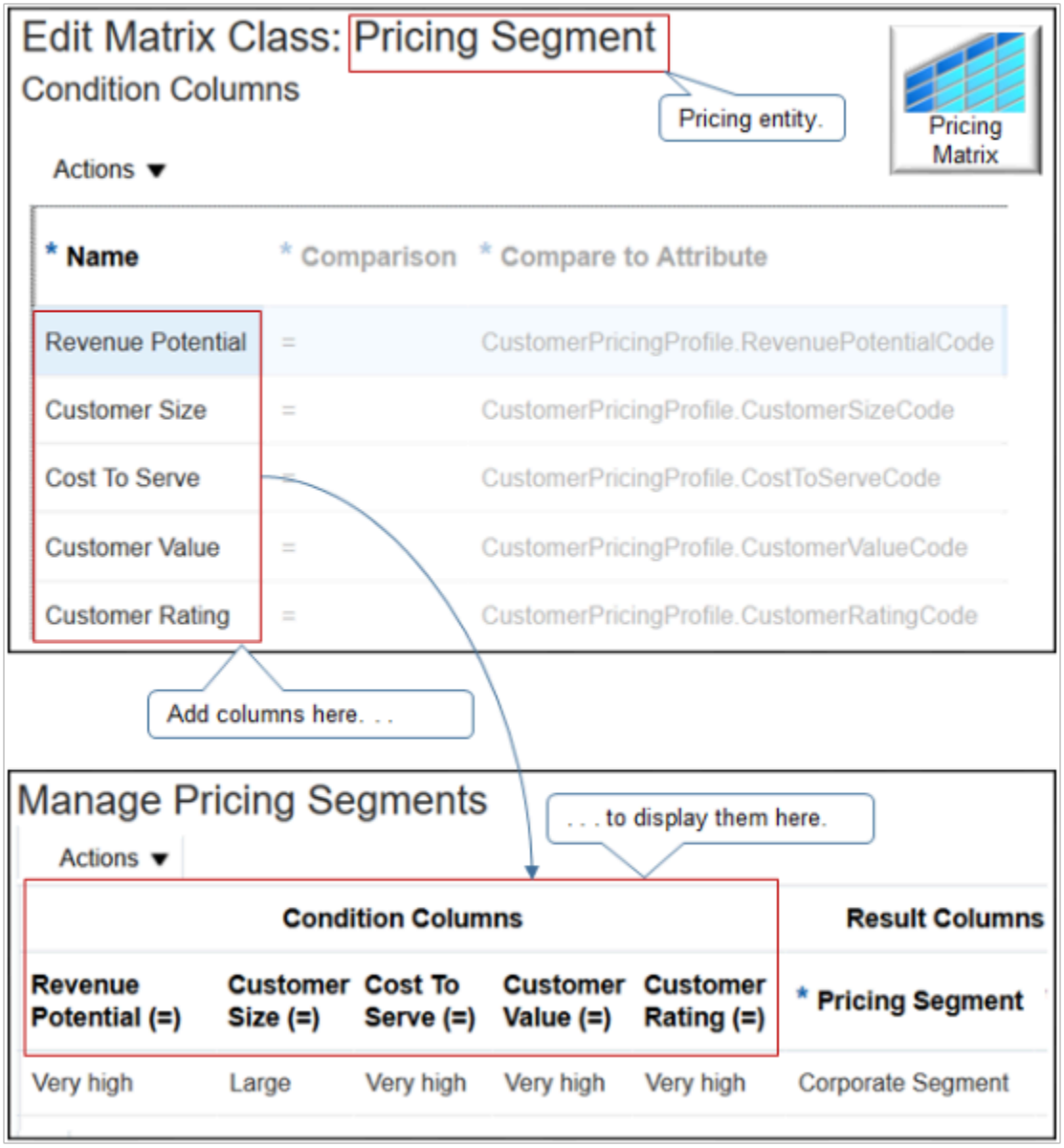
In pseudocode, it means:

- If Revenue Potential equals the value of the RevenuePotentialCode attribute of the CustomerPricingProfile entity.
- If Customer Size equals the value of the CustomerSizeCode attribute of the CustomerPricingProfile entity.
- If Cost to Serve equals the value of the CosttoServeCode attribute of the CustomerPricingProfile entity.
- If Customer Value equals the value of the CustomerValueCode attribute of the CustomerPricingProfile entity.
- If Customer Rating equals the value of the CustomerRatingCode attribute of the CustomerPricingProfile entity.

then

- Set the Pricing Segment to a value from the ORA_QP_CUST_PRICING_SEGMENTS lookup.

You can use the Pricing Segment matrix class to specify the condition columns that the Pricing Administration work area displays for the pricing segment, such as Revenue Potential:



Note

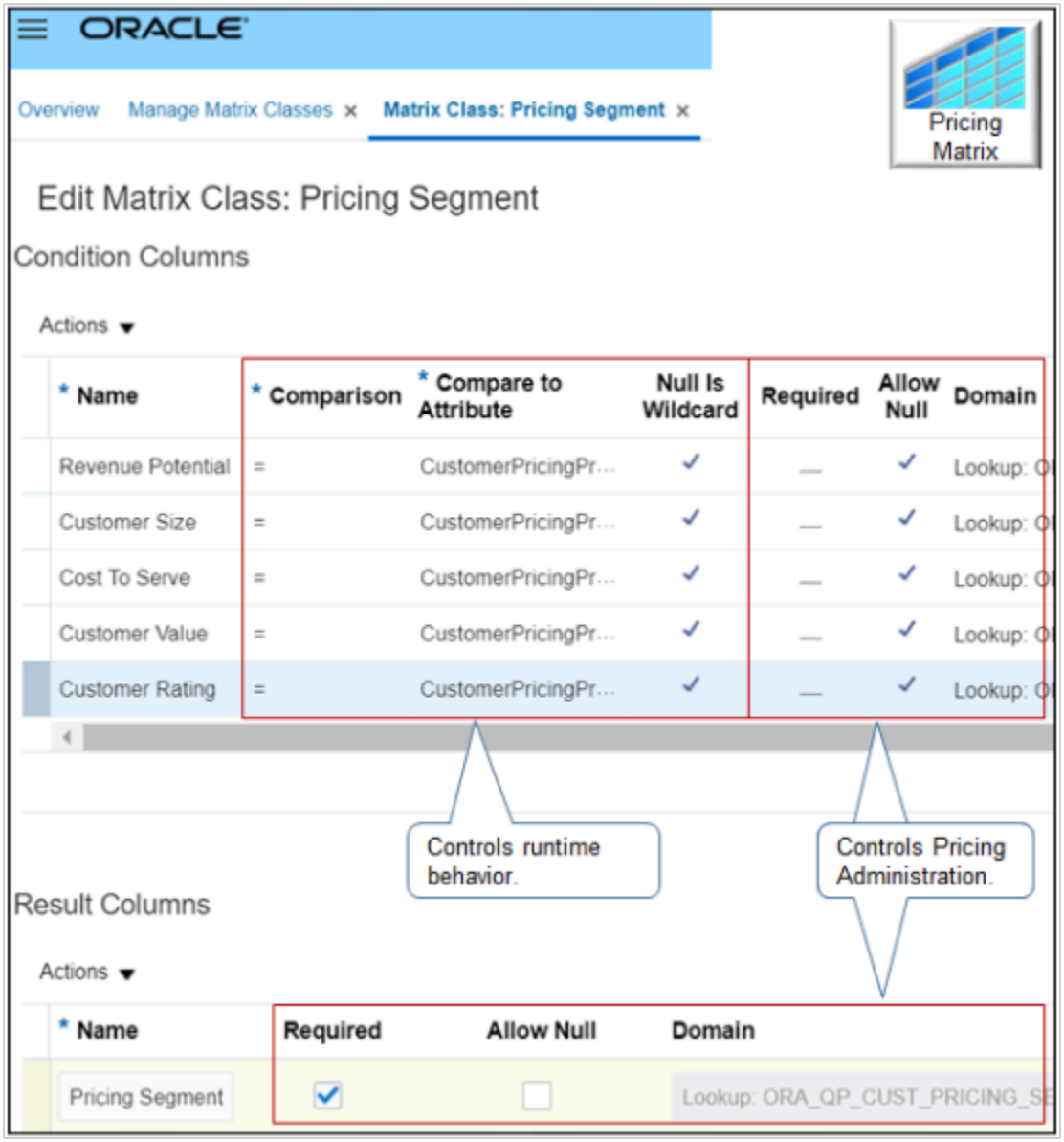
- Oracle Pricing comes predefined with matrix types and matrix classes. Each type references one, and only one, class. Each class specifies conditions and results:

Predefined Matrix Type	Predefined Matrix Class
Pricing Segment	Pricing Segment
Sales Pricing Strategy Assignment	Sales Pricing Strategy Assignment
Price List Charge Adjustment	Price List Charge Adjustment

Predefined Matrix Type	Predefined Matrix Class
Discount Adjustment	Pricing Term Adjustment
Pricing Term Adjustment	Pricing Term Adjustment
Line Strategy	Line Pricing Strategy Assignment
Pricing Charge Guideline	Pricing Charge Guideline

- Use the predefined classes, when possible.
- If you must modify a predefined class, or add a new one:
    - You typically create a new condition column.
    - You only rarely need to create a result column.
    - You can use only a predefined result column with a predefined pricing algorithm.
    - Don't remove a predefined result column. It might cause an error when Pricing runs the pricing algorithm.
    - If you add a new result column, then you must modify the pricing algorithm in the same way to make sure the pricing algorithm uses the new result. For example, if you add the Precedence column to the Strategy Assignment pricing matrix, then you must also modify the pricing algorithm so it sequences data according to precedence.
- Use the matrix type to identify the matrix class that Pricing uses for each pricing entity. For example, use the Pricing Segment matrix type to instruct Pricing to use the Pricing Segment matrix class when it processes a pricing segment.
- Use the Setup and Maintenance work area to set up a matrix type. For details, see *Manage Pricing Matrix Types*.

Here's a helpful way to think about columns:



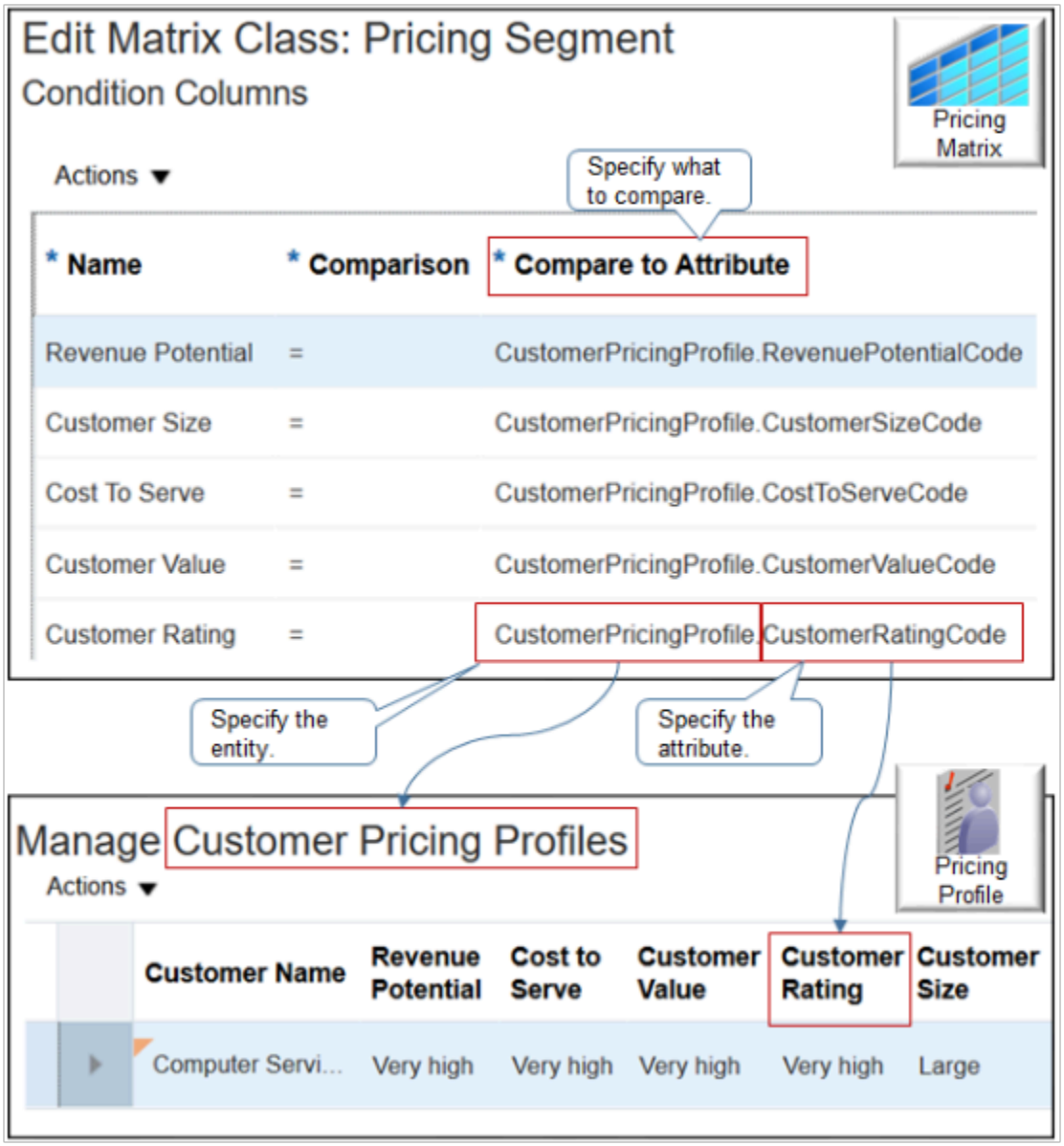
Note

- The Required, Allow Null, and Domain attributes control behavior in the Pricing Administration work area. For example, if you enable Allow Null in the Revenue Potential row, then you don't have to set a value for Revenue Potential when you set up the pricing segment in the Pricing Administration work area. The Domain attribute in the Revenue Potential row controls the values that you can select for Revenue Potential when you set up the pricing segment.
- The Comparison, Compare to Attribute, and Null Is Wildcard attributes control runtime behavior. For example, Pricing uses Comparison and Compare to Attribute at run time to assign the pricing segment according to the value that the Order Entry Specialist sets in the Customer attribute on the order header when creating a sales order in the Order Management work area.

Set the Compare-to Attribute

Specify the entity and the attribute in the entity that the condition compares. Each matrix class specifies a different set of entities and attributes to test.

In this example, each row in the class examines an attribute of the customer pricing profile because a pricing segment determines how to organize profiles into market segments:



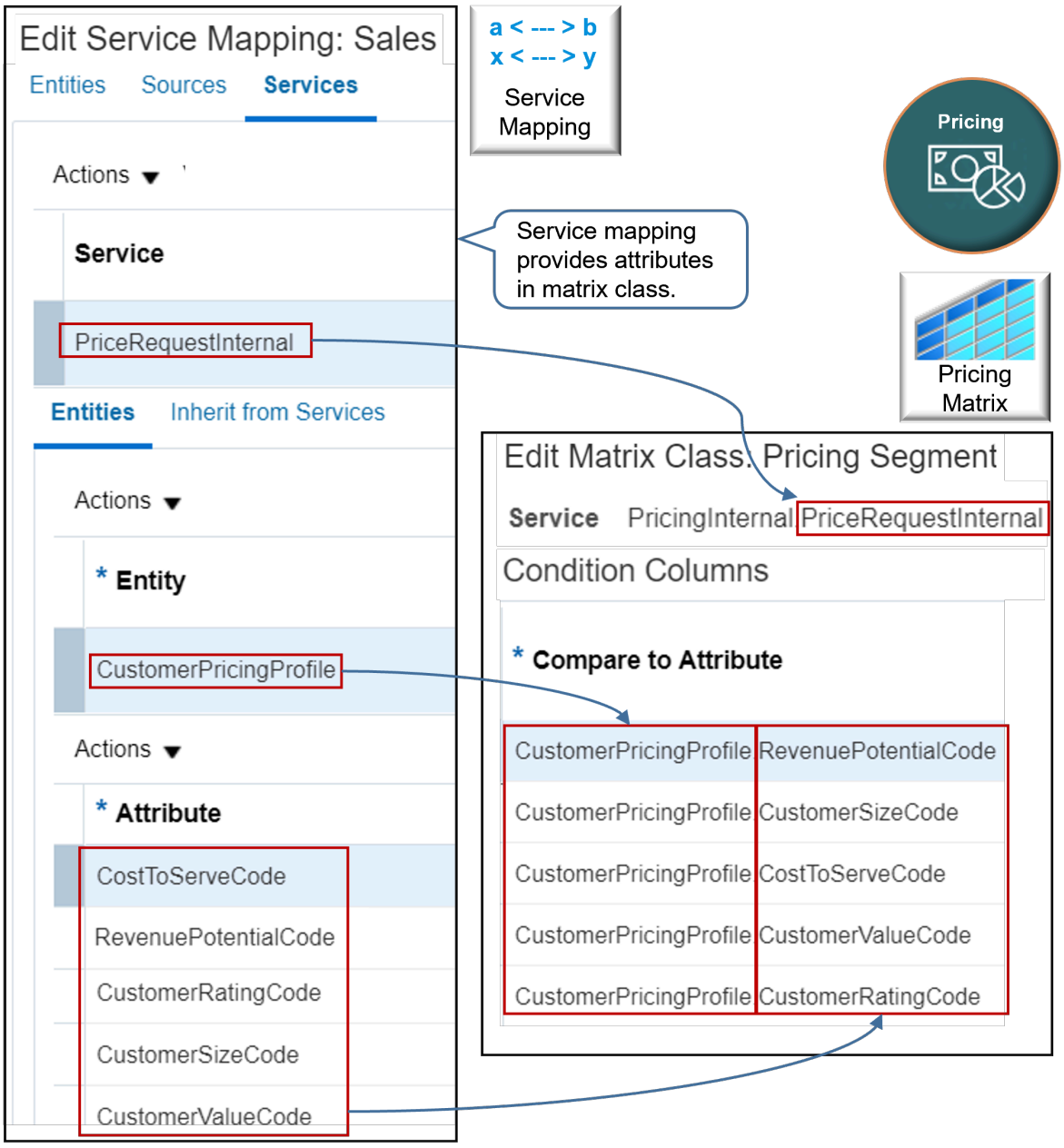
The Customer Rating condition column examines the value of the CustomerRatingCode attribute. CustomerRatingCode is part of the CustomerPricingProfile pricing entity.

Specify Values That Display as Choices

As an option, you can use the Domain attribute to specify the values that Pricing displays as choices for each attribute in the matrix. For details, see *Select Choices in Pricing Matrixes*.

Extend the Compare-to Attributes

For a more advanced set up, use a service mapping to determine the compare-to attributes that the matrix class uses:



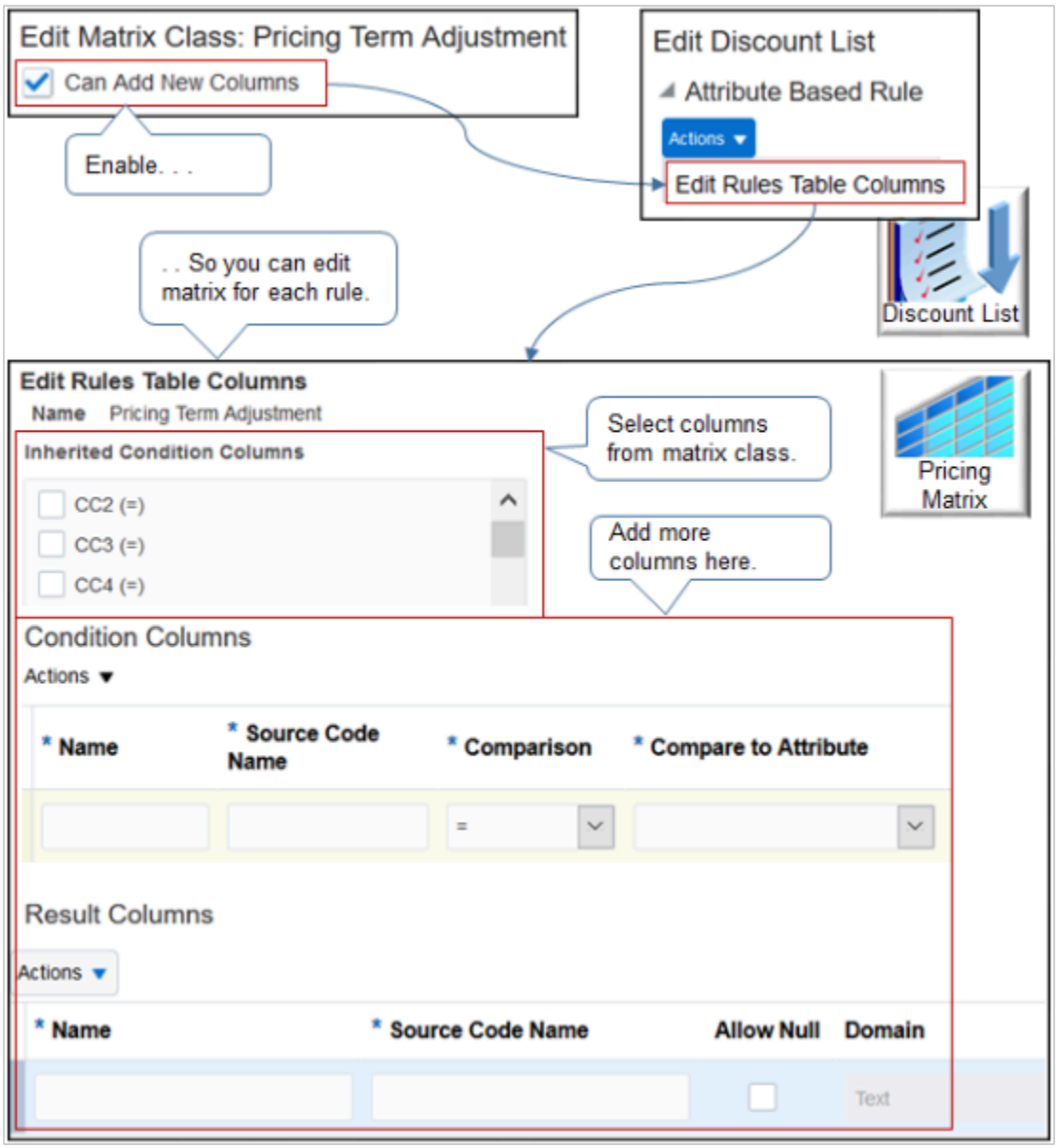
Note

- A service mapping provides the compare-to attributes in the matrix class.
- In this example, the Pricing Segment matrix class uses the PriceRequestInternal service of the predefined Sales service mapping.
- The CustomerPricingProfile entity of PriceRequestInternal contains the compare-to attributes.
- You can't delete predefined attributes in a predefined service mapping, but you can add new ones.
- You can use an attribute that exists in Pricing, such as CustomerValueCode.

- You can use an attribute that resides outside of Oracle Pricing, such as an attribute on the order header of a sales order from Oracle Order Management.

Make Rules More Flexible

Enable the Can Add New Columns option on the matrix class so you can edit the matrix for each new rule that you create:



Note

- Add conditions and results to meet your matrix requirements.
- Use this feature to refine your matrix at a more detailed level than what your matrix class specifies.
- The Inherited section lists columns that the pricing matrix inherits from the matrix class. To include a column from the matrix class, you can enable it.

- Use a mix of inherited columns and columns that you add only for the rule.
- Don't enable any inherited columns if you don't need them.

Examples

Examine a predefined matrix class:

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click **Pricing Segment**.
3. On the Edit Matrix Class page, notice the values:

Condition Columns	Result Columns
Revenue Potential Customer Size Cost To Serve Customer Value Customer Rating	Pricing Segment Precedence

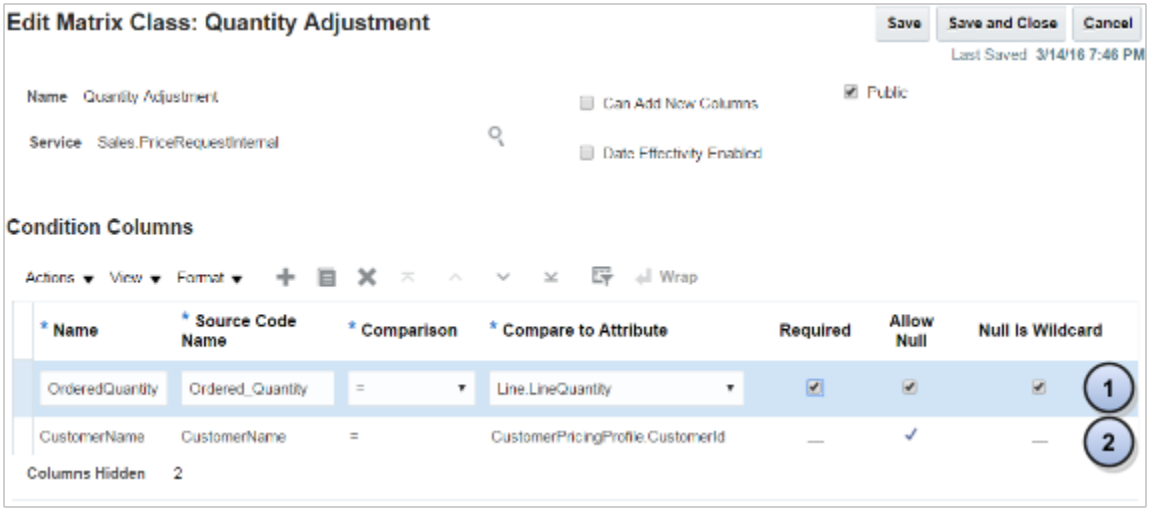
4. Click **Tasks > Manage Pricing Segments**.
5. Notice that the predefined Pricing Segment matrix class that you examined in step 3 determines the condition columns and the result columns that you can specify for a pricing segment.
6. On the Edit Matrix Class page, in the Revenue Potential row, notice the attributes:

Attribute	Value
Compare to Attribute	CustomerPricingProfile.CustomerRatingCode In this example, the pricing matrix gets values from the CustomerRatingCode attribute of the CustomerPricingProfile pricing profile.
Domain	Lookup:ORA_QP_CUSTOMER_RATING_VALUES The pricing matrix gets values from the ORA_QP_CUSTOMER_RATING_VALUES lookup.

These attributes determine the values that you can select for the Revenue Potential column in the pricing matrix on the Manage Pricing Segments page.

Another Example

Here's an example of a matrix class that you can use with a quantity adjustment:



Here are the conditions that this class implements:

1. If you don't enter a quantity for the sales order, or if you enter any quantity.
2. If you don't select a value for the customer on the sales order.

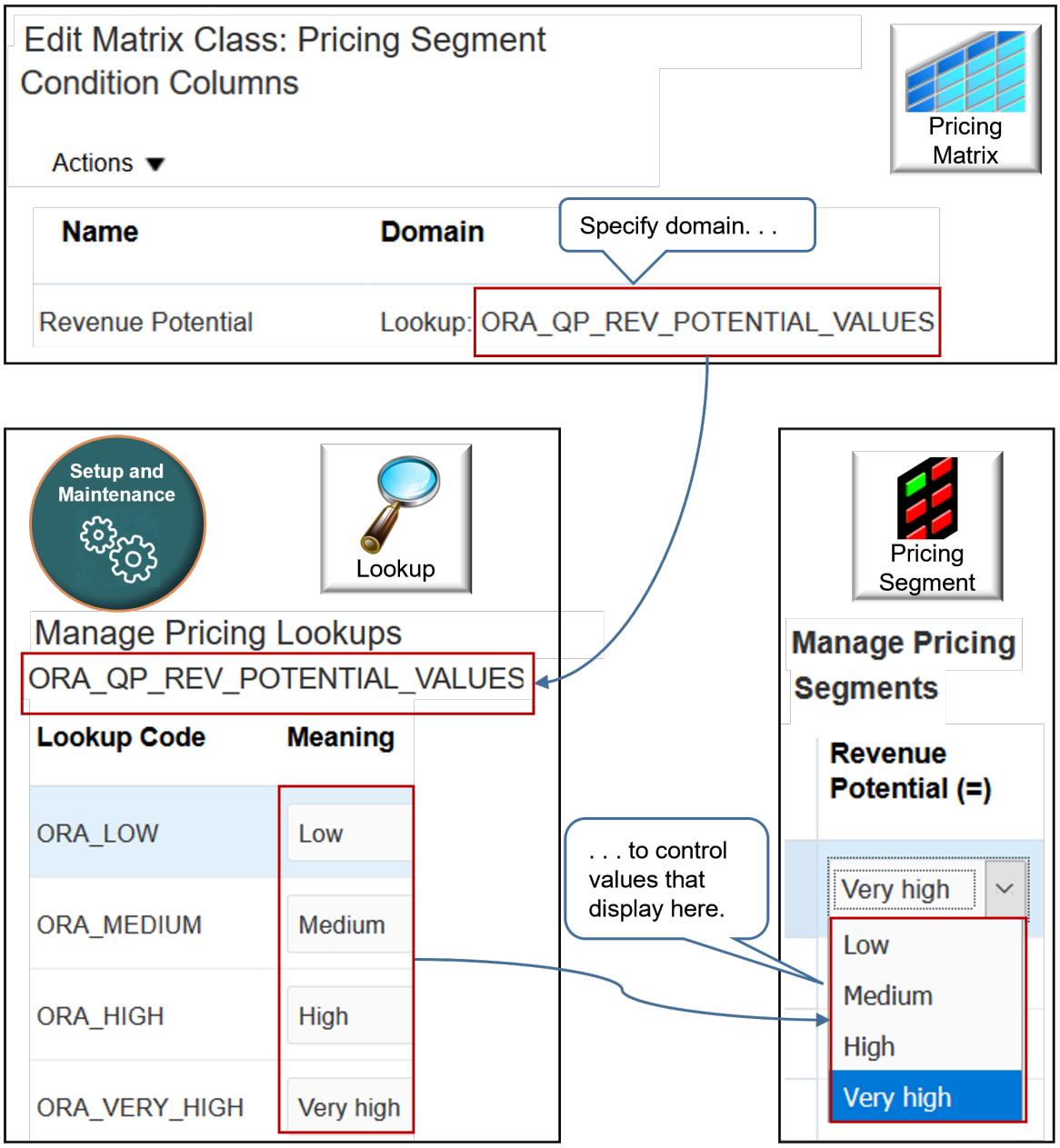
Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Matrices
- Manage Pricing Segments
- Pricing Rules

Select Choices in Pricing Matrixes

Use the Domain of the pricing matrix's class to specify where to get the values that the pricing matrix displays as choices for each attribute in the matrix.

Assume the Revenue Potential attribute of the Pricing Segment matrix class references the ORA_QP_REV_POTENTIAL_VALUES lookup:



Oracle Pricing gets values from the lookup when you use the Manage Pricing Segments page, then displays them as choices in the Revenue Potential attribute of the segment.

Specify Where to Get Values

Use the domain type to specify where to get the values:

Use This Domain Type	To Get the Value From
Lookup	A lookup that you specify on the Manage Pricing Lookups page. For example, to get values for approval status, select CSD_APPROVAL_STATUS.

Use This Domain Type	To Get the Value From
Item Extensible Attribute	An attribute that you specify on the Manage Item Attribute Groups and Attributes page in the Product Management work area. For details, see *Use Your Own Attributes in Pricing*.
Value Set	A set of values. The set of values can be Independent, Dependent, Subset, or Table.
View Object Query	An object that Pricing displays in the Pricing Administration work area.
Custom	Your set up in the Edit Column Domain Values dialog. Allows you to use your own values.
None	Text that you enter when you use the pricing matrix in the Pricing Administration work area.

View Object Query

View Object Query requires a more complex set up. Here are the attributes that you set:

Attribute	Description
Domain Type	Select View Object Query.

Application Module	Select the part of Pricing that displays the object that contains the value. For example, if the value:
	• Resides in a pricing matrix, then select MatrixDomainAM.
	• Is part of the pricing process, such as a matrix class or pricing algorithm, then select PricingProcess.
	Pay careful attention to the dot notated string. Make sure you select the string for the correct application.
	For example:
	`PricingProcessAM (oracle.apps.scm.doo.common.pricingIntegration.ApplicationModule.PricingProcessAM).doo`
	where
	• `doo` means Distributed Order Orchestration, which is an earlier name for Order Management. Use this choice for Order Management.
	Here's another example.
	`PricingProcessAM (oracle.apps.scm.fos.orchestrationProcesses.transferPrice.publicModel.pricingIntegrat`
	where
	• `fos` identifies the application as Financial Orchestration. Don't use this choice for Order Management.

Configuration	If Application Module doesn't display the value that you need, then enter it in the Configuration attribute. The dialog filters values that you can enter in Configuration according to the value you set for Application Module. To determine the values that are available, enter a percent symbol (%) in

Attribute	Description
	the Configuration attribute. For example, set Application Module to MatrixDomainAM enter % in Configuration, wait a moment, then click MatrixDomainAMLocal. In most situations, use MatrixDomainAMLocal when you set up a matrix class.
View Object	Select the view object that displays the value. For example, to get the value from an object that's part of the pricing strategy, select PricingStrategyPVO. PricingStrategyPVO displays objects that are part of these pages: • Manage Customer Pricing Profiles • Manage Pricing Segments • Manage Pricing Strategies • Manage Pricing Strategy Assignments • Manage Currency Conversion Lists For another example, select CustomerPVO. Use CustomerPVO to get attributes that describe the customer, such as PartyId, PartyName, and so on.
Key Attribute	Select the attribute that identifies the object that the result column must use. For example, select OrgId to get the value from the Business Unit attribute of the pricing strategy.
Display Attribute	Select the attribute that contains the value that the matrix must display.
View Criteria	As an option, set the view criteria to filter the data.
Data Type	Set the data type so it supports the attribute that you select in Key Attribute. For example, if you set Key Attribute to: • PartyId, then set Data Type to Number because PartyId contains numeric data. • PartyName, then set Data Type to Text because PartyName contains text data.
View Object Bind Variables	Add a join condition to filter the result.

Add a Join Condition

Use View Object Bind Variables to add a join condition. For example, assume you reference a view that displays these rows:

Row	Attribute X	Attribute Y
1	V1	R1
2	V2	R2
3	V3	R3

You can set up this condition:

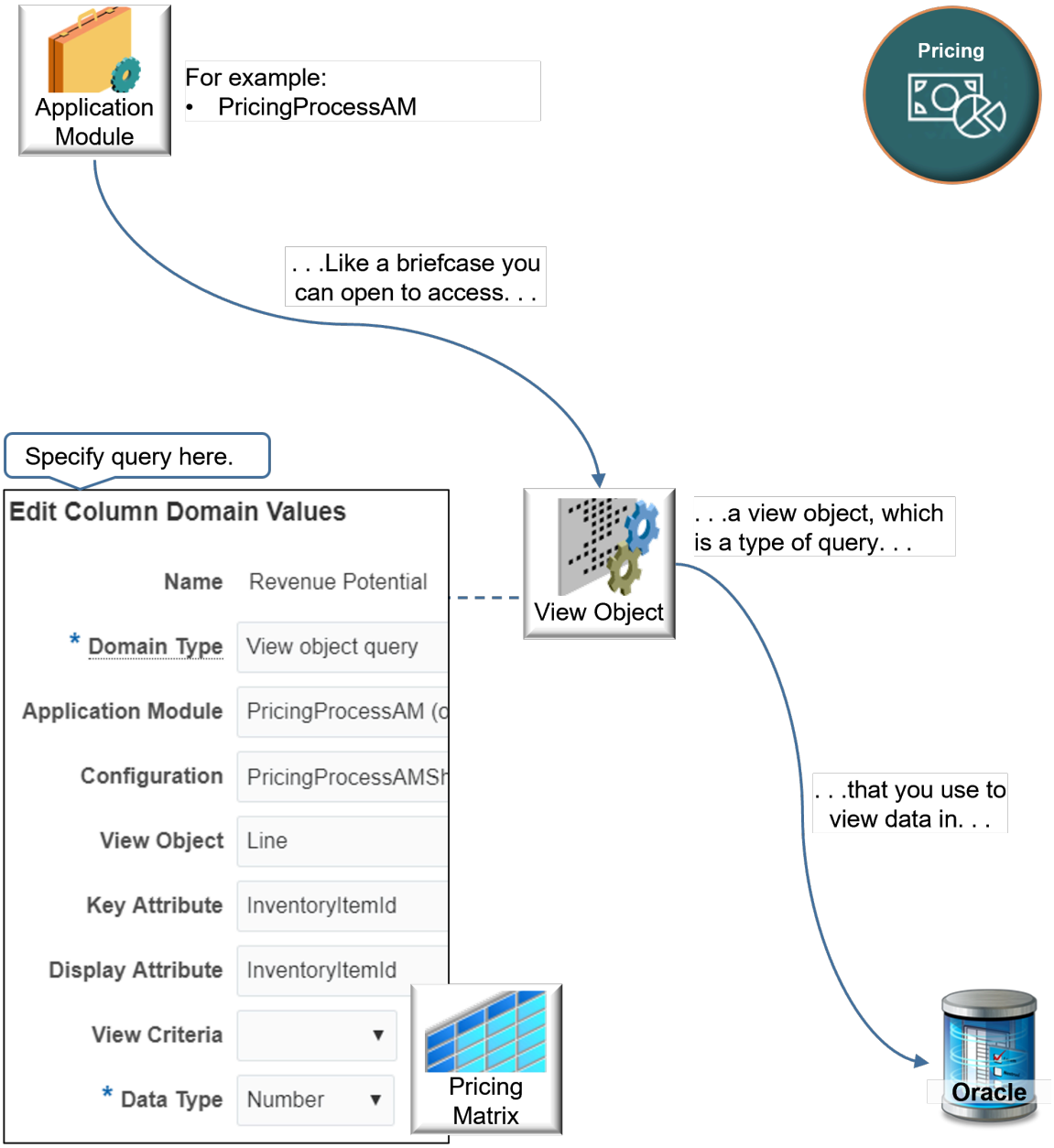
- If attribute y equals R1 and R2, then display only rows 1 and 2 in the Pricing Administration work area, such as on a price list.

What's a Public View Object?

A **view object** is a type of database query. It describes how Pricing views and updates data in the Oracle database.

A **public view object** is a type of view object that an application outside of Pricing, such as Order Management, uses to communicate with Pricing. Its public, which means the data that it gets from the database isn't private and only visible to Pricing.

For example, assume you developed a new item with a very large revenue potential, the Heart Wrist Watch that monitors heart rhythms. You need to use it as input to Revenue Potential on your pricing matrix. If the order line contains the wrist watch, then you need to set Revenue Potential to Very High:



Notes

- An application module is a like a briefcase that you can open to access the Oracle database. PricingProcessAM (Pricing Process Application Module) is an example of an application module.
- Use the view object to query the Oracle database.
- In the Condition Columns area of the pricing matrix, In the Domain column, click the pencil, then set the Domain Type to View Object Query.
- Use the dialog to specify how to use view object to query the Oracle database.

- Here are the settings that you use for this example:

Attribute	Value
Domain Type	View Object Query
Application Module	PricingProcessAM (oracle.apps.scm.doo.common.pricingIntegration.ApplicationModule.PricingProcessAM)
Configuration	PricingProcessAMShared
View Object	Line The order line contains the item, so select Line.
Key Attribute	InventoryItemId InventoryItemId identifies the item that the order line contains.
Display Attribute	InventoryItemId
View Criteria	Leave empty.
Data Type	Number InventoryItemId is a number data type.
View Object Bind Variables	Leave empty.

Value Set

Get values for an attribute on a pricing matrix according to a value set. Use the Domain Type attribute in a pricing matrix or matrix class to specify the value set:

- Use values from a value set in a matrix rule to adjust pricing according to an attribute on a price list or discount list.
- Use values from a value set to determine the pricing segment or pricing strategy assignment.
- Use the Value Set domain type in the condition column of a pricing matrix or for a matrix class so you can use values from a value set as part of the condition.

You can also use it when you manage your pricing rules through:

- **REST API**. Manage price lists, discount lists, pricing segments, or pricing strategy assignments.

For details and examples, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Document Prices**.

- **File-based data import**. Import and update a batch of price lists or a discount list.
- **ADF Desktop Integration**. Manage a price list or discount list in a spreadsheet.

Make sure you set up the value set before you use it in your matrix.

Example One

Assume you want to use a value set named My Value Set.

1. Go to the Setup and Maintenance work area, click **Tasks**, then search for and open the Manage Value Sets task.
2. On the Manage Value Sets page, click **Actions > Create**, then set the values:

Attribute	Value
Value Set Code	QP_MY_VALUE_SET
Module	Pricing Use the advanced search on this attribute. Search for Contains Pricing, then select the result that has APPLICATION for the Module Type and QP for the Module Key.
Validation Type	You must use one of these: o Independent o dependent o Subset o Table If you select Table, then make sure you set the: o Value Column Length attribute to a value that's less than 401 o ID Column Length attribute to a value that's less than 101
Value Data Type	You must use one of these: o Character o Number

Example Two

Assume you have an object named AK_LEGAL_ENTITY, its Value Data Type is Character but the ID Column and the ID Column Type for AK_LEGAL_ENTITY is Number. In the condition for your matrix class you need to use AK_LEGAL_ENTITY as a Number data type, not a Character data type.

You can create a value set with these values:

Attribute	Value
Value Set Code	AK_LEGAL_ENTITY
Module	Oracle Middleware Extensions for Applications
Validation Type	Table
Value Data Type	Character
FROM Clause	HR_LEGAL_ENTITIES
Value Column Name	NAME
Value Column Type	VARCHAR2
Value Column Length	50
ID Column Name	ORGANIZATION_ID
ID Column Type	NUMBER
ID Column Length	40

More Examples

Have a look at some other detailed examples that set the domain:

- *Assign Pricing Strategy According to Precedence*
- *Create Price Lists for Each Customer*
- *Use Extensible Flexfields with Pricing*

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Pricing Matrices
- Manage Pricing Segments
- Pricing Rules

Manage Pricing Matrices

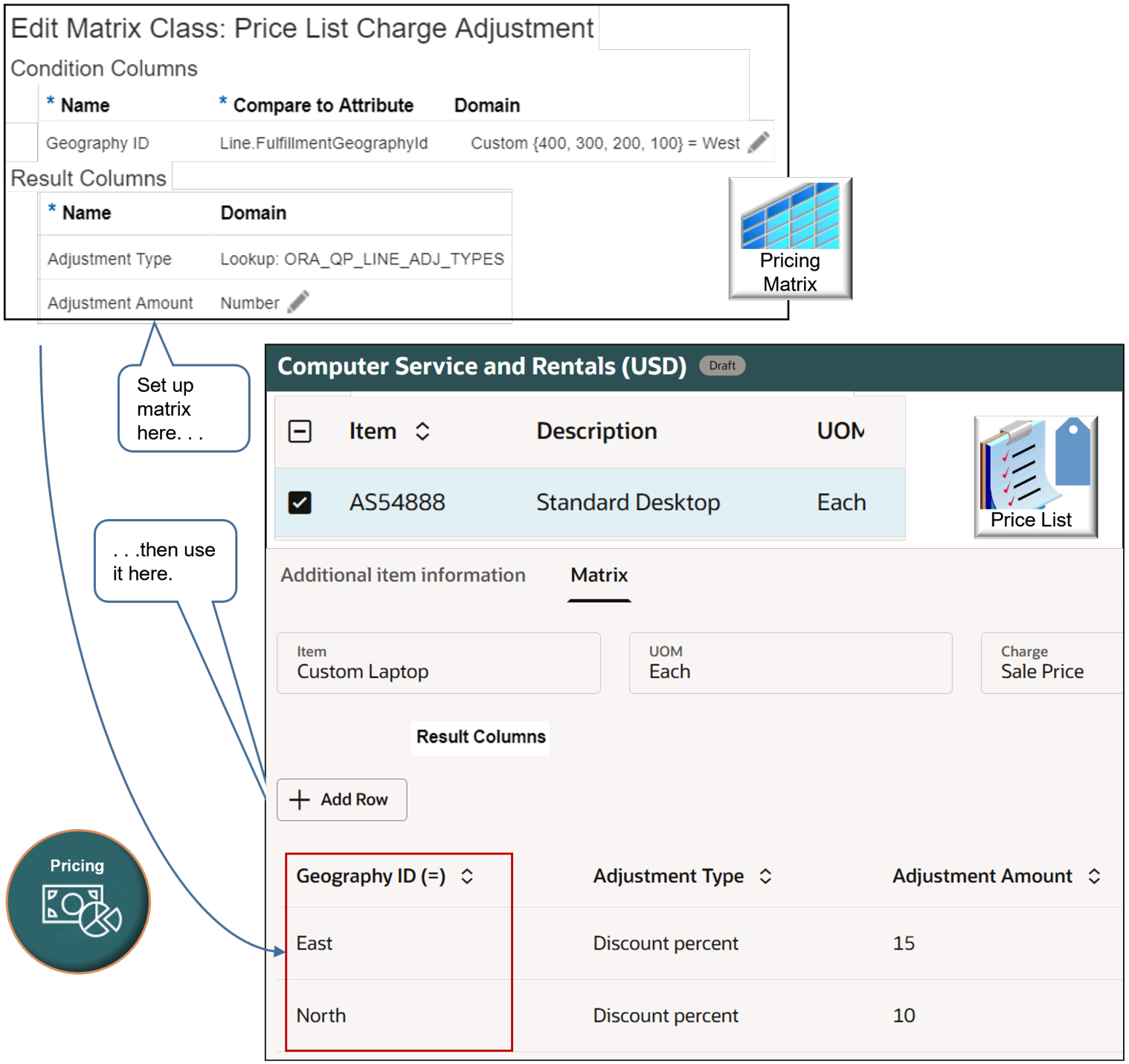
In this example you modify a predefined pricing matrix so you can add adjustments to a pricing rule according to the value in the Customer Region attribute.

Customer Region	Adjustment
North	10% discount
South	0% discount

Customer Region	Adjustment
East	5% discount
West	15% discount

The Customer Region is the condition, and the Adjustment Type of **% discount** and Adjustment Amount are the results. For example, if the Customer Region contains North, then reduce the price by 10%.

Here's a summary:



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

Manage a matrix class:

1. Set up the condition column.
2. Test your set up.

Set up the Condition Column

1. Go to **Home > Order Management > Pricing Administration > Tasks > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click **Price List Charge Adjustment**.

   You can modify the predefined Price List Charge Adjustment matrix class. It determines the conditions and results that you can specify in the pricing matrix that you add to a price list. Oracle Pricing comes predefined with a number of matrix classes. To reduce maintenance, modify a predefined matrix class to meet your business requirement rather than create a new one.
3. On the Edit Matrix Classes page, notice the attributes:

Attribute	Description
Service	Specify a service that a service mapping references. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.
Public	Allow an application other than Oracle Pricing to use the matrix class.
Date Effectivity Enabled	Allow the pricing rule that uses the matrix class to have a start date and an end date. If you add a check mark to the Date Effectivity Enabled option, then the page in Administering Pricing that displays the pricing rule will display the Rule Start Date and the Rule End Date columns, and you'll be able to set those dates on your rule. If you don't add a check mark, then you can't set those dates and the rule will always be in effect until you manually delete it.
Can Add New Columns	Allows you to add new columns to a pricing rule.

4. Scan the Condition Columns area and the Result Columns area.

   These areas determine the conditions and the results that the adjustment matrix displays when you add a pricing matrix. Examine an example of this matrix. For details, see *Adjust Price for Pricing Rules*.
5. In the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Geography ID
Source Code Name	GeographyID
Comparison	=
Compare to Attribute	Choose the attribute that Pricing examines to determine whether the condition is true. For this example, choose Line.FulfillmentGeographyId.

Attribute	Value
	For example, assume a sales order in Oracle Order Management contains an attribute that stores the value for the FulfillmentGeographyId attribute on the order line. At runtime, if Geography ID equals the value that FulfillmentGeographyId contains, then Pricing applies the result that the adjustment matrix specifies.
Required	Leave empty. If you add a check mark, then you must provide a value for the attribute that you choose in Compare to Attribute when you create the pricing rule in Pricing Administration.
Allow Null	Leave empty If you add a check mark, then Pricing doesn't require you to provide a value for this attribute when you use the matrix during set up in Pricing Administration. It allows an empty value or the string **`Null`** in the database.
Null is Wildcard	Leave empty. If you add a check mark to: o Null is Wildcard. Pricing allows the user to provide any value for the attribute at runtime, and Pricing will use that value during the comparison. o Allow Null and to Null is Wildcard, Pricing allows no value, or any value.

6. Set up the domain:
   - In the row you just added, in the Domain column, click **Edit Domain**.
   - In the Edit Column Domain Values dialog, set the values:

Attribute	Value
Domain Type	Custom
Data Type	Text
Default Value	For this example, enter **`West`**. Pricing will display the value you enter in the Condition column of the pricing matrix, by default.
Default Is Fixed Value	Leave empty. If you add a check mark, then Pricing doesn't allow you to modify the default value in the pricing matrix.

Attribute	Value

- Click **Add Row** four times, then add values:

Domain Value	Display Value
100	North
200	South
300	East
400	West

- Click **OK > Save**.

Test Your Set Up

1. Create a price list, then add item AS54888 to it.

   See *Manage Price Lists*.
2. Select the line that has your item, then click **More Actions > Add Matrix Pricing**.
3. Click **Add Columns**, select Geography ID (=), then click **Add Columns**.
4. Click **Add Rows**, then add them:

Geography ID	Adjustment Type	Adjustment Amount
North	Discount Percent	10
East	Discount Percent	5
West	Discount Percent	15

5. Click **Save > Done**.

Related Topics

- Manage Service Mappings
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Pricing Rules
- Pricing Matrices
- Manage Price Lists

Manage Pricing Matrix Types

Map a matrix class to a matrix type to make sure each pricing entity in the Pricing Administration work area references the correct matrix class.

Use the Manage Matrix Types page to assign one matrix class to a matrix type. For example, Pricing maps the predefined Price List Charge Adjustment matrix class to the predefined Price List Charge Adjustment matrix type. This mapping makes sure the condition columns and result columns of the Price List Charge Adjustment matrix class display correctly in the Price List page in the Pricing Administration work area, and makes sure you use these columns when you create a pricing rule that adjusts price according to an attribute.

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

Manage a pricing matrix type:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Matrix Types task.
2. On the Manage Pricing Matrix Types page, examine the pricing matrix types that come predefined with Pricing.

   To reduce maintenance, use a predefined pricing matrix type instead of creating a new one.

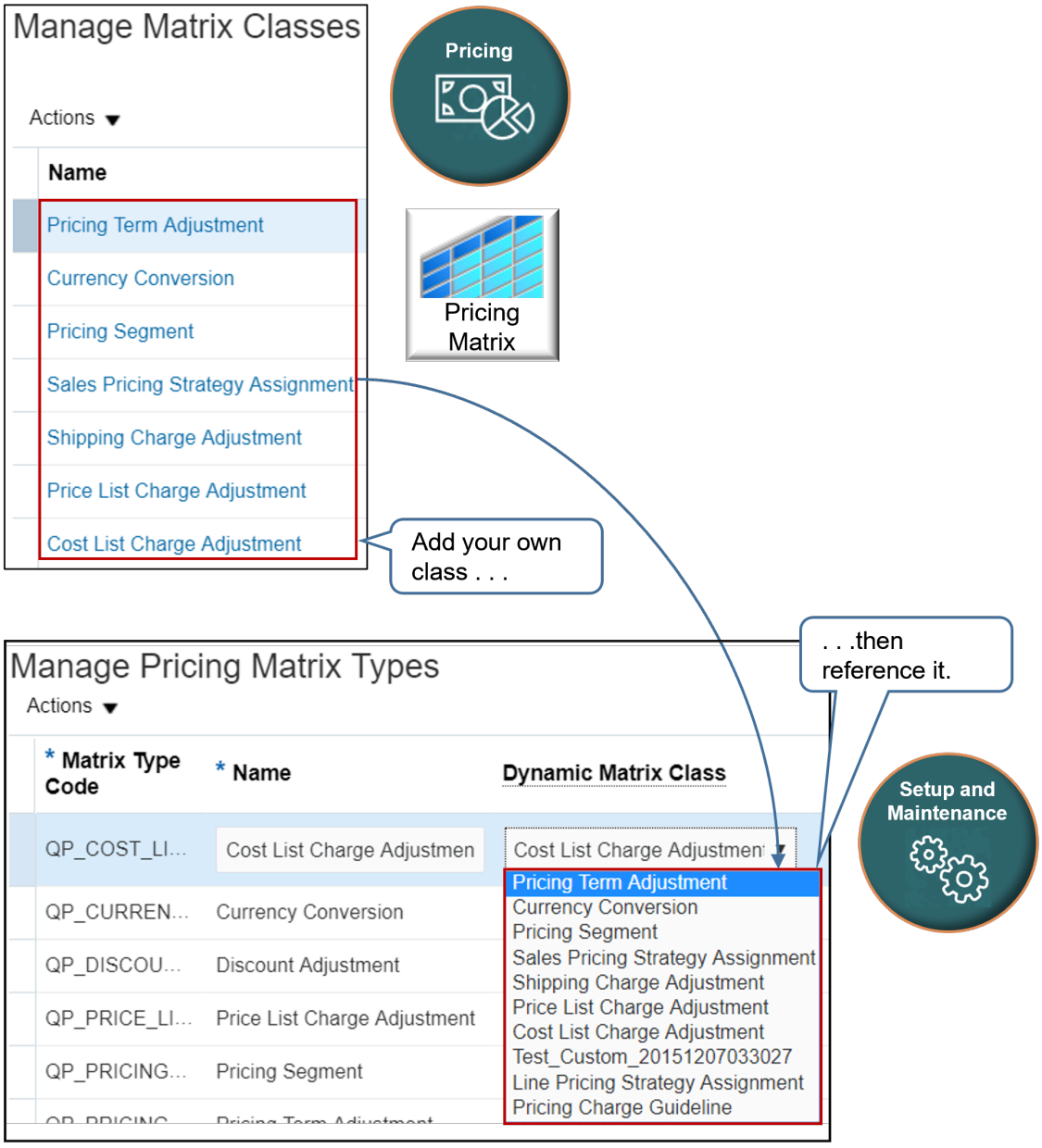
   Examine the mapping between each matrix type and matrix class so you know which matrix class where Pricing will add the attribute. You must also know the details of the attribute, its domain, and mappings within the matrix class. After implementing the matrix class, you can set up the matrix rules that will use the new attribute.
3. If you can't locate a pricing matrix type that meets your requirements, then click **Actions > Add Row**, then set the values:

Attribute	Value
Matrix Type Code	Enter text that Pricing can use to reference this pricing matrix type. Pricing assigns each pricing entity that it associates with a pricing matrix to this matrix type code. A `QP_ (QP underscore)` prefix identifies the predefined pricing matrix types.
Name	Enter text that describes the pricing matrix type. Pricing displays the text you enter in the Pricing Administration work area: o Use headline capitalization, such as My Pricing Matrix Type. o Use the same text that you choose for the Dynamic Matrix Class. Using the same text helps to visualize the mapping.

Attribute	Value
Dynamic Matrix Class	Choose the Dynamic Matrix Class that identifies the conditions, results, and domain values. Dynamic Matrix Class displays the list of matrix classes from the Manage Matrix Classes page in the Pricing Administration work area. Use the Manage Matrix Classes page to create your own matrix class, then use Manage Pricing Matrix Types to map the type to the class.
Allow Multiple Matrixes	Add a check mark to allow more than one matrix for the pricing entity. For example, Pricing comes predefined to include: o More than one matrix for pricing strategy assignments. o Only one matrix for pricing segments.
Active	Add a check mark to make the matrix is available at runtime.

4. Click **Save**.

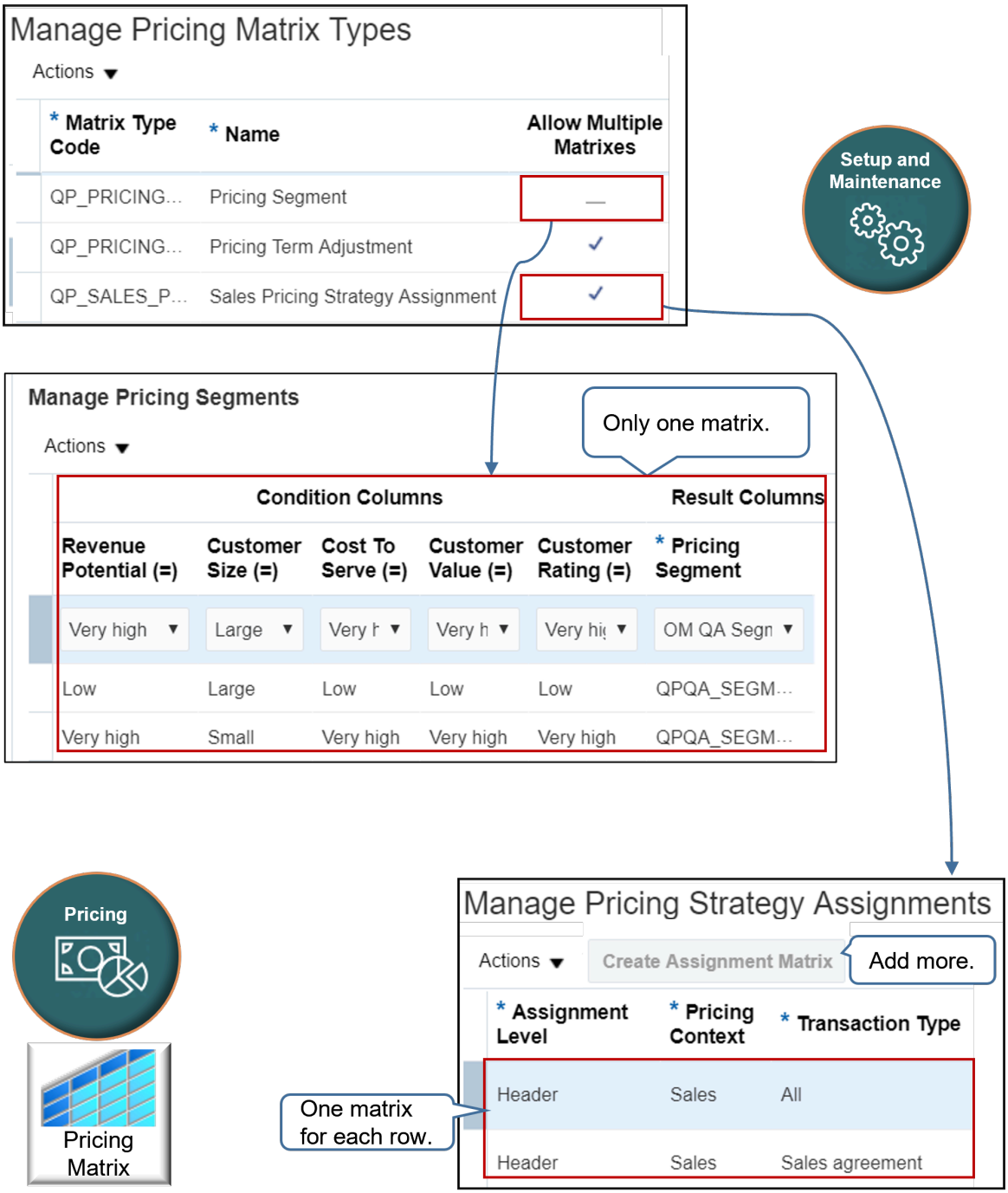
How Dynamic Matrix Class works:



Note

- Use Dynamic Matrix Class to choose a matrix class from the Manage Matrix Classes page of the Pricing Administration work area. If no predefined class meets your needs, then add your own.

How Allow Multiple Matrixes works:



Enable the Allow Multiple Matrixes option when you need more than one matrix for an entity. For example:

- Allow Multiple Matrixes is disabled for the predefined Pricing Segment matrix type. So the Pricing Segment entity in the Pricing Administration work area includes only one matrix, and you can't add more.
- Allow Multiple Matrixes is enabled for the predefined Sales Pricing Strategy Assignment matrix type. So the Pricing Strategy Assignment entity in the Pricing Administration work area includes more than one matrix. It includes a separate matrix for each assignment. Pricing Administration also displays the Create Assignment Matrix button that you can use to add a matrix.

Related Topics

- Pricing Matrices
- Pricing Rules
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing

Modify Algorithm When Using Matrix

If you modify the result column of a pricing matrix or use an extensible flexfield with a pricing matrix, then you must modify an algorithm step.

Assume you add an extensible flexfield to a discount adjustment. Here's the set up you need to do:

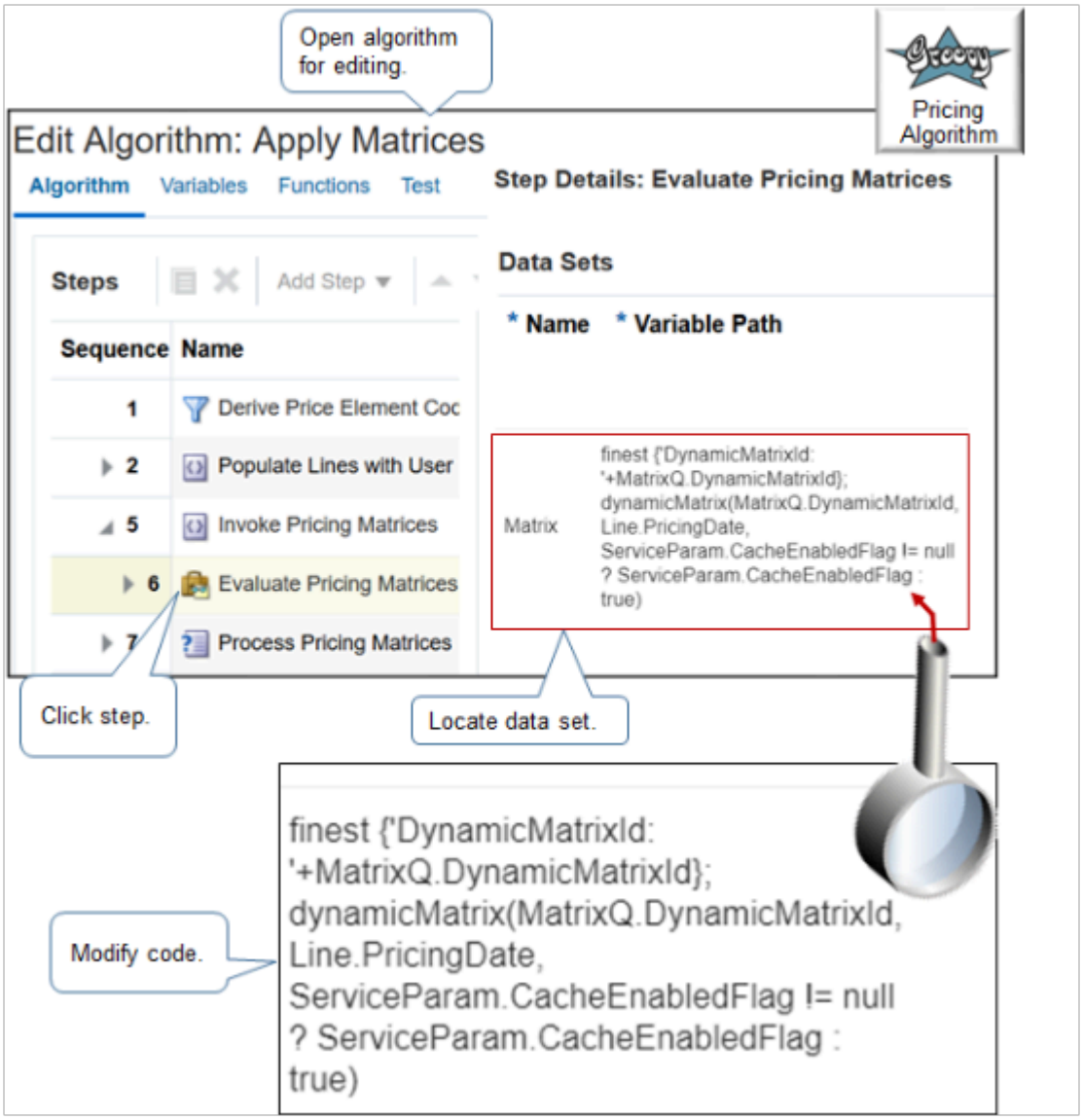
1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, search for, then open the algorithm you must edit:

If You Use This Matrix Type	Then Modify This Algorithm Step	In This Pricing Algorithm
Pricing Segment	`Derive Pricing Segment`	`Get Sales Pricing Strategy`
Sales Pricing Strategy Assignment	`Get Header Strategy`	`Get Sales Pricing Strategy`
Price List Charge Adjustment Discount Adjustment Pricing Term Adjustment	`Evaluate Pricing Matrices`	`Apply Matrices`
Line Strategy	`Get Pricing Strategy`	`Get Line Pricing Strategy`
Pricing Charge Guideline	`Evaluate Pricing Matrices`	`Apply Matrix Rules for Guidelines`

If You Use This Matrix Type	Then Modify This Algorithm Step	In This Pricing Algorithm

Pricing runs the predefined step at run time to process the pricing matrix.

For this example, search for, then open `Apply Matrices` for editing:



3. On the Edit Algorithm page, click the **step** that contains `Evaluate Pricing Matrices` in the Name column.

4. In the Step Details area, click the **step** that contains Matrix in the Name Column, then examine the code in the Variable Path column.

```
finest {'DynamicMatrixId: '+MatrixQ.DynamicMatrixId}; dynamicMatrix(MatrixQ.DynamicMatrixId,
 Line.PricingDate, ServiceParam.CacheEnabledFlag != null ? ServiceParam.CacheEnabledFlag : true)
```

Here's the format of the path to the Matrix data set.

```
dynamicMatrix(MatrixId, effectivityDate, enableCache)
```

where

Parameter	Description
MatrixId	Identifies the pricing matrix to evaluate.
effectivityDate	Date to compare against the start date and end date of the pricing matrix.
enableCache	Contains: - **True**. Cache the matrix rules. - **False**. Don't cache.

Most steps come with other predefined data sets, such as:

- Charge
- Line
- Header
- LineAttribute
- ItemExtensibleAttribute
- ServiceParam

To reduce maintenance, use a predefined data set. If none of the predefined data sets meet your needs, then add a new one.

If you use an item extensible attribute or extensible flexfield in a condition column, then you must add a data set. For example, here's a data set you could use to support an extensible flexfield that you add on the order header:

Name	Variable Path	Cardinality	Data Set Join
PricingHdrEFF_Custom	PriceRequest.Name where	Zero or one	[HeaderIdCustom: {Header.HeaderId}]

Name	Variable Path	Cardinality	Data Set Join
	○ Name is identical to the data set name. For this example. PriceRequest.PricingHdrEFF_Custom		

5. In the Step Details area, scroll down, locate the First Row Actions section, then examine the code:
    - The algorithm runs this code for the first record that the query returns.
    - Pricing runs this code for each pricing matrix type except for the pricing guideline. It runs the Each Row Action for the pricing guideline.
    - Keep these points in mind during setup, maintenance, and troubleshooting.

Here's the complete code for the first row action. You must modify it in some situations. For details, see *Manage Pricing Algorithms*.

```
finest('Processing Matrix RuleId: '+Matrix.DynamicMatrixRuleId)
MatrixQ.AdjustmentBasisId = Matrix.AdjustmentBasisId
MatrixQ.AdjustmentValue = Matrix.AdjustmentAmount
MatrixQ.AdjustmentTypeCode = Matrix.AdjustmentTypeCode
MatrixQ.DynamicMatrixRuleId = Matrix.DynamicMatrixRuleId
MatrixQ.ConditionString = Matrix.MatchConditions.join(' ')
MatrixQ.ResultString = Matrix.MatchResults.join(' ')


/*if ( Matrix.AdjustmentTypeCode?.contains('PERCENT') )
 Basis = (BasisComp.locate([PriceElementCode:Matrix.AdjustmentBasis]))?.UnitPrice?.Value
finest('Basis Value: ' + Basis)
MatrixQ.AdjustmentValue = pricingUtil.computeUnitAdjustment(Matrix.AdjustmentTypeCode, Basis,
 Matrix.AdjustmentAmount, Charge.RunningUnitPrice?:0)
MatrixQ.DynamicMatrixRuleId = Matrix.DynamicMatrixRuleId

// currency conversion
if (MatrixQ.FromCurrencyCode!=Charge.CurrencyCode ) {
 if ( 'ERROR'==ConvRate?.MessageTypeCode ) {
 finest('creating line message')
 Line.MessageTypeCode = 'ERROR'
 Charge.MessageTypeCode = 'ERROR'
 Term.MessageTypeCode = 'ERROR'
 msg =
 Message.locate([ParentEntityCode:'LINE',ParentEntityId:Line.LineId,MessageText:ConvRate.PrcErrorMessage])
 if ( msg==null ) {
 // create new error message for Line
 msg = Message.insert([PricingMessageId:getNextId()])
 msg.MessageName = ConvRate.PrcMessageName
 msg.MessageText = ConvRate.PrcErrorMessage
 msg.ParentEntityCode = 'LINE'
 msg.ParentEntityId = Line.LineId
 msg.MessageTypeCode = ConvRate.MessageTypeCode
 }
 }
 else {
 MatrixQ.AdjustmentValue *= ConvRate.ConversionRate?:1
 finer('\tConverted currency, 1 '+MatrixQ.FromCurrencyCode+' = '+ConvRate.ConversionRate?:1+'
 '+Charge.CurrencyCode)
 }
}
```

```
// end currency conversion
*/
```

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Matrices
- Manage Pricing Segments
- Pricing Rules

Troubleshoot Pricing Matrixes

Troubleshoot problems with a pricing matrix.

Trouble	Shoot
Pricing doesn't apply the matrix rule.	Make sure the value that the Domain references identifies the same data that the Compare to Attribute condition column references. For example, If Compare to Attribute equals Header.CustomerId, then get the domain value from CustomerId. Don't get it for Customer Name. Make sure you use the correct data type. For example, if the service mapping uses the Data data type, then make sure the data that your Compare to Attribute references also uses the Data data type.
I add a Customer Name condition to a matrix class and assign it to a pricing segment so I can use a different price for each customer. At runtime, pricing seems to work as I expect for customers that I have created on the Manage Customers page in the Setup and Maintenance work area, but not for customers that I create through import.	Its possible the customers that you create through import aren't getting populated correctly in the Oracle database. Run the *Maintain Party and Location Current Record Information* scheduled process. This process updates people, organizations, and locations that are currently in the Oracle database.

I encounter an error when I edit a rule for a discount list. It is similar to.

```
An application error occurred. See the incident log for more information. View row with key
 oracle.jbo.Key[null ] is not found in __DimensionVO__For__Reference__For_DCL__.
```

Note

- This problem might happen because Pricing can't find a condition or dimension in a pricing matrix or matrix class.
- Don't delete a condition column in a pricing matrix or matrix class after you finish setting up your matrix rule.
- If your pricing matrix references a dimension in the matrix class, then don't delete that dimension.

- If you add your own dimension, then don't delete it, or if you must delete it, then make sure you modify your pricing matrix so it doesn't reference that dimension.
- Follow these recommendations for all pricing entities.

Assume you create a matrix class for a price list adjustment. You add dimensions Color, Size, and Style to the class.

Assume you create a price list for three items, and you set Adjustment Type to Discount Amount for each price list:

- Price list for item A:

Color	Size	Adjustment Amount
Red	M	10
Red	L	15

- Price list for item B:

Color	Style	Adjustment Amount
Red	001	10
Red	002	15

- Price list for item C:

Color	Season	Adjustment Amount
Red	Winter	10
Red	Spring	15

The Season condition dimension for the item in price list C isn't part of the matrix class but is part to the pricing matrix. So, if you have a rule that references the Season condition, then don't delete the Season condition.

Identify Matrix Type That Contains the Error

Identify the matrix type that might contain the error. Pricing displays a slightly different message according to the matrix type that its processing when the error happens:

Message	Predefined Matrix Type
`Pricing didn't apply attribute pricing to the transaction.`	Price List Charge Adjustment

Message	Predefined Matrix Type
`An error occurred while determining the pricing segment for the transaction.`	Pricing Segment
`An error occurred while determining the pricing strategy for the transaction.`	Sales Pricing Strategy Assignment Line Strategy
`Pricing didn't apply attribute pricing to the charge.`	Price List Charge Adjustment
`Pricing didn't apply attribute pricing to the adjustment.`	Discount Adjustment
`An error occurred while determining a guideline rule for charge {CHARGE_NAME} in guideline {GUIDELINE_NAME}.`	Pricing Charge Guideline

After you identify the matrix class, make sure the Compare to Attribute condition column in the class references the correct data type.

Other Fixes

Examine the Latest Version of the Pricing Algorithm

For example, if the problem happens with the Price List Charge Adjustment matrix type, then examine the `Apply Matrices` pricing algorithm, and make sure the `Evaluate Pricing Matrices` step contains all the required data sets.

For another example, if you add extensible flexfield x on the order header, and if you add a Source Order dimension, then make sure the dimension references extensible flexfield x, and make sure you add flexfield x to the data set.

Examine Modifications in the Pricing Algorithm

- Make sure your modifications don't contain typographical errors, such as an incorrect spelling.
- Make sure you correctly set the Data Set Join.
- Make sure the Path uses PriceRequest.EntityName, where EntityName matches exactly the value that the service mapping uses.
- Make sure the Alias uses the same value as the EntityName.

Related Topics

- Pricing Rules
- Manage Price Lists
- Manage Cost Lists
- Manage Currency Conversion Lists
- Manage Discount Lists

Covered Items

Pricing for Covered Items

Set up pricing rules that affect coverage on a price list. Manage the rules between each coverage item and each covered item.

A coverage item is an item that adds value by providing coverage for a covered item. For example, a six month warranty is a coverage item for a laptop computer, and the laptop computer is the covered item:

- Create a pricing rule for a coverage item that covers all items on a price list.
- Create a one-time charge or a recurring charge.
- Use an amount or a percent of the covered item price in each rule.
- Create one or more charges for each rule between the coverage item and the covered item. Set each charge as a one-time charge or a recurring charge.
- Calculate each coverage pricing rule according to an amount or to a percent of the covered item price for a duration. For example, the price for the Gold Warranty coverage for a laptop for three years is $10 for each year, with a total amount of $30 over the full three year period.
- Import and export coverage pricing rules from price lists.

Note

- Pricing supports coverage pricing through an integration with Oracle Service Contracts. For details about how to set up service contracts, see *Implementing Enterprise Contracts*.
- If you set the Service Duration Type attribute to Variable on a subscription, then you can apply a discount on the subscription. For details, see *Set Up Coverages for Sales Orders*.
- You can't apply a discount on a coverage.

Differentiate Pricing for Coverage

You can sell an item and also sell coverage for an item. In some situations, you might need Pricing to price a coverage item for a warranty or support differently than it prices a coverage item for a durable good. The coverage item price might depend on the item that it covers. For example, assume a company that builds medical devices sells items to medical clinics:

Covered Item	Coverage Item
MRI machine	Technical support for an MRI machine
CT scanner	Technical support for a CT scanner

A technician typically applies a more sophisticated skill set when supporting an MRI machine than when supporting a CT scanner. Pricing allows the medical device company to differentiate the price of the support that it offers for the MRI machine from the support that it offers for the CT scanner.

Calculate One Time and Recurring Charges for Covered Items

Here's the calculations that Pricing uses to calculate the charge for each covered item:

Calculate One-time Charge	Calculate Recurring Charge
To calculate the extended amount, multiply the one-time charge by the quantity of units of the covered item. For example, if the one-time charge is 25, and if the quantity is 4, then the one-time charge is 100. Note that a one-time charge applies to the entire duration of the coverage.	Calculate the recurring charge for each covered quantity for each primary UOM (unit of measure), when the UOM is Time: 1. If the duration UOM isn't equal to the frequency's UOM, then set the duration UOM to the frequency's UOM. For example, if the duration UOM of the coverage is 1 Year, and if the frequency's UOM is 6 Months, then set the duration UOM to 6 Months. 2. Multiply the recurring charge by the quantity of units of the covered item. 3. Multiply the result of step 2 by the duration. For example, if the unit price for the monthly charge is $10, and if the quantity of units is 1, and if 12 months have passed, then the result is: • Extended amount of $10 ($10 multiplied by 1, where 1 is the quantity of units of covered items) • Total coverage amount of $120 ($10 extended amount multiplied by 1 unit multiplied by 12 months)

Related Topics

- Manage Price Lists
- Set Up Coverages for Sales Orders

Manage Pricing for Covered Items

In this example, you set up Oracle Pricing to calculate three years of warranty service for a laptop computer, starting on January 1, 2017, billed monthly at the rate of 10% of the sale price.

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

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Bases**.
2. On the Manage Pricing Bases page, set the pricing basis to use for the covered item.

   More than one charge might exist for a covered item. You must set up a pricing basis that determines the charges and price element, such as list price or net price, to use for the percent calculation.

   For example, assume your covered item includes a separate charge for the sale price, but doesn't include charges for the monthly recurring sale price or a one-time recycling fee. You must specify which of these charges to use as the basis when calculating the coverage charge for the covered item. In this example, the price of the coverage is 10% of the list price of the covered item.

3. Click **Actions > Create**, then set the values. This basis specifies the list price as the charge to evaluate. For details, see *Manage Pricing Bases*:

Attribute	Value
Name	Coverage_Basis_for_Desktop_PC
Usage	Coverage Basis
Price Element	List Price
Price Type	Recurring
Charge Type	Sale
Charge Subtype	Price
Frequency	Month
Active	Contains a check mark

4. Click **Save and Close**.
5. Go to **Home > Order Management > Pricing Administration >Price Lists**.
6. Search for price list Corporate Segment Price List.
7. Click **Standard Items > Coverages**, then add an item:

Attribute	Value
Item	WR11531
Description	Desktop PC Warranty
UOM	Year
Service Period	Year
Service Duration	3

Note

- To verify you added a coverage item, make sure the Associated Items column displays the Managed Covered Item icon, and that Service Duration Period and Service Duration are grey and disabled.
- This example assumes that Product Management includes the coverage item WR11531. If the search can't find the coverage item you require, then you must add it in the Product Management work area.

8. In the Associated Items column of the row you just added, click the Manage Covered Item **icon**.
9. On the Manage Covered Items page, click **Add Row**, set the values, then click **Save**:

Attribute	Value
Pricing UOM	Each
Coverage UOM	Year
Action Type	Add

10. Click **Create Charge**.
11. In the Charge details area, set values, then click **Save**:

Attribute	Value
Charge	Recurring Sale Price
Frequency	Month
Calculation Method	Covered Item Price Percent Use this method for a coverage pricing rule when the rule bases the price of the coverage on a percent of the covered item price.
Coverage Basis	Coverage_Basis_for_Desktop_PC
Calculation Amount	10
Start Date	Set a date.
End Date	Set a date.

Related Topics

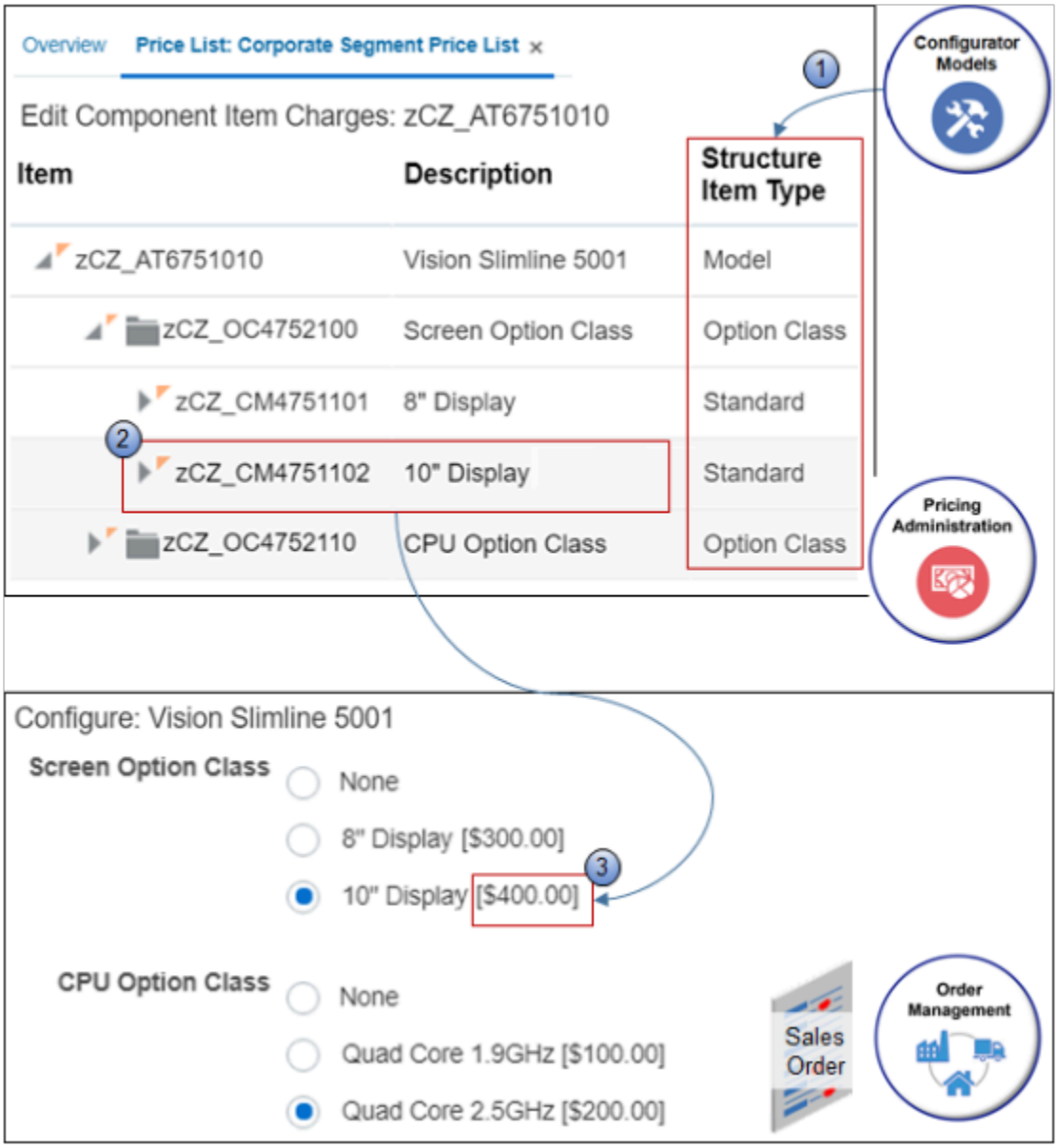
- Manage Price Lists
- Manage Pricing Bases
- Pricing for Covered Items

Configured Items

Set Up Pricing for Configuration Models

Set up pricing for a configuration model in a price list or discount list.

Here's an example:



You use the:

1. Configurator Models work area to set up the model for the zCZ_AT6751010 Vision Slimline 5001 item, including the hierarchy, option classes, and configurable options for each class.
2. Pricing Administration work area to you add the item to the Corporate Segment Price List, then use the Edit Component Item Charges page to create a charge for each option. For example, you create a $400 Base Price charge for the 10" display.
3. Order Management work area to search for the zCZ_AT6751010 on the order line, click the structure icon, and then select the options. The page displays the charge that you created for each option, such as $400 for the 10" display.

Note

- This example assumes you already used the Configurator Models work area to set up the configuration model. For details about configuration models, including how to set one up, see *Modeling Configurations for SCM*.

- This topic describes how to set up pricing for a simple configuration model. For a more complex model, see *Set Up Pricing for Your Configured Item*.
- This topic uses example values. You might need different values, depending on your business requirements.

1. Go to **Home > Order Management > Pricing Administration > Price Lists**..
2. On the Price Lists page, search for and open Corporate Segment Price List.
3. On the Price List page, in the search results, click **Actions > Add Row**.
4. In the row you just added, search for the item:

Attribute	Value
Item	zCZ_AT6751010

5. In the Associated Items column, click the **icon**.

   The icon displays only for a configured item or for a coverage item.
6. On the Edit Component Item Charges page, in the Item Structure Details area, expand the list, then notice the hierarchy displays for the parent zCZ_AT6751010 Vision Slimline 5001 and its child option classes.
7. Create the charge for the parent model.

   Click the **row** that contains Vision Slimline 5001 in the Description column, click **Create Charge**, then set the values:

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	100
Start Date	Set the date when to start using this charge.
End Date	Set an optional date when to stop using the charge. If you don't set an end date, Pricing will use this charge continuously unless you modify it at some later time.

You must create a charge for the parent model. If you don't, the Order Management work area will display an error message when you submit. The error indicates that a charge is missing.

8. Create charges for the Screen Option Class:
    - Expand zCZ_OC4752100, the Screen Option Class, then notice the configurable options that display:

Item	Description
zCZ_CM4751101	8" Display
zCZ_CM4751102	10" Display

    You create a charge only on the configurable option, not the option class.
    - Click the **row** that contains 8" Display in the Description column, click **Create Charge**, then set the values:

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	300

    - In the Item Structure Details list, in the 8" Display row, notice the Charge column displays a check mark.

    The check mark indicates you created a charge for the item.
    - Click the **row** that contains 10" Display in the Description column, click **Create Charge**, then set the values:

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Price
Base Price	400

9. Repeat step 8 for each option class. Use the same values for Pricing Charge Definition and Calculation Method that you used in step 8. For example:

Configurable Option	Base Price
Quad Core 1.9GHz	100

Configurable Option	Base Price
Quad Core 2.5GHz	200

10. Click **Save and Close**.
11. Verify your set up:
    - Sign out of Oracle Pricing, then sign into Oracle Applications with the privileges that you need to manage sales orders.
    - Go to the Order Management work area.

      Create a sales order, search for the zCZ_AT6751010 Vision Slimline 5001 item on the order line, click the structure icon, then verify that the page displays the charges that you set up for each configurable option.

Note

- Pricing doesn't create the charge from the discount list. The discount rule for a configurable option is specific to the configuration model. For example, Pricing applies the discount rule for the hard drive in the context of the configuration model. If you set up the discount rule outside of the configuration model, the Pricing applies the discount rule separately.
- You can apply pricing to the configured item's base charge or to the rolled up charge when you set up a discount on the configured item.

Rollup Charges

Each child configurable option can have its own charge. You typically add these charges together to come up with the total price for the parent configuration model. We refer to the charge on the configurable option as a *rollup charge* and the total of the rolled up charges as the *aggregate charge* because Pricing rolls up these child charges to the parent.

You can set the RollupFlag attribute to Y to tell Pricing that you want to roll up the charge. Assume the rollup charge for the 10" display is $400.00, and it's $200.00 for the Quad Core 2.5GHz CPU. The aggregate charge will be $600.00. For an example of how Pricing rolls up charges, see *Manage Pricing Algorithm Steps*.

Apply Pricing to Rollup Charges on Discount Lists

If you add a check mark to **Apply to Rolled-up Charge** when you create the discount rule for the model, then Pricing applies the discount that you set for the model on the configured item and on the configurable options. It applies the rollup discount and also adds any discounts that you add on each item.

Assume you set up discount rules for the Vision Slimline 5001 model and the configurable options. Here are the calculations that Pricing does at run time:

Description	Charge	Discount	Calculation Without Rollup	Calculation With Rollup
Charge for the Vision Slimline 5001 model	$100	10% of the $100 charge equals $10	$100 price minus $10 discount equals $90	$90
Charge for the 10" Display configurable option	$400	10% of the $400 charge equals 40	$400 charge minus $40 discount equals $360	$360 minus $40 (10% roll-up from configured item) equals $320

Description	Charge	Discount	Calculation Without Rollup	Calculation With Rollup
Charge for the Quad Core 2.5GHz option	$200	No discount	$200	$200 minus $20 (10% roll-up from configured item) equals $180
Total price	Not applicable	Not applicable	$90 plus $360 plus $200 equals $650	$90 plus $320 plus $180 equals $590

Manage Configurable Options Across Configuration Models

You can specify pricing for a configurable option independently of and across more than one configuration model. It isn't necessary to set up pricing for the configurable option in the context of each configuration model. Instead, you can maintain configurable option pricing for all configuration models on the same price list.

You can also maintain configurable option pricing:

- In a tier adjustment or pricing matrix
- For price list rules at the All Items level

Assume you set up:

- A configuration model that includes three configurable options.
- A sales charge for the configuration model at $1000.
- A sales charge for the first two configurable options at $20 each.
- A price of the third configurable option at $10, and specify it as a separate line item on the price list. Its the same price across all configuration models.

Pricing does the calculation at run time:

- Price the first two configurable options at $20 each.
- Price the third configurable option at $10.
- Price the configuration model at $1050.

Get Pricing Details for Configured Items

You can use these attributes in your priceSalesTransaction REST API payload to get charge details for the parent model and the model's children:

Attribute	Description
RootLineId	Specify the line number that contains the configuration model.
ComponentIdPath	Specify the InventoryItemNumber of the configuration model, and then the inventory item ID of the child. Assume the model's ID is PTO54222, and the child's is OP44136. Here's your code: `"ComponentPath": "PTO54222> OP44136",`

Attribute	Description

Assume you have this configuration model.

```
PTO54222 Laptop Computer
 OP44136 Keyboard Option Class
 KB18761 Standard Keyboard
 KB18759 Keyboard with Touchpad
```

where

- PTO54222 identifies the parent model.
- OP44136 identifies an option class that's in the PTO54222 model.
- KB18761 identifies a configurable option that's in the OP44136 option class.
- KB18759 identifies a configurable option that's in the OP44136 option class.

Here's an example payload that gets price details for the parent model and it's children.

```
{
 "Header": [
 {
 "AllowCurrencyOverrideFlag": false,
 "CalculatePricingChargesFlag": true,
 "CalculateShippingChargesFlag": false,
 "CalculateTaxFlag": false,
 "CustomerId": 1006,
 "HeaderId": 1,
 "SellingBusinessUnitId": 204,
 "SellingLegalEntityId": 204,
 "TransactionTypeCode": "ORA_SALES_ORDER"
 }
 ],
 "Line": [
 {
 "HeaderId": 1,
 "InventoryItemNumber": "PTO54222",
 "InventoryOrganizationId": 204,
 "LineCategoryCode": "ORDER",
 "LineId": 1001,
 "LineQuantityUOMCode": "Ea",
 "LineQuantity": {
 "Value": 1,
 "UomCode": "Ea"
 },
 "LineTypeCode": "ORA_BUY"
 },
 {
 "HeaderId": 1,
 "ComponentPath": "PTO54222> OP44136",
 "InventoryItemNumber": "KB18759",
 "InventoryOrganizationId": 204,
 "LineCategoryCode": "ORDER",
 "LineId": 1002,
 "LineQuantityUOMCode": "Ea",
 "LineQuantity": {
 "Value": 1,
 "UomCode": "Ea"
 },
 "LineTypeCode": "ORA_BUY",
 "RootLineId": 1001
```

```
},
{
"HeaderId": 1,
"ComponentPath": "PTO54222> OP44136",
"InventoryItemNumber": "KB18761",
"InventoryOrganizationId": 204,
"LineCategoryCode": "ORDER",
"LineId": 1003,
"LineQuantityUOMCode": "Ea",
"LineQuantity": {
"Value": 1,
"UomCode": "Ea"
},
"LineTypeCode": "ORA_BUY",
"RootLineId": 1001
}
],
"PricingServiceParameter": {
"PricingContext": "SALES",
"PerformValueIdConversionsFlag": "true"
}
}
```

For more, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Document Prices,**, then click **Price Sales Transaction**.

Related Topics

- Manage Price Lists
- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Strategies
- Adjust Price for Pricing Rules
- Set Up Pricing for Your Configured Item

Migrate Through Updates

Promote Pricing Algorithms Into the Latest Update

Use the Promote All action to incorporate new features, performance improvements, and corrections that Oracle Pricing provides in the latest update into pricing algorithms from earlier updates.

CAUTION: You must use the Promote All action to promote your pricing algorithms immediately after you move to the latest update. You must promote your algorithms in every update. If you don't, then each of your current and extended algorithms will remain on older versions and you risk not taking advantage of new features, performance improvements, and corrections that Oracle provides in the latest update. You also risk accumulating a lot of change across updates, and that makes it more difficult to promote your algorithms or in some cases even not possible to promote them. This can affect your ability to accurately test your flows when you finally do promote. If you have any algorithms that are at version 1 or greater, then you **must** reconcile and promote with each new update or patch. Some new features might include new or updated algorithms and will work correctly only after you reconcile and promote. If all of your algorithms are at version 0, then you don't need to reconcile and promote. If you no longer need your extended algorithms, then remove all of the ones that are version 2 or higher, and then use the Delete All Version 1 action. This way, you won't need to reconcile and promote algorithms when you update or patch. For background, see *Manage Algorithms* and *Performing Your Quarterly Update (Doc ID 2337485.1)*.

Try it:

1. Finish installing the latest update.

   Here's what Pricing does when you install the update:

   - Adds new features, performance improvements, and corrections to version 0 of each predefined algorithm.
   - Sets each new pricing algorithm that it's introducing as part of the update to version 0.
   - Uses version 0 as the baseline for each predefined pricing algorithm.

   Pricing uses these versions so you can refer back to version 0 and compare the baseline to your modified version. It's useful when you must modify or troubleshoot your set up.
2. Go to **Home > Order Management > Pricing Administration > Manage Algorithms** under Pricing Configuration.
3. Click **Actions > Promote All**.

   This action applies all the new features, performance improvements, and corrections from the latest update to all pricing algorithms.
4. Reconcile the modifications that you made during earlier updates so they work with the current update. For details, see *Manage Modifications Through Updates*.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Algorithm Steps
- Pricing Algorithms

Manage Modifications Through Updates

Reconcile the modifications that you made in earlier updates into the latest update.

Overview

The latest update might include new features, performance improvements, and corrections that affect your extended pricing algorithms. Reconcile means you test the changes that the latest update makes to your extended pricing

algorithms in a test environment before you promote them to your production environment. Reconcile helps to make sure your extended algorithms work as expected.

Here's a summary of what you need to do.

1. Determine whether you still need each of your extended algorithms. See the Remove Extended Algorithms That You Don't Need section later in this topic.
2. Install the latest update.
3. Do this in a test environment:
    - Identify your extended algorithms. See the Tips for Managing Extended Algorithms section later in this topic.
    - Reconcile your extended algorithms. See the example later in this topic.
    - Promote all pricing algorithms. See *Promote Pricing Algorithms Into the Latest Update*.
    - Test your extended algorithms as part of your transaction flows. See *Test and Troubleshoot Pricing Algorithms*.
    - Migrate your extended algorithms from the test environment to a production environment. See *Migrate Pricing Algorithms from Test to Production*.
4. Do this in a production environment:
    - Promote all pricing algorithms.
    - Test your transaction flows again.

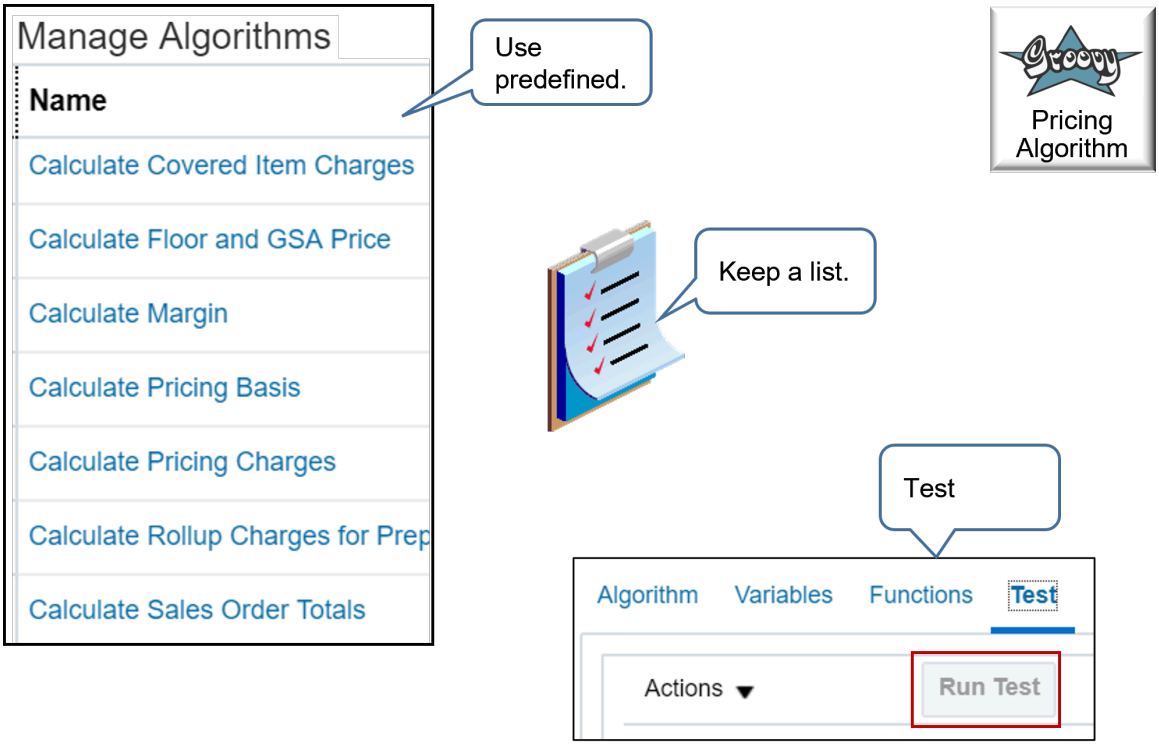
Remove Extended Algorithms That You Don't Need

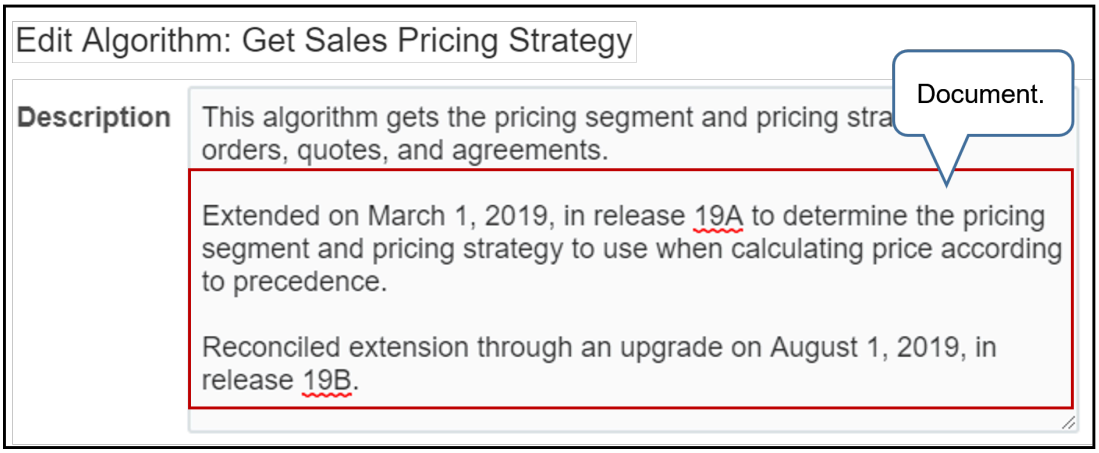
An update might affect some of the functionality that your extension implemented. For example, assume the update includes a new manual price adjustment feature. You must disable any extended algorithms that you created for manual price adjustment because manual price adjustments are now part of the update.

For another example, assume the update includes an override net price feature. If you added an extended algorithm that overrides net price in an earlier update, then you must disable it in the latest update:

- Learn about new features and corrections that Oracle Pricing introduces in each update. Evaluate whether they replace your extended algorithms. For details, see the What's New section and the Readiness Training section at *Oracle Cloud Application Update Readiness*.
- For each new feature that replaces your extended algorithm:
    - Deactivate and remove any extended algorithm that involves the feature.
    - Remove any service mappings you created that involves the feature.

Tips for Managing Extended Algorithms





Minimize management, maintenance, and troubleshooting:

- Use predefined algorithms instead of extending algorithms, when possible.
- If the predefined algorithms don't meet your needs, then extend a predefined algorithm.
- If extending a predefined algorithm doesn't meet your needs, then create a new one.
- Keep a list of the algorithms that you extend. Use it to help identify your extended algorithms when you install the latest update to make sure you don't overlook one.
- Use the Description in each extended algorithm to document your extension.
- Test each extended algorithm after you install the latest update.

Example

Assume you set up a pricing strategy in an earlier update and your strategy references different pricing segments according to pricing precedence. Assume you:

- Modified the matrix class so it includes a new result dimension that equals Precedence.
- Assigned the pricing strategy so it uses Precedence.
- Modified the pricing algorithm.

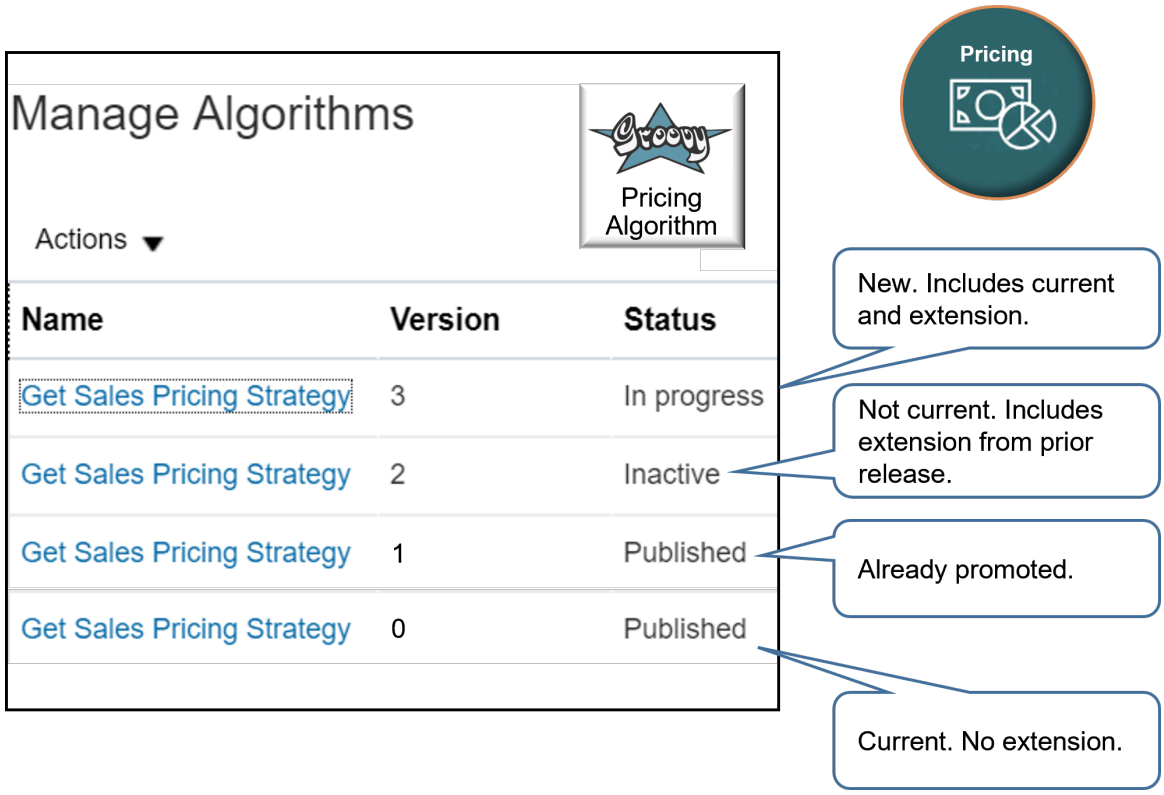
Here's what you must do.

1. Finish installing the latest update.
2. Log into your test environment.
3. Use the Edit Matrix Class page to examine the Sales Pricing Strategy Assignment matrix class that you modified. Make sure it includes the Precedence column that you added in the earlier update, and it includes the default value of 10.
4. Use the Manage Pricing Strategy Assignments page to make sure it includes the pricing strategy assignment that you added in the earlier update.

   Assume you did the setup in *Assign Pricing Strategy According to Precedence* and have already added these assignments:

Pricing Segment	Pricing Strategy	Pricing Precedence
Corporate Pricing Segment	Corporate Pricing Strategy	10
International Pricing Segment	International Pricing Strategy	20
Domestic Pricing Segment	Domestic Pricing Strategy	30

5. Go to the Manage Algorithms page and verify that it includes Version 2 of the Get Sales Pricing Strategy algorithm:



Note

Version	Description
3	The version that you will create to bring your extended algorithm from the prior update into the latest update.
2	The version that you that extended in the prior update as part of this example. The update automatically brings your extended algorithms from the prior update into the latest update, but version 2 doesn't include the changes Oracle made as part of the update, so you will deactivate it.
1	The version that you already promoted.
0	The version that Oracle includes in the latest update. It includes changes that Oracle made as part of the update, but it doesn't include your extensions. You will create a new version 3 from version 0

6. Click the row that includes version 0, and then click **Actions > Create Version**.

   Pricing creates version 3. It's identical to version 0.
7. Click the link in the row that includes version 3.

8. On the Edit Algorithm page, add a line to the description that indicates you reconciled an extended algorithm through an update:

Attribute	Value

Description	`This algorithm gets the pricing segment and pricing strategy for sales orders, quotes, and agreements.`
	
	`Extended on January 1, 2024, in update 24A to determine the pricing segment and pricing strategy to use when calculating price according to precedence.`
	
	`Reconciled extended algorithm through an update on March 1, 2024, in update 24B.`

   Make sure you include the date and update. For example:

   `Reconciled extension through an update on March 1, 2024, in update 24B.`

9. Reconcile your changes:
   - Click the **Get Header Strategy** step.
   - In the Data Sets area, click the **row** that includes Matrix in the Name attribute.
   - Set the Order By attribute to Precedence.
   - Click **Save and Close**.

   Recall that you made this change in *Assign Pricing Strategy According to Precedence*. You must reconcile each change that you made.
10. Promote your pricing algorithm.
11. Refresh the Manage Algorithms page.

    You must refresh to display the correct version. Use the filter to query the list.
12. Click the row that includes version 2, then click **Actions > Deactivate**.

    You don't need the old version, so it's helpful to deactivate it to avoid confusion across versions.
13. Click the **row** that includes version 3, then click **Actions > Publish**.
14. Test your changes:
    - Sign out of Oracle Pricing, then sign into Oracle Applications with the privileges that you need to manage sales orders.

      The browser caches the pricing algorithm. You must sign out, and then sign in to refresh the browser's cache.
    - Go to the Order Management work area, create a sales order, then make sure it assigns the pricing strategy according to precedence.
15. Troubleshoot your changes:
    - Use the View Source feature to compare the new algorithm's source code to your extended algorithm.
    - Pricing uses the algorithm's highest, published version. Make sure the highest version contains your extensions.

- Refresh the browser cache. Make sure you sign out of Order Management and sign back in each time you publish the algorithm.

16. Migrate your extended algorithm from the test environment to a production environment.
17. Log into your production environment, promote your pricing algorithm, then test your transaction flows again.

For more details about versions, see *Promote Pricing Algorithms Into the Latest Update* and *Migrate Pricing Algorithms from Test to Production*.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Algorithm Steps
- Pricing Algorithms
- Assign Pricing Strategy According to Precedence

Performance and Troubleshooting

Consider Runtime Behavior When You Modify Rules or Service Mappings

Improve application performance. Refresh pricing data to reduce the number of times required to initialize objects and to minimize round trips to the database.

Here's the data you can refresh:

Data	Description
Pricing Rule	Pricing queries rules and entities to calculate the price of an item, such as shipping charges, discounts, margin, and taxes. Here are the objects that you can refresh in your pricing rules: • Price list • Cost list • Shipping charge list • Discount list • Pricing strategy • Pricing segment • Customer pricing profile • Return price list • Pricing strategy assignment • Currency conversion list • Matrix rule Here's some more data you can refresh:

Data	Description
	• Configurations in the Setup and Maintenance Work Area • Pricing setup data • Matrix classes • Item attributes • Customer attributes Starting in update 19C, Pricing doesn't come predefined to refresh pricing rules. The Oracle Application will include changes you make to a pricing rule as soon as you make it. Example applications include Order Management, Supply Chain Financial Orchestration, Subscription Management, and so on.
Pricing Algorithm	Refresh algorithm metadata to avoid reading and parsing it more than one time from the database.
Service Mapping	Refresh your service mappings to optimize performance.

What Happens When I Modify Pricing Rules

Consider an example:

- User 1 and User 2 are each an Order Entry Specialist who use the Order Management work area.
- The Administrator is a Pricing Administrator who uses the Pricing Administration work area.

Behavior Without Refresh

Assume the Pricing Administrator set up the Standard Desktop with a unit price of $300:

Step	Description	Unit Price with Refresh Enabled	Unit Price with Refresh Disabled
1	User 1 adds the Standard Desktop to an order line.	$300	$300
2	User 2 adds the Standard Desktop to an order line.	$300	$300
3	Administrator updates the price for the Standard Desktop to $399.	Not applicable	Not applicable
4	User 1 signs out, signs in, creates a new sales order, and then adds Standard Desktop to an order line.	$300	**$399**
5	User 2 signs out, signs in, creates a new sales order, and then adds Standard Desktop to an order line.	$300	**$399**

Behavior With Refresh

Assume you set up the Standard Desktop with a unit price of $300, data refreshes every 60 minutes, and the Order Management users wait 60 minutes before signing back in:

Step	Description	Unit Price with Refresh Enabled	Unit Price with Refresh Disabled
1	User 1 adds the Standard Desktop to an order line.	$300	$300
2	User 2 adds the Standard Desktop to an order line.	$300	$300
3	Administrator updates the price for the Standard Desktop to $399.	Not applicable	Not applicable
4	User 1 signs out, **waits 60 minutes**, signs in, creates a new sales order, and adds Standard Desktop to an order line.	**$399**	$399
5	User 2 signs out, **waits 60 minutes**, signs in, creates a new sales order, and adds Standard Desktop to an order line.	**$399**	$399

Behavior With Two Charges, Without Refresh

Assume you set up the Standard Desktop with charge 1 of $300, and charge 2 of $200:

Step	Description	Unit Price with Refresh Enabled	Unit Price with Refresh Disabled
1	User 1 adds the Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300
2	User 2 adds the Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300
3	Administrator adds charge 2 of $200 to the Standard Desktop.	Not applicable	Not applicable
4	User 1 signs out, signs in, creates a new sales order, and adds Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300 charge 2 equals $200
5	User 2 signs out, signs in, creates a new sales order, and adds Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300 charge 2 equals $200

Behavior With Two Charges, With Refresh

Assume you set up the Standard Desktop with charge 1 of $300 and charge 2 of $200. Assume data refreshes every 60 minutes, and the Order Management users wait 60 minutes before signing back in:

Step	Description	Unit Price with Refresh Enabled	Unit Price with Refresh Disabled
1	User 1 adds the Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300
2	User 2 adds the Standard Desktop to an order line.	charge 1 equals $300	charge 1 equals $300
3	Administrator adds charge 2 to the Standard Desktop equal to $200	Not applicable	Not applicable
4	User 1 signs out, **waits 60 minutes**, signs in, creates a new sales order, and adds Standard Desktop to an order line.	charge 1 equals $300 **charge 2 equals $200**	charge 1 equals $300 charge 2 equals $200
5	User 2 signs out, **waits 60 minutes**, signs in, creates a new sales order, and adds Standard Desktop to an order line.	charge 1 equals $300 **charge 2 equals $200**	charge 1 equals $300 charge 2 equals $200

Behavior With Price Adjustment, Without Refresh

Assume you set up the Standard Desktop with a unit price of $300 and a $20 markup:

Step	Description	Unit Price with Refresh	Unit Price without Refresh
1	User 1 adds the Standard Desktop to an order line.	$300	$300
2	User 2 adds the Standard Desktop to an order line.	$300	$300
3	Administrator creates an adjustment matrix that adds a $20 markup on the Standard Desktop.	Not applicable	Not applicable
4	User 1 signs out, signs in, creates a new sales order, and adds Standard Desktop to an order line.	$300	**$320**
5	User 2 signs out, signs in, creates a new sales order, and adds Standard Desktop to an order line.	$300	**$320**

What Happens When Pricing Algorithms Change at Runtime

Pricing uses your runtime session to refresh each pricing algorithm:

- Pricing loads the pricing algorithm the first time it gets the algorithm when it prices a transaction, then reads the algorithm from the session during each subsequent price request.
- Pricing continues to use the session data until you sign out of the session.
- Pricing doesn't refresh the data until you sign back into the application.

Consider an example:

- User 1, User 2, and User 3 are each an Order Entry Specialist who use the Order Management work area.
- Administrator is a Pricing Administrator who uses the Pricing Administration work area.

1. The administrator creates an pricing algorithm named Calculate My Shipping Charges, creates version 1 for the algorithm, then publishes it.
2. User 1 signs into Order Management, creates a sales order, then adds the Standard Desktop to an order line.
3. Order Management sends a request to Pricing to price the order line.
4. Pricing looks for Calculate My Shipping Charges in the session data but doesn't find it because this is the first request in this session.
5. Pricing gets Calculate My Shipping Charges from the database and loads it into the session data.
6. User 2 signs into Order Management, creates a sales order, then adds the Standard Desktop to an order line.
7. Order Management sends a request to Pricing to price the order line.
8. Pricing looks for Calculate My Shipping Charges in the session data, finds it, and uses it to price the order line.

Assume the administrator modifies the Calculate My Shipping Charges pricing algorithm, creates version 2 of it, then publishes it:

1. User 1, who hasn't signed out of Order Management, creates a new sales order, then adds the Standard Desktop to an order line.
2. Order Management sends a request to Pricing to price the order line for User 1.
3. Pricing looks for Calculate My Shipping Charges in the session data, finds version 1, and uses it to price the order line for User 1.
4. User 2 signs out of and signs into Order Management, creates a sales order, and adds the Standard Desktop to an order line.
5. Order Management sends a request to Pricing to price the order line for User 2.
6. The administrator published version 2, so Pricing looks for version 2 of Calculate My Shipping Charges in the session data but doesn't find it. So, Pricing loads version 2 into the session data, then uses version 2 to price the order line for User 2.
7. User 3 signs into Order Management, creates a sales order, and adds the Standard Desktop to an order line.
8. Order Management sends a request to Pricing to price the order line for User 3.
9. Pricing looks for version 2 of Calculate My Shipping Charges in the session data, finds it, and uses it to price the order line for User 3.

What Happens When I Modify Service Mappings

Pricing publishes each service mapping to session data in Oracle Metadata Service. If you don't modify any service mappings, then Pricing reads the service mapping from the data in Oracle Metadata Service when it prices your item.

Here's what Pricing does if you modify a service mapping:

1. Gets the service mapping that you modified.
2. Creates the schema for the mapping you modified.

3. Publishes the schema to the data in Oracle Metadata Service.
4. Reads the mapping from the data in Oracle Metadata Service to price your item.

For details, go to *Oracle Middleware Understanding Oracle Application Development Framework*, then see the *Oracle Metadata Services* chapter.

Related Topics

- Manage Pricing Algorithms
- Pricing Rules
- Service Mapping

Troubleshoot Your Pricing Setups

Troubleshoot problems you encounter that involve pricing.

Various Lists

Trouble	Shoot
I encounter an error when I create a sales order in the Order Management work area. `A matching price list cannot be found for this transaction for the pricing strategy` This problem might happen when I set up more than one business unit, I set up a price list for each unit, and the error only happens for one of the units. You might not have assigned the correct access set to the price list.	Assume you created price list m for business unit n, price list y for business unit x, and the problem happens only when you create a sales order that references business unit x: 1. Go to the Pricing Administration work area. 2. Click **Price Lists**. 3. Search for, then open price list y for editing. 4. Click **Access Sets**. 5. Add the Common access set.
I add an item to a cost list and save. I add the item on the order line in Order Management, then click the price on the order line, but the charge breakdown doesn't contain any text or values.	If you add an item to the cost list but don't create a charge for it, pricing will fail. Make sure you click **Create Cost Charge**, set up the charge, then click **Save** when you add the item to the cost list on the Cost List page in Pricing Administration.

I create a sales order for customer Computer Service and Rentals in the Order Management work area, add item AS54888 on the order line, submit, then an error displays.

```
The application can't find a matching price list for this transaction for pricing strategy Corporate Pricing
 Strategy.
```

Its possible you didn't add the item to the price list. For details, see *How Profiles, Segments, and Strategies Work Together*.

Try this:

1. Examine the profile:
    - In the Pricing Administration work area, click **Tasks > Manager Customer Pricing Profiles**.
    - On the Manager Customer Pricing Profiles page, search for customer Computer Service and Rentals.
    - In the search results, notice the value:

Customer Name	Customer Value
Computer Service and Rentals	Very High

2. Examine the segment:
    - Click **Tasks > Manage Pricing Segments**.
    - On the Manage Pricing Segments page, notice the values:

Pricing Segment	Customer Value
High Value	Very High

    In this example, notice Pricing assigns the High Value segment to each customer that has a Very High value. Computer Service and Rentals has a Very High value, so Pricing places Computer Service and Rentals in the High Value segment.
3. Examine the strategy assignment:
    - Click **Tasks > Manage Pricing Strategy Assignments**.
    - On the Manage Pricing Strategy Assignments page, on the top part of the page, click the **row** that includes Header in the Assignment Level attribute, then notice the values that display in the matrix:

Pricing Strategy	Pricing Segment
Corporate Pricing Strategy	High Value

    In this example, notice Pricing assigns the Corporate Pricing Strategy to the High Value segment. In the previous step, you determined Pricing places Computer Service and Rentals in the High Value segment, so Pricing uses the Corporate Pricing Strategy for Computer Service and Rentals.
4. Examine the strategy:
    - Click **Tasks > Manage Pricing Strategies**.
    - On the Manage Pricing Strategies page, search the Name attribute for Corporate Pricing Strategy.
    - In the search results, click **Corporate Pricing Strategy**.
    - Click **Price Lists**, then, in the Segment Price Lists area, notice you assigned one price list to the strategy:

Name	Currency
USD Price List	USD

At this point, you have confirmed that the profile, segment, strategy assignment, and strategy result in using the USD Price List for customer Computer Service and Rentals. So, its very likely you didn't add the AS54888 item to the price list, and that's what's causing the error.

5. Examine the price list:
    - In the Name column, click **USD Price List**.
    - On the Price List page, in the Items area, search the Item attribute for AS54888.
    - The search likely doesn't return a result, so, in the search results, click **Actions > Add Row**, then add the AS54888.

I create a sales order in the Order Management work area, add an order line, but the currency on the order line defaults to a value I didn't expect.

Assume you used the Pricing Administration work area:

- Created a price list named RMB Price List, set the currency for the price list to RMB, then added the AS54600 item to it.
- Added the RMB Price List to the Corporate Pricing Strategy.
  For details, see *Use Different Currencies for the Same Customer*.

Next, in the Order Management work area, you add the AS54600 to an order line. The currency on the order line defaults to USD. You expect it to default to RMB.

Its possible you didn't set up the default currency or currency conversion correctly:

1. Identify your strategy:
    - Make sure you have the privileges that you need to manage sales orders.
    - Go to the Order Management work area, open your sales order, set the Customer attribute, then click **Additional Order Details > Pricing and Currency**.
    - Copy the strategy name, such as Corporate Pricing Strategy, to your clipboard.
    - Sign out of Order Management.
2. Examine your set up:
    - Make sure you have the privileges that you need to administer pricing.
    - Go to the Pricing Administration work area.
    - In the Pricing Administration work area, click **Tasks > Manage Pricing Strategies**.
    - On the Manage Pricing Strategies page, search for, then open Corporate Pricing Strategy.
    - On the Edit Pricing Strategy page, notice the value for the Default Currency attribute.
      In this example, the order line defaults to USD, so its likely the Default Currency attribute contains USD. This set up is ok if you plan to use this strategy primarily for order lines that require USD.

- Click **Currency Conversion Lists**, then examine the list.

3. Fix your set up:
   - Create a currency conversion list that converts USD to RMB. For details, see *Manage Currency Conversion Lists*.
   - Add the currency conversion list to your pricing strategy.

Pricing Rules

- Make sure you add the list that contains the pricing rule to a pricing strategy.
- Make sure you approve the list. Click Approve on the page that you use to create the list so you can add it to a pricing strategy and so the pricing algorithm can use it. Pricing displays Approve only if you sign in with pricing privileges, and only if you haven't already approved the list.
- Make sure the transaction happens during the time frame that you specify when you set the start date and end date for the list.

Charge Breakdown Doesn't Display a Value

I set up a discount rule, create a new sales order and add an item to it, but one of the lines in the charge breakdown doesn't display a value until I save or submit. I don't want to save or submit the order yet because I am providing a quote to my customer.

Try this:

1. Go to Pricing Administration.
2. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
3. Click **Tasks > Manage Service Mappings** under Pricing Configuration.
4. Search for and open the service mapping you use with your rule, such as the Sales mapping.
5. Click **Sources**.
6. Map your attribute.

   Assume you use the RoundingBillToPartyId attribute in your discount rule:
   - In the Sources list, click **OrderCatalogLine**.
   - In the Entity Mapping list, click **Header**.
   - In the Attribute Mappings list, click **Actions > Add**, then map the RoundingBillToPartyId attribute to the PartyId view object attribute.
7. Repeat step 5, but map RoundingBillToPartyId to PartyId in the OrderLine source.

Note

- Examine the predefined mapping of RoundingBillToPartyId to PartyId in the Header entity of the OrderHeader source. The OrderHeader source provides pricing when you reprice or save the sales order.
- The OrderCatalogLine source provides pricing on the order line when you search.
- The OrderLine source provides pricing when you add an item to the sales order.
- OrderCatalogLine and OrderLine don't come predefined with the mapping. You must add whatever attributes you use with your rule to OrderCatalogLine and OrderLine.

Service Mappings

Trouble	Shoot
In the Pricing Administration work area, I click **Tasks > Manage Service Mappings** under Pricing Configuration, open my service mapping for editing, then click Sources. But I can't map an attribute because the Add Row action on the Attribute Mappings tab is grey and not active.	Here are some solutions you can try: • Make sure you have the privileges that you need to administer pricing. If you don't have these privileges, there are many actions that you can't do. • If you opt into the Unified Sandbox during set up, then you must create and active it. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
I set up a matrix class or algorithm variable and notice that the list of values for the Service doesn't contain any values, or doesn't contain the entity I set up in the service mapping.	Remove any values you entered in the Alias of an entity.
I encounter an error that's similar to this one. **`The definition of inherited attribute LineItemRating_Custom of entity Line in the service mapping isn't valid.`** In this example, the data type for LineItemRating_Custom is probably different across service mappings. For example, you might have set it up as a String in the Sales service mapping and as a Decimal in the MaterialTransfer service mapping.	Make sure you use the same data type for your custom attribute in the Sales service mapping and in the MaterialTransfer service mapping.
The list of values for the Join Entity is empty.	Refresh the row. Click the next row in the list of entities, then click the row where the Join Entity is empty. If the problem persists, make sure you correctly set up your attribute. For example, make sure it doesn't contain any typographical errors.
The Save action is disabled.	Do: • Cancel your changes. • Save changes frequently, such as after you set up each attribute. • Redo your set up on the entity or attribute.
Pricing doesn't save my changes.	Do: • Use the View Source XML action to verify your changes and make sure they're what you expect. • Create a new Test for your pricing algorithm. Verify that the example PriceRequest payload includes the changes you expect. • If you modify a source mapping, then create or reprice a sales order and make sure it doesn't contain any errors.
I move my service mappings from my test environment to my production environment, but production doesn't include my set up.	In your test environment, on the Manage Service Mappings page, click **Actions > View Source XML**, then verify the XML includes your changes and that the changes are what you expect. In particular, examine your _Custom objects, make sure they exist, and you set them up correctly.

Trouble	Shoot
	Do it again in your production environment, and make sure the set up is the same in test and production.
I encounter an incident error when creating or pricing a sales order.	Do: • Examine recent changes you made to your service mappings. Make sure they're correct. • If necessary, remove the source map, set it up again, and retry the flow. • Make sure View Object Name and View Object Attribute in your source mapping don't contain typographical errors. • Use Auto Suggest when you set View Object Name to make sure you set a value that doesn't contain typographical errors.
The attributes I set up don't match their intended use.	Examine the descriptions for the attributes and make sure you're using them correctly. Go to *SOAP Web Services for Oracle SCM Cloud*, expand **Price Request Service**, then click **Price Sales Transaction**.

Pricing Algorithms

You might encounter one of these errors when you test your pricing algorithm:

Trouble	Shoot
I encounter an error message. `An application error occurred. See the incident log for details.` I encounter a java.math.BigDecimal exception. This error typically happens when Pricing can't process the price override.	Add a check mark to the Allow Manual Adjustment option in the Price List Charge area of the price list. If caching is: • **Turned off**. Restart your test. • **Turned on**. Wait 30 minutes, and then restart your test.
I encounter an error message. `Unable to parse the variable[PriceRequest] using the service definition [Sales.PriceRequestInternal]. Please check the variable value or service schema.` This error typically indicates that you didn't format the XML syntax in the test case input correctly.	Use an XML editor to make sure you format the XML correctly.
I encounter an error message. `Failed to get data for Service Parameter Mapping "Sales" Source "Orderheader" Service "PriceSalesTransaction". Caused`	Restart each OrderOrchestration Weblogic Server (WLS) that you use in your test.

Trouble	Shoot
`by failing to process entity "LinePrcOverrideEFF_Custom".` This error happens because the Weblogic Server doesn't include the service data object, the entities, or the attributes that you added to the PriceSalesTransactions service.	
I encounter an error message. `Another user changed the row with primary key oracle.jbo.Key[300100078134872]` This error might happen when you modify a pricing algorithm, save your changes, and then immediately publish the pricing algorithm.	Try one of these solutions, then publish again: • On the Manage Algorithms page, enter `%Custom%` in the Query by Example to refresh the list of algorithms. • Sign out of Pricing, and then sign back in.
I encounter one of these errors when I click Run Test: • `java.lang.NullPointerExceptic` • `Attempt to access dead entity in TestcaseIOEO, key=oracle.jbo.Key[3001000781` • `Attribute VariableName in SetTransformAlgorithmAM.Testc is required.` These errors happen when an unsaved modification exists in the input payload and you click Run Test.	Click **OK** in the error dialog, cancel your algorithm edit, navigate to the Manage Algorithm page, open the algorithm, make your changes, save them, then click **Run Test**.
A modification that I make to a pricing rule doesn't seem to go immediately into effect. There's a delay. I encounter an application error when I import a sales order.	Make sure the EnableCache variable equals False on your pricing algorithm. EnableCache comes predefined as False. Don't set it to True.

Import

You might encounter one of these errors when you import pricing:

Trouble	Shoot
I encounter an error when I run the *Import Price Lists Batch* scheduled process. `java.sql.SQLException: ORA-06533: Subscript beyond count ORA-06512: at "FUSION.QP_IMPORT_UTIL", line 1681`	This problem might happen because there's no value set for the Default Price Periodicity UOM Class pricing parameter. To fix it, set a value for the parameter, then redo your import. For details, see *Manage Pricing Parameters*.

Trouble	Shoot

`ORA-06512: at "FUSION.QP_PL_CHARGES_IMPORT", line 1307`
`ORA-06512: at "FUSION.QP_PL_ITEMS_IMPORT", line 1453`
`ORA-06512: at line 1`	

I import a sales order into Order Management but the Your Price attribute on the order line is 0. This might happen if you import the sales order with the FreezePriceFlag attribute set to True. If FreezePriceFlag is True, then Pricing doesn't calculate price for the order line. Instead, it copies attribute values from your import payload, such as from the Extended Amount attribute on the DOO_ORDER_LINES_ALL_INT worksheet when you use file-based data import.	Make sure your import payload has values in the attributes that Pricing needs to set price when you freeze pricing. For details, see *Freeze Price on Sales Orders* and *Import Price Lists*.
I create an order revision, enable the Freeze Pricing and Shipping Charges option on it, then submit. At some later time, I copy the order, add an order line to the copy, but then encounter an error when I submit. `The order was not priced because the charge for the item does not contain a value. Include a value, and then reprice the order.`	You can't use the Order Management work area to add the line because you froze pricing when you revised the original order, but you can import a line through order import. If you decide to import, make sure your import payload has the pricing details that Pricing needs to price the line. You can freeze price when you copy a sales order. For details, see *Copy Sales Orders*.

Runtime

Troubleshoot problems that happen in your runtime environment.

Problems with Pricing Set Ups That Have Expired

Trouble	Shoot
I create and submit a sales order in the Order Management work area on March 1, 2021. The next day, on March 2, 2021, I revise the sales order and submit it, but encounter an error. `Order Revision Fails With: An Application Error Occurred. See the Incident Log for More Information.`	Its possible that part of your pricing set up has expired. 1. Open the sales order in the Order Management work area, click **Additional Order Details > Pricing and Currency**, then examine the values. Assume Pricing Segment equals Corporate Segment. 2. Make sure you have the privileges that you need to administer pricing, go to the Pricing Administration work area, then make sure none of the objects that are involved in pricing the sales order have expired. For example, make sure the End Date for the Corporate Segment happens after March 2, 2021. Here are some of the objects that have end dates that you should examine: • Price List

Trouble	Shoot
	• Return Price List • Cost List • Discount List • Shipping Charge List • Pricing Rule • Pricing Guideline • Pricing Profile • Pricing Segment • Pricing Strategy • Pricing Strategy Assignment • Currency Conversion List • Pricing Algorithm If the end date has expired, then set it to a later date. As an alternative, redo your pricing set up so none of the objects have expired. For example, if the price list expired but you don't to use it any more, then create a new price list and add the item to the new price list.
In January, I create a sales order in the Order Management work area, add an order line, set the UOM on the line to Box, then submit the order. In April, I revise the order but get an error when I submit it. `An application error occurred. See the incident log for more information.` This problem might happen because the price list that Pricing used to price the item in January is no longer active. Pricing used a new price list to price it April, but the new price list doesn't have a charge for the Box UOM.	1. Open the sales order in the Order Management work area. 2. On the order line, in the Amount column, click the value, then look for the name of the price list, such as Base List Price Applied from Corporate Segment Price List. 3. Go to the Pricing Administration work area, open the price list that you noticed on the order line, then add a charge for the Box UOM to the item on the price list. This allows the pricing algorithm to determine the price when you reprice the order, save it, or submit it. As an alternative, activate the old price list. For details, see *Manage Price Lists*.
I add the AS54888 item to the Corporate Segment Price List and set the Pricing UOM attribute to Pallet when I add the item to the list. But at runtime, the UOM on the order line defaults to Each.	Make sure the End Date attribute on Corporate Segment Price List is empty or contains a date that happens after the current system date.

Problems with Pricing Strategies and Pricing Segments

Trouble	Shoot
I create a new sales order in Order Management and encounter a problem on the New Sales Order page.	If the problem persists, make sure you create a profile, segment, strategy, and strategy assignment for the customer that you select in the Customer attribute on the New Sales Order page in Order Management.

Trouble	Shoot
The order header doesn't contain a value for the pricing segment and the strategy defaults to Corporate Pricing Strategy. I click the error icon and notice an error. **`The pricing strategy was not determined for the current transaction.`** **`No matching pricing segment was found for the customer.`**	
I create a sales order in the Order Management work area. On the order header, I click **Additional Order Details > Pricing and Currency**, but the Order Currency attribute is empty. Or, I encounter one of these errors. **`Pricing strategy was not determined for the transaction`** **`No matching pricing segment found for the transaction`** **`No matching pricing segment found for the customer`** Its possible that you haven't set up a default pricing segment.	Set up a default pricing segment. For details, see *Manage Pricing Segments*.
I create a sales order in the Order Management work area and leave the Customer and Business Unit attributes empty. On the order header, I click **Additional Order Details > Pricing and Currency**, but the Order Currency attribute is empty. If I set the Customer and Business Unit first, click **Additional Order Details > Pricing and Currency**, then the Order Currency attribute displays a value. I expect the New Sales Order page to display my currency before I set any values on the order header.	Set the value of the Default Currency attribute on your pricing strategy. For details, see *Manage Pricing Strategies*.
I encounter an error: **`An error occurred while deriving a pricing strategy for the transaction`** This problem might happen because an attribute that has a unique value was changed. For example, while migrating between environments or upgrading to a new update, you change the value in the Source Code Name attribute on	Attribute values are case sensitive. Make sure your attribute values match each other exactly, including upper case and lower case.

Trouble	Shoot
the Sales Pricing Strategy Assignment matrix class from PricingStrategyId to PricingStrategyID.	

Other Problems

Trouble	Shoot
I click Enable Returns on a sales order, add an item on the Return Lines tab, add a line, don't enter the original order, then click **Apply > Save > Add**, but then the return line doesn't have a price.	Try this: • Go to the Setup and Maintenance work area and enable the Return Items or Cancel Services Without a Reference Order opt-in feature. • Go to the Pricing Administration work area and use the Promote All action to promote your pricing algorithms. • Examine your pricing rules and pricing algorithms. Make sure you have set them up correctly. • If you're importing a sales order, then make sure your import payload has values for the attributes that pricing needs to price a return line.
I add an item to a price list, but the order line I create in Order Management doesn't include the changes I made in the Pricing Administration work area. I am collecting planning data and using the *Refresh and Start the Order Promising Server* scheduled process to publish my changes.	Wait for the scheduled process to finish and for your changes to propagate through your environment. Wait about one hour.
The charge breakdown on the order line in Order Management doesn't contain the rows I need, such as Margin or Cost of Goods Sold.	Set up the pricing results presentation. For details, see *Manage Order Line's Price Details*.

Other Resources for Troubleshooting

See sections in *Administering Pricing* for other troubleshooting tips.

Related Topics

- Troubleshoot Pricing Matrixes
- Pricing Rules
- Test and Troubleshoot Pricing Algorithms
- Create a Sandbox So You Can Edit Service Mappings
- Create and Activate Sandboxes

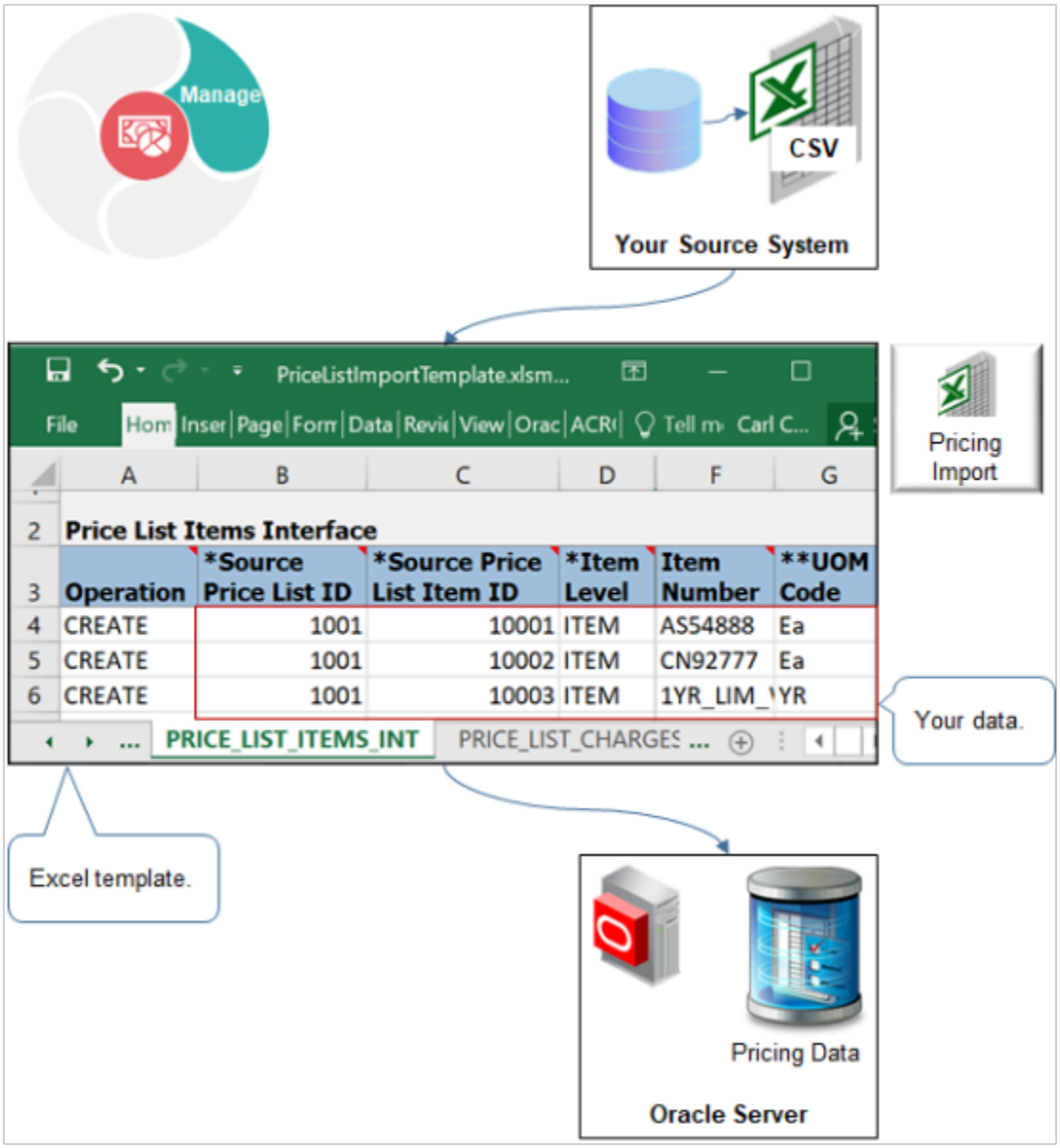
7 Import and Export

Import Price Lists

Import Price Lists

Use the Price List Import Template to import price lists from your source system into Oracle Pricing.

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. It includes columns on each tab, each of these columns represents a column in the database table, and it specifies the data type that Oracle requires for each database column.

You:

- Must use the Excel template that Oracle provides.
- Convert your price list into a CSV (comma separate values) file that uses the same structure that the template uses.
- Use a scheduled process to upload data from the template into Pricing Administration.

Summary of the Steps

1. Prepare price list.
2. Create the import file.
3. Upload price list.
4. Import price list.

Note

- Use this procedure to create a price list that includes child objects. You can also use the Pricing Administration work area to create a price list, but you must manually add each child object.
- The tools, guidelines, and procedures that you use are similar to importing sales orders. For details, see *Overview of Importing Orders Into Order Management*.
- For a demonstration, see *Oracle Pricing Cloud Enhancements, Import or Update a Large Number of Price Lists*. The demonstration starts at 08:11.

Oracle introduced PriceListsImportBatchTemplate.xlsm to replace PriceListImportTemplate.xlsm. We recommend that you use PriceListsImportBatchTemplate.xlsm instead of PriceListImportTemplate.xlsm:

- You can use the Update operation with PriceListsImportBatchTemplate.xlsm but not with PriceListImportTemplate.xlsm.
- PriceListsImportBatchTemplate.xlsm is faster, even if you are importing a single price list.
- Oracle might no longer support PriceListImportTemplate.xlsm in a future update.

For details, see *Import Batches of Price Lists*.

Prepare Price List

Convert your price list into a CSV (comma separate values) file that uses the same structure that the Price List Import Template uses.

Using the template helps to make sure that your converted price list uses the same structure that the Oracle database requires. This topic describes one way to convert your price list. The details of your conversion might require a different way. If you can't use the Price List Import Template for some reason, then consult Oracle resources about how to use an open interface when importing data.

Convert your price list into a CSV file.

1. Download the Price List Import template:
    - Go to *File-Based Data Import for Oracle Supply Chain Management Cloud*.
    - Expand **Order Management**, then click **Price List**.
    - In the Price List area, click **PriceListImportTemplate.xlsm**, then notice that the file automatically downloads to your local computer.

2. Use a spreadsheet editor that can read a CSV file, such as Excel, to open the Price List Import Template, then familiarize yourself with the structure it uses.
3. Use a data manipulation tool to structure your price list so it mirrors the structure that the Price List Import Template contains, then save your data to a CSV file.

   Use SQL (Structured Query Language), ODI (Oracle Data Integrator), or some other tool to convert your price list into a CSV file.

Guidelines for Converting Price List Into a CSV File

Create a CSV file that includes data for interface tables and columns when you do the conversion.

Here are some guidelines that you can use to make sure the structure that the CSV file contains mirrors the structure that the Price List Import Template contains:

- Include the same table names and column names in your CSV file that the templates uses.
- Sequence the tables in your CSV file in the same sequential order that the templates uses. The tab sequence that the template uses determines the sequential order.
- Include the same columns in each table in your CSV file that the templates uses, and arrange these columns in the same sequential order inside each table.
- Use the same data type for each column.
- Include data for all required columns. The Price List Import Template uses an asterisk (*) to indicate required columns. For example, the Calculation Method column on the PRICE_LIST_CHARGES_INT tab is required.

  Import might require some other columns depending on the conditions that apply to your usage.

  The Price List Import Template uses double asterisks (**) to indicate a set of required columns. You must include a value for at least one column in each set. For example, the template uses double asterisks with UOM Code and UOM Name to indicate that they constitute one set. You must include a value for at least one of these columns.

View an example of the CSV structure.

1. Open the Price List Import Template in Microsoft Excel.
2. Click the **PRICE_LISTS_INT** tab.
3. Select **File > Save As**, then click **Excel Workbook**.
4. Save the file as a .csv file, then use a text editor to examine the output.

Create the Import File

1. Prepare the Price List Import Template:
   - Use a spreadsheet editor that can read a CSV file, such as Excel, to open the Price List Import Template.
   - Delete the example data from the Price List Import Template.

     This template comes predefined with example data that helps you to determine the type of data you must include. For example, row five of the PRICE_LISTS_INT tab includes example data. Make sure you delete all example data from all tables in the spreadsheet, even from tables that you don't need.
2. Copy and paste your price list into each of the tables in the Price List Import Template:
   - Use a spreadsheet editor to open the CSV file that contains your price list.
   - Copy your line data to the clipboard.

- In the Price List Import Template, click the **PRICE_LISTS_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 Name column contains VARCHAR data, and that the Source Price List ID column contains NUMBER data.
- Continue copying data for each table until you finish copying all your price lists into the Price List Import Template.
- Don't modify the tab sequence in the template or the column sequence on each worksheet. Modifying the template in this way will cause errors.
- Don't delete any tabs.

Save your work after each copy.

3. Create the import file:
   - Click the **Create CSV** tab, then click **Generate CSV File**.

     If the Generate CSV File button 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 you can use 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.

Upload Price List

1. Sign into Oracle Applications. Make sure you have the Import Price Lists privilege or the Import Approved Price Lists privilege.
2. Go to the Scheduled Processes work area.
3. On the Scheduled Process page, click **Schedule New Process**.
4. In the Schedule New Process dialog, set the value, then click **OK**:

Attribute	Value
Name	Load Interface File for Import See *Load Interface File for Import Process*.

5. In the Process Details dialog, set the parameters:

Parameter	Description
Import Process	Select Import Price Lists Batch.
Data File	Do these steps.

Parameter	Description
	a. Select the Data File attribute. b. Scroll down, then 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 Price List 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 before you continue.

The Import Process parameter and the Data File parameter are required. For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

6. Submit.
7. In the Confirmation dialog, note the value of the Process ID, click **OK**, then click **Close**.
8. Click **Actions > Refresh**.
9. Use the Process ID that you noted earlier to locate your scheduled process, then make sure the Status attribute for the process displays Succeeded.

   The Succeeded status indicates that the scheduled process successfully uploaded your price list:

   - If the upload fails on any row, then the status displays Error.
   - If the search results don't display your process, then click **Refresh** until it does.

10. 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 the data that caused the error.
    - Use Excel to open the Price List Import Template that contains your price list, then correct the price list that's causing the error.
    - In the Price List Import Template, click **Generate CSV File**, then run the scheduled process again.

    Repeat this step until the scheduled process successfully uploads your price list.

Import Your Price List

1. Run the *Import Price Lists Batch* scheduled process. In the Process Details dialog, set the parameters:

Parameter	Description
Price List Name	Enter the name of the price list.
Import Request ID	Enter the identification number that you noted in step 8 of the Upload Your Price List section of this topic.
Commit Point	Accept the default value of 100, or enter an integer that sets the maximum number of records to import at one time. For example, if Commit Point is 100, then the import will import 100 records at a time.

Parameter	Description
Number of Child Processes	Enter an integer that sets the maximum number of child processes to run during the import. If the number of records to import is: - **Less than one hundred.** The import runs two child processes, by default. - **More than one hundred.** The import runs four child processes, by default.

The Price List Name parameter and the Import Request ID parameter are required.

2. Make sure the Status for the Import Price Lists Batch scheduled process that you submitted is Succeeded.

   The Succeeded status indicates that the scheduled process successfully imported your price list. If the import fails on any row, then the status displays Error.
3. Verify your import:
   - Sign into Oracle Pricing with administrative privileges.
   - Go to **Home > Order Management > Pricing Administration > Price Lists**..
   - Search for the price list that you imported.
   - Verify that the Price Lists page displays the price lists you imported.

Interface Tables That the Price List Import Template Uses

The Price List Import Template includes a tab for each of the interface tables:

Table Name	The Table Must Contain Data About The
QP_PRICE_LISTS_INT	Price list header
QP_PRICE_LIST_SETS_INT	Access set
QP_PRICE_LIST_ITEMS_INT	Item line
QP_PRICE_LIST_CHARGES_INT	Pricing charges defined for the item.
QP_TIER_HEADERS_INT	Tier header defined for the charge.
QP_TIER_LINES_INT	Tier lines defined for the tier header.
QP_PL_COMP_ITEMS_INT	Component item line.
QP_MATRIX_DIMENSIONS_INT	Attribute matrix defined for the charge.
QP_MATRIX_RULES_INT	Attribute matrix rules defined for the matrix.

Table Name	The Table Must Contain Data About The
QP_PL_COVERED_ITEMS_INT	Covered item.

Troubleshoot

See:

- *Troubleshoot Importing Price Lists*
- The Import section in *Troubleshoot Your Pricing Setups*

Related Topics

- Manage Price Lists
- Guidelines for Importing Batches of Discount Lists

Import Batches of Price Lists

Use an Excel spreadsheet to import a batch of price lists into Oracle Pricing.

You can use FBDI to import pricing details on the Microsoft Windows or on the Apple macOS operating systems.

The tools and guidelines that you use and the procedure you do are very similar to importing a batch of discount lists, but with a few important differences:

Difference	Discount List	Price List
Name of the FBDI Template	DiscountListImportTemplate.xlsm	PriceListsImportBatchTemplate.xlsm
Scheduled processes that you run	• Set the Import Process parameter to Import Price Lists Batch when you run the Load Interface File for Import scheduled process. See *Load Interface File for Import Process*. • *Import Discount Lists*. • Import Discount List Headers and Access Sets. • Import Discount List Items and Rules.	• Set the Import Process parameter to Import Price Lists Batch when you run the Load Interface File for Import scheduled process • *Import Price Lists Batch* • Import Price List Headers and Access Sets • Import Price List Items and Charges
Interface tables	QP_DISCOUNT_LISTS_INT QP_DISCOUNT_LIST_SETS_INT QP_DISCOUNT_LIST_ITEMS_INT QP_PRICING_TERMS_INT QP_MATRIX_DIMENSIONS_INT QP_MATRIX_RULES_INT	QP_PRICE_LISTS_INT QP_PRICE_LIST_SETS_INT QP_PRICE_LIST_ITEMS_INT QP_PRICING_TERMS_INT QP_MATRIX_DIMENSIONS_INT QP_MATRIX_RULES_INT

Difference	Discount List	Price List
		QP_TIER_HEADERS_INT QP_TIER_LINES_INT
Operations that you can use	Create, Update, or No-Op.	Create, Update, No-Op, or End Insert.
Attribute that identifies the list	SOURCE_DISCOUNT_LIST_ID	SOURCE_PRICE_LIST_ID
Attribute that identifies the item	SOURCE_DISCOUNT_LIST_ITEM_ID	SOURCE_PRICE_LIST_ITEM_ID
Page in Pricing Administration that you use to verify your import	Edit Discount List	Price List

For details, see *Guidelines for Importing Batches of Discount Lists*.

Create a Charge

Use the Source Charge Id to identify your price list charge. The import maps the Source Charge Id attribute in the template to the External System Reference Id attribute in the Price List Charges table in the database.

Here's an example that creates two charges:

Row	OPERATION_CODE	SOURCE_PRICE_LIST_ID	SOURCE_CHARGE_ID	PARENT_ENTITY_TYPE_CODE	PARENT_SOURCE_ID	CHARGE_DEFINITION_CODE	BASE_PRICE
1	CREATE	1001	6001	PRICE_LIST_ITEM	10001	QP_SALE_PRICE	100.00
2	CREATE	1001	6002	PRICE_LIST_ITEM	10001	QP_SALE_PRICE	120.00

Note

- Row 1 creates a charge for Id 6001 with a base price of 100, and maps 6001 to the External System Reference Id attribute.
- Row 2 creates a charge for Id 6002 with a base price of 120, and maps 6002 to the External System Reference Id attribute.

You can also update charges. For example:

Row	OPERATION_CODE	SOURCE_PRICE_LIST_ID	SOURCE_CHARGE_ID	PARENT_ENTITY_TYPE_CODE	PARENT_SOURCE_ID	CHARGE_DEFINITION_CODE	BASE_PRICE
1	UPDATE	1001	6001	PRICE_LIST_ITEM	10001	-	110.00

Row	OPERATION_CODE	SOURCE_PRICE_LIST_ID	SOURCE_CHARGE_ID	PARENT_ENTITY_TYPE_CODE	PARENT_SOURCE_ID	CHARGE_DEFINITION_CODE	BASE_PRICE
2	UPDATE	1001	6002	PRICE_LIST_ITEM	10001	-	132.00
3	UPDATE	1001	6003	PRICE_LIST_ITEM	10003	1	249.95

Note

- Rows 1 and 2 update charges from the first example.
- Row 1 updates charge Id 6001 to a base price of 110.
- Row 2 updates charge Id 6002 to a base price of 132.
- Use the same value in SOURCE_CHARGE_ID to update the charge.
- Row 3 updates charge 6003 to a base price of 249.95. If you don't use the template to create the charge, then you can use the CHARGE_LINE_NUMBER attribute to update it instead of SOURCE_CHARGE_ID.

  For example, if you use the Pricing Administration work area, REST API, Application Development Framework Desktop Integration (ADFDI), or an older version of the FBDI template, then you can use CHARGE_LINE_NUMBER.

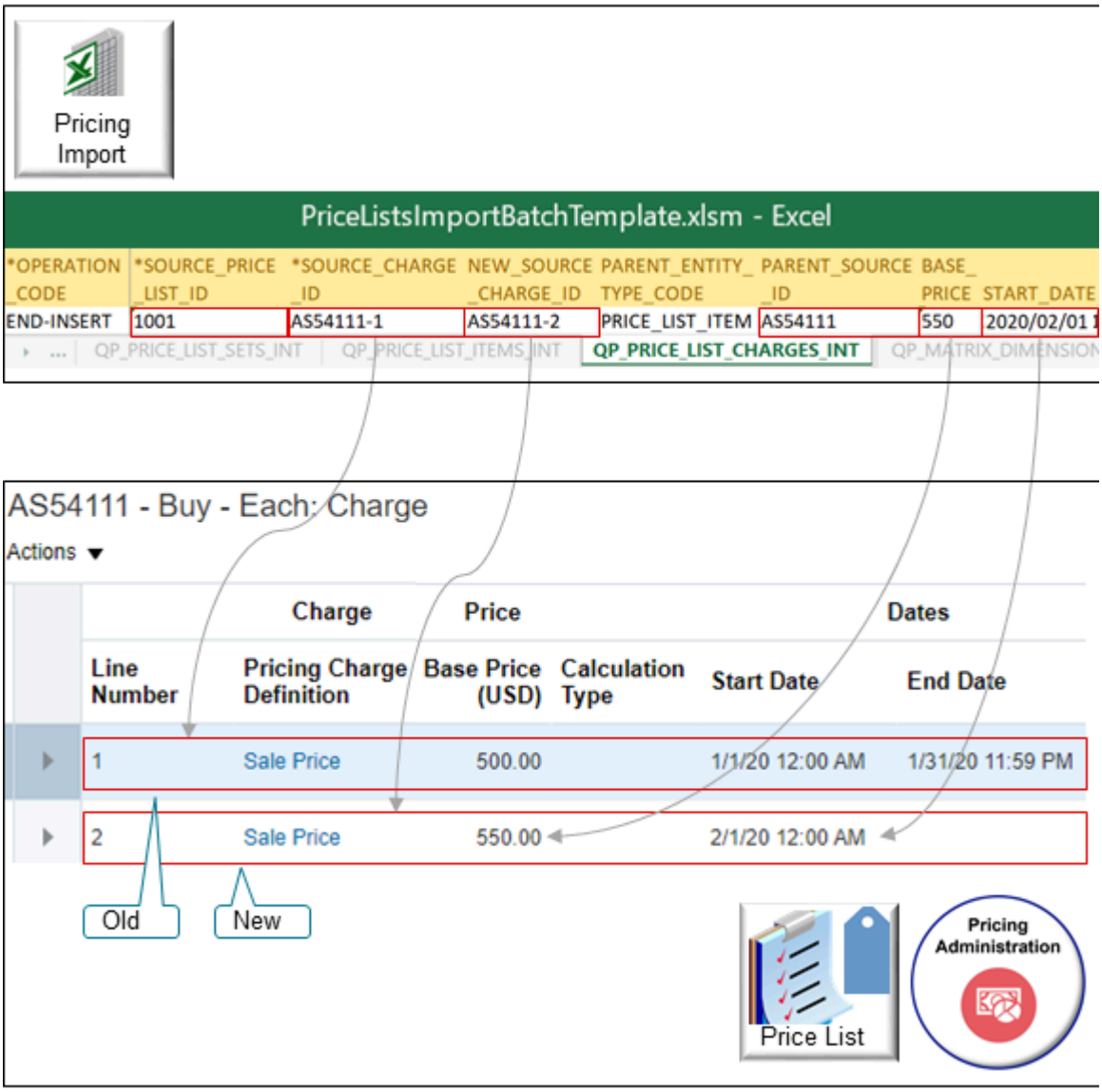
  If you use CHARGE_LINE_NUMBER, then you must also update Source Charge Id, then use Source Charge Id in all future update operations. Assume you used REST API to create charge 6003, so you set CHARGE_LINE_NUMBER to 1.

For more, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Document Prices**, then click **Price Sales Transaction**.

End a Charge and Create a New One

Use the End Insert operation to end date a charge that already exists, then create a new one.

Assume you need to change the base price of the sale price for the AS54111 item:



Try It

1. Set the values for these attributes on the QP_PRICE_LIST_CHARGES_INT worksheet:

Attribute	Description
SOURCE_PRICE_LIST_ID	Identify the price list that contains the current charge.
SOURCE_CHARGE_ID	Identify the current charge that you're ending. Use the `item-line` format. where

Attribute	Description
	○ **item** is the name of the item, such as A54111. ○ - (dash) separates the name of the item from the line. ○ **line** is the line number of the charge that you're ending, such as 1. For example, **AS54111-1** specifies to end the charge that's on line 1 of the charges for the AS54111 item.
NEW_SOURCE_CHARGE_ID	Identify the new charge that you're creating. Use the same format that you use for SOURCE_CHARGE_ID, except increment the line by one. For example, **AS54111-2** specifies to create line 2 in the charges for the AS54111 item.
PARENT_ENTITY_TYPE_CODE	Set it to the type of entity where the charge resides. The charge in this example is for an item on a price list, so set it to PRICE_LIST_ITEM.
PARENT_SOURCE_ID	Identify the item that contains the charges that you're modifying. For example, AS54111 identifies the AS54111 item.
BASE_PRICE	Specify the base price to use for the new charge you're creating, such as 550.
START_DATE	Set the date to start the new charge. End Insert uses the value in START_DATE to set the end date for the old charge. If START_DATE happens before the end date of the old charge, then End Insert sets the end date of the old charge to START_DATE minus one second. End Insert does this to avoid a date overlap.
END_DATE	As an option, set the date to end the new charge. If you don't include a value in END_DATE, then the import will set the end date of the charge that you identify in SOURCE_CHARGE_ID to START_DATE minus 1 second.

Attribute	Description

For example:

OPERATION_CODE	SOURCE_PRICE_LIST_ID	SOURCE_CHARGE_ID	NEW_SOURCE_CHARGE_ID	PARENT_ENTITY_TYPE_CODE	PARENT_SOURCE_ID	BASE_PRICE	START_DATE
END-INSERT	1001	AS54111-1	AS54111-2	PRICE_LIST_ITEM	AS54111	550	2020/02/01 00:00:00

2. Finish the import.

   End Insert will use the values that you import to end date the old charge and create a new one.
3. Go to the Pricing Administration work area, open the price list that contains the AS54111 item, then open the charge for the AS54111.
4. Verify that your import end dated the old charge, added the new charge, and set values for the new charge to the values that you specified:

Charge	Value	Operation	Start Date	End Date
Sale Price	500.00	-	1/1/20 12:00 AM	1/31/20 11:59 PM
Sale Price	550.00	-	2/1/20 12:00 AM	-

If you use the:

- **Create operation**. The value in SOURCE_CHARGE_ID in your worksheet will replace the value in the External System Ref ID attribute in the Oracle database.
- **Update operation or the End Insert operation**. The import uses SOURCE_CHARGE_ID to locate the charge to update or end date.

If you don't include a value in the END_DATE attribute, then the import will set the end date of the charge that you identify in SOURCE_CHARGE_ID to START_DATE minus 1 second.

Guidelines for End Insert

- You can use End Insert only on the QP_PRICE_LIST_CHARGES_INT worksheet.
- Use End Insert to end an existing charge and create a new one on a single row in the QP_PRICE_LIST_CHARGES_INT worksheet. This way, you don't have to end the charge on one row and then create a new charge on another row.
- End Insert will create a new charge, and the new charge will contain the values that you enter in NEW_SOURCE_CHARGE_ID, BASE_PRICE, and START_DATE.
- If you use End Insert for a charge, then you don't need to set the OPERATION_CODE attribute to NO-OP on the QP_PRICE_LISTS_INT worksheet or on the QP_PRICE_LIST_ITEMS_INT worksheet for that charge.

You must include values for these attributes when you use End Insert:

- SOURCE_CHARGE_ID
- NEW_SOURCE_CHARGE_ID
- BASE_PRICE
- START_DATE

End Insert will get the values for other attributes from the charge that you specify in SOURCE_CHARGE_ID, such as the charge definition, price type, frequency, and so on.

Guidelines for Charge Line Number

If you use the:

- **Create operation**. The import automatically assigns the Charge Line Number and you don't need to specify a value in the CHARGE_LINE_NUMBER attribute on the QP_PRICE_LIST_CHARGES_INT worksheet.
- **Update operation**. You must set the CHARGE_LINE_NUMBER attribute on the QP_PRICE_LIST_CHARGES_INT worksheet to the same value that the Charge Line Number attribute in the Oracle database already contains for this charge on the price list. The import typically uses the Source Charge Id attribute to identify the record. If the import can't identify the record, then it uses the Charge Line Number to identify it.

Import or Update a Large Number of Pricing Tiers

You can periodically import tier data for your price lists in batches. Assume you update your tier data during each quarter of the year, and you need to periodically import this data into Pricing Administration. You can use batch import to import tier data.

You can:

- Manage tiered adjustments for price list charges.
- Create, append, or update tier headers and tier lines, including data that you store in a flexfield.
- Monitor the import and use error messages to troubleshoot problems that happen during import.

Realize these benefits.

- Migrate data from your source system into Pricing Administration.
- Efficiently create and update pricing tiers on your price lists.
- Quickly update a large number of tier headers and lines.
- Periodically maintain the data that you migrate.
- Use a stable, repeatable, and efficient process.

Guidelines

- Use the Tier Headers and Tier Lines worksheets in the PriceListsImportBatchTemplate.xlsm import template.
- Set the Operation Code attribute on each worksheet to CREATE, UPDATE, or NO-OP.
- Use the Tier Line Number attribute to update a tier line that already exists.
- Use the output file from the scheduled process that you use during import to get details about the tier headers and tier lines that you imported.
- Update tier data that you create through some other technology, such as through the Pricing Administration work area, ADFDI, REST API, or another File-Based Data Import template.

Delete Pricing Data from Interface Tables

If you use file-based data import to periodically import pricing and discount data, then the interface tables might grow in size when you process new batches. You might notice a gradual decrease in performance when you do your imports. To avoid this problem, you can use the *Delete Pricing Data from Interface Tables* scheduled process to delete data from the interface tables after you import each batch.

Try it.

1. Go to the Scheduled Processes work area, then run the scheduled process that you use to import your pricing data, such as Import Price Lists Batch.
2. Wait for the scheduled process to finish.
3. Run the Delete Pricing Data from Interface Tables scheduled process.

Note

- Set up a schedule when you run this process so it runs at regular intervals. For example, run it one time every morning at 2AM. This will help prevent the tables from becoming really big with lots of old data. Big tables with old data degrade performance.
- You can set the Delete Option parameter only to one value, which is to delete all import data from the interface tables. This process deletes all data from all interface tables regardless of your import's status.
- You can run the Delete Pricing Data from Interface Tables scheduled process only after you finish the import. You can't run it at the same time that you run these scheduled processes:
    - Import Price Lists Batch
    - Import Discount Lists

More

Import or Update a Large Number of Coverage Items for Price Lists

Related Topics

- Manage Price Lists
- Guidelines for Importing Batches of Discount Lists

Troubleshoot Importing Price Lists

Click the job for the scheduled process that you run when you use the Price List Import Template, open the attachments in the details section, then examine the log files.

Log File	Description
.txt files created for the Import Price Lists Batch Header job, the Access Set, or the Import Price List Items job.	Describes the error.

Log File	Description
Log files created for the Import a Price List job or the Import Price List job.	Describes the status of the child jobs.

Tips for Using Scheduled Processes

- Make sure the status of the Load Interface file for the scheduled process is Succeeded and isn't in a Warning or Error state.
- Make sure you correctly set the parameters when you run the *Import Price Lists Batch* scheduled process.
- Examine the server logs. If the scheduled process doesn't create the .txt file for the job, then a problem might exist with the database or some unknown exception might have happened. Examine the server logs for the job that doesn't create a .txt file.
- Examine the interface tables. See the section later in this topic.
- If you have to run more than one batch, then run them sequentially, one after the other. Don't run them at the same time. This will help to improve performance.
- Run your scheduled processes during nonpeak hours. For example, if you typically don't do many business transactions between midnight and 8AM, then schedule them to run at 2AM.

Other Tips you Can Try

- Make sure you include only price lists in PriceListsImportBatchTemplate.xlsm and only discount lists in DiscountListImportTemplate.xlsm.
- Don't include discount lists PriceListsImportBatchTemplate.xlsm.
- Don't include price lists in DiscountListImportTemplate.xlsm.
- Keep your template files up to date. Oracle periodically modifies the templates. Download the latest file with each update.
- Make sure you set up your pricing parameters, including the Default Price Periodicity UOM Class parameter. Pricing uses these parameters during your import to validate your import data. For details, see *Manage Pricing Parameters*.
- Make sure you set up your administrators. For details, see *Set Up Roles and Privileges for Pricing Administrators*
- Make sure the source Id is unique and meaningful. For example, make sure the SOURCE_PRICE_LIST_ID attribute for the Price List Item is unique in the context of the price list. This helps the import process know how to handle the attribute. For example, if an identifier attribute isn't part of an entity, such as a price list header, and if the identifier isn't unique, then the import can't create a relationship between it and the entity.
- You can use a computer that runs on the Macintosh operating system only with PriceListsImportBatchTemplate.xlsm. You can't use it with PriceListImportTemplate.xlsm.
- Make sure the value that you enter for the Batch Name is meaningful. For example, don't use a number. Instead, enter text that describes what you are uploading, such as Import Discount Lists from Vision Operations.

Dates

Make sure you use the correct format for each data type in your import payload. For example, you must use the `yyyy/mm/dd hh:mm:ss` format for most date attributes. The descriptions in the FBDI spreadsheet tells you the format you must use.

If you use the Date data type or the Timestamp data type in the Domain attribute of a pricing matrix's condition column, then you must represent that date as a string in the FBDI spreadsheet. Use this format in the spreadsheet:

```
'yyyy-mm-dd hh:mm:ss
```

where

- The first character is an apostrophe. The apostrophe instructs FBDI to store the value as a string and prevents FBDI from using Excel's standard date format.
- Include a dash between year and month, and between month and day.

For example:

```
'2024-05-24 18:45:00
```

Examine the Interface Tables

1. Get administrative access so you can query the Oracle database.
2. Query the interface tables, then examine the status:

Status	Description
NEW	Not processed.
ERROR	Processed but includes a data error.
VALIDATED	Processed but not imported because one or more children contain an error.
IMPORTED	Successfully imported.

   For details, see the Interface Tables That the Price List Import Template Uses section in this topic.
3. Examine the IMPORT_STATUS_CODE column of the QP_PRICE_LISTS_INT table. This column describes the overall status of the price list:

Status	Description
COMPLETED	The scheduled process imported the price list and each child entity.
PARTIAL	The scheduled process imported only part of the price list.

Here's the hierarchy that Pricing uses. Use the status of each item in this hierarchy to help isolate where the error happens.

```
Price List Header
 Item
 Component Item
```

```
Charge
Charge
Tier
Tier Header
Tier Line
Matrix
Matrix Dimension
Matrix Rules
```

For example, Item is a child of Price List Header, Component Item is a child of Item, and Charge is a child of Component Item.

Pricing processes each child entity only if the parent is valid. If the parent:

- **Doesn't fail validation.** If a child errors, then the status of the parent is VALIDATED and the child status is ERROR, VALIDATED, or NEW.
- **Fails validation.** The parent status is ERROR and the status for each child is NEW.

For example, if a charge error happens, then the hierarchy will contain these values.

```
Price List Header, IMPORTED
 Item, VALIDATED
 Component item
 Charge
 Charge
 Tier, NEW
 Tier Header
 Tier Line
 Matrix, NEW
 Matrix Dimension
 Matrix Rules
```

For another example, if a tier header error happens, then the hierarchy will contain these values.

```
Price List Header, IMPORTED
 Item, VALIDATED
 Component item
 Charge
 Charge, VALIDATED
 Tier
 Tier Header
 Tier Line, NEW
 Matrix, VALIDATED
 Matrix Dimension
 Matrix Rules
```

Related Topics

- Manage Price Lists
- Guidelines for Importing Batches of Discount Lists

Use REST API to Manage Pricing Details

You can use REST API to create, read, update, or delete pricing details.

Create and Approve a Price List

Assume you need to create a price list named My Price List for the Vision Operations business unit. You can use REST API to create and approve it.

1. Send a REST API request to create the price list:
    - Use a POST action with the priceLists resource.

      Here's the cURL command.

      ```
      https://servername/fscmRestApi/resources/version/priceLists
      ```

    - Here's the body.

```
"PriceListName": "My Price List",
"PriceListDescription": "Created from REST API",
"PriceListType": "Segment price list",
"PriceListTypeCode": "SEGMENT",
"BusinessUnit": "VISION OPERATIONS",
"Currency": "US Dollar",
"CurrencyCode": "USD",
"Status": "Approved",
"StatusCode": "APPROVED",
"StartDate": "2021-01-01T19:58:00+00:00",
"EndDate": null,
"PricingChargeDefinition": "Sale Price",
"PricingChargeDefinitionCode": "QP_SALE_PRICE",
"LineType": "Buy",
"LineTypeCode": "ORA_BUY",
"CalculationMethod": "Price",
"CalculationMethodCode": "PRICE",
"items": [
{
"Item": "PMC - Std Item",
"PricingUOM": "Each",
"PricingUOMCode": "Ea",
"LineType": "Buy",
"LineTypeCode": "ORA_BUY",
"PrimaryPricingUOM": "Y",
"ServiceDurationPeriod": null,
"ServiceDurationPeriodCode": null,
"ServiceDuration": null,
"ItemLevelCode": "ITEM",
"ItemLevel": "Item",
"charges": [
{
"PricingChargeDefinition": "Sale Price",
"PricingChargeDefinitionCode": "QP_SALE_PRICE",
"PricePeriodicity": null,
"PricePeriodicityCode": null,
"CalculationMethod": "Price",
"CalculationMethodCode": "PRICE",
"CalculationType": null,
"CalculationTypeCode": null,
"AllowManualAdjustment": "Y",
"BasePrice": 10,
"CostCalculationAmount": null,
"StartDate": "2000-04-06T22:30:00+00:00",
"EndDate": null,
"ChargeLineNumber": 1,
"MatrixId": null
```

```
 }
 ]
 }
 ],
 "accessSets": [
 {
 "AccessSetName": "Common Set",
 "AccessSetCode": "COMMON",
 "AccessSetId": 0,
 "AccessSetDescription": null
 }
 ]
}
```

- Assume you receive a response.

```
"PriceListName": "My Price List",
"PriceListId": 300000081320934,
"PriceListDescription": "Created from REST API",
"PriceListType": "Segment price list",
"PriceListTypeCode": "SEGMENT",
"BusinessUnit": "VISION OPERATIONS",
"BusinessUnitId": 300000002843138,
"Currency": "US Dollar",
"CurrencyCode": "USD",
"Status": "Approved",
"StatusCode": "IN_PROGRESS",
"StartDate": "2021-01-01T19:58:00+00:00",
. . .
```

Notice that your request includes `"Status": "Approved"`, and `"StatusCode": "APPROVED"`, but the response includes `"Status": "Approved"`, and `"StatusCode": "IN_PROGRESS"`. The POST action can set StatusCode to IN_PROGRESS but not APPROVED. Next, you will send a PATCH request to set StatusCode to APPROVED.

2. Approve the price list. Use a PATCH action with the priceLists resource:

   - Here's the cURL format.

     ```
     https://servername/fscmRestApi/resources/version/priceLists/PriceListId
     ```

     Here's the cURL command for this example.

     ```
     https://servername/fscmRestApi/resources/version/priceLists/300000081320934
     ```

   - Here's the body.

     ```
     {
      "Status": "Approved",
      "StatusCode": "APPROVED"
     }
     ```

Troubleshoot REST API

Trouble	Shoot
I use the value SYSDATE to set the start date for the unit price for my item in a priceSalesTransaction REST API payload. However, after I finish the import, the date	Make sure you use the correct UTC format, which is YYYY-MM-DDTHH:MM:SS. For example, 2019-08-20T10:40:51.88.

Trouble	Shoot
that I see in the Pricing Administration work area is different than what I expected.	Note that UTC is on 24 hour clock, not 12 hour. For details, see: • *Set the Time Zone for Pricing Administration* • *Manage Time Zones in Order Management* • *Import Source Orders That Include Coverages and Subscriptions*

For more, see *Troubleshoot Your Pricing Setups* and *Manage Price Lists That Have Rate Plans*.

Import Discount Lists

Guidelines for Importing Batches of Discount Lists

Save time and work more efficiently. Import a batch of discount lists instead importing them individually.

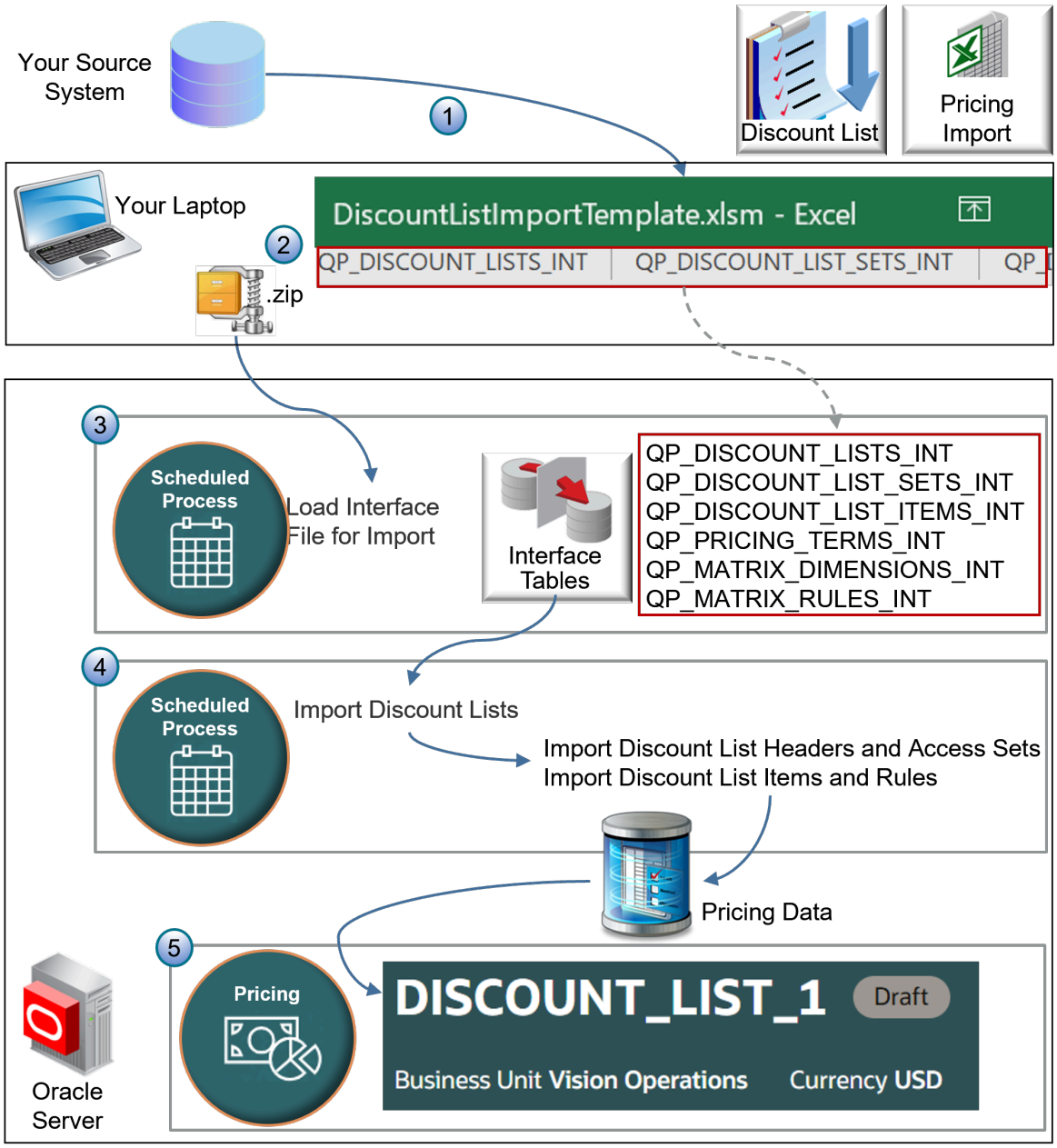
Assume you maintain pricing details in a source system that resides outside of Oracle Pricing. You need to periodically import discount lists through data files into Pricing. You prefer to use an automated process that accepts your pricing data files, validates them, then stores your data in Oracle Pricing.

You can:

- Manage data for more than one discount list in a single batch.
- Reuse your batch during a subsequent import.
- Import entities for more than one discount list:
    - Headers and access sets
    - Items
    - Pricing terms for simple rules
    - Pricing terms, matrix dimensions, and matrix rules for attributes
    - Descriptive flexfields
- Use output files and error logs to examine details about the number of records imported, records that are in error, and to get suggestions on how to fix errors.

Use an Established Import Procedure

Here's a summary of how you import:



Assume you import a discount list named DISCOUNT_LIST_1:

1. Copy discount list data from your source system into the DiscountListImportTemplate.xlsm Excel workbook.
2. Click **Generate CSV File** in DiscountListImportTemplate.xlsm to create a zip file.
3. Run the Load Interface File for Import scheduled process. See *Load Interface File for Import Process*.

   Each worksheet in DiscountListImportTemplate represents an interface table. For example, the QP_DISCOUNT_LISTS_INT sheet uses the same structure that the QP_DISCOUNT_LISTS_INT interface table on the Oracle server uses. The Load Interface File for Import process imports data from your zip file into the interface tables.
4. Run the *Import Discount Lists* scheduled process. This parent process creates two child scheduled processes:
   - Import Discount List Headers and Access Sets

- Import Discount List Items and Rules

They import your pricing data from the interface tables into the Oracle database.

5. Go to the Pricing Administration work area and verify the import.

   The work area displays pricing data from the Oracle database. DISCOUNT_LIST_1 includes any child objects that you imported, such as items, pricing terms, and pricing matrixes.

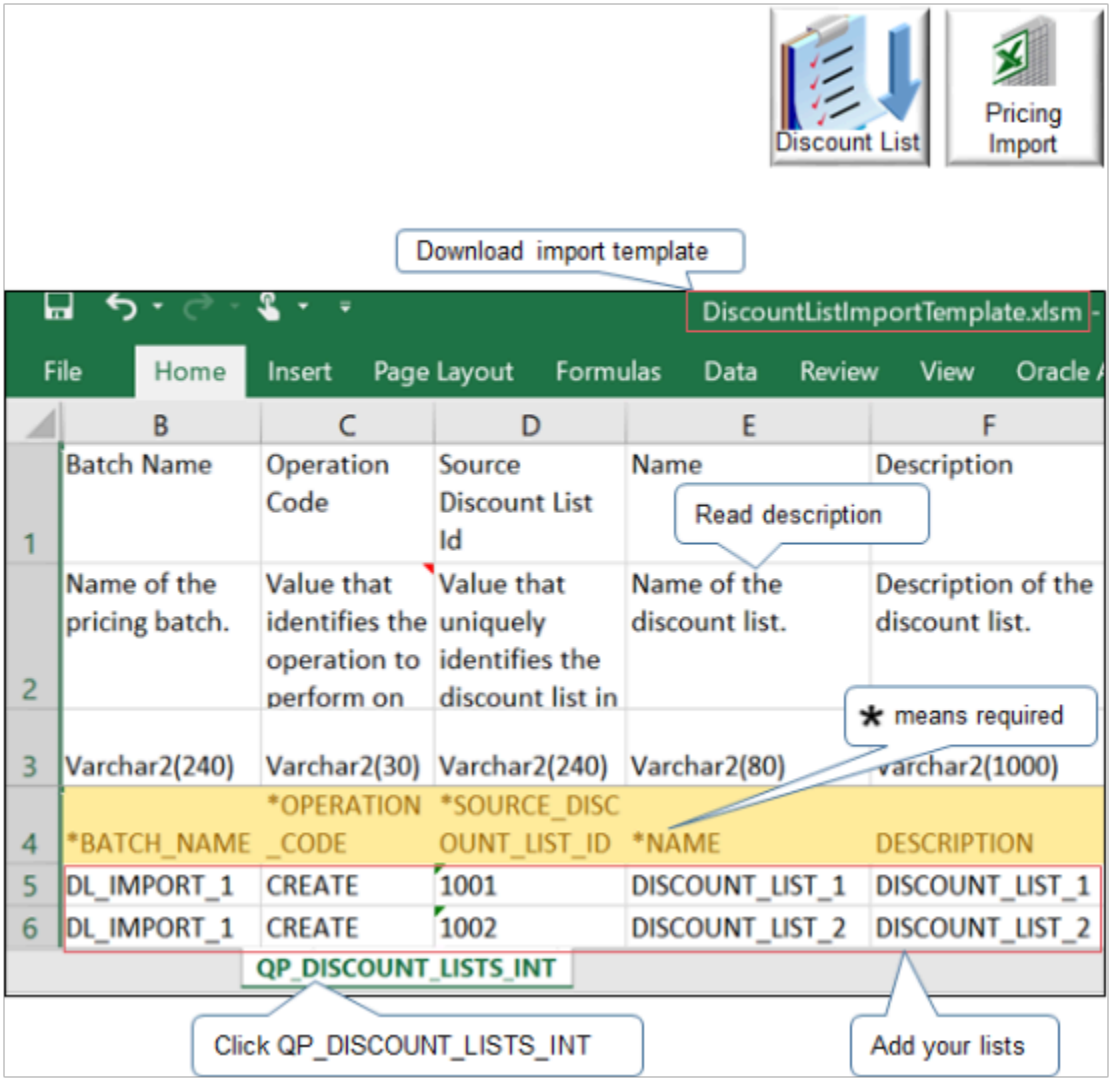
Interface Tables That the Discount List Import Template Uses

- QP_DISCOUNT_LISTS_INT
- QP_DISCOUNT_LIST_SETS_INT
- QP_DISCOUNT_LIST_ITEMS_INT
- QP_PRICING_TERMS_INT
- QP_MATRIX_DIMENSIONS_INT
- QP_MATRIX_RULES_INT

where

- QP is a code for pricing.
- INT is an abbreviation for the word interface.

Add Discount Lists



Note

- Use the DiscountListImportTemplate.xlsm template.
- Use the QP_DISCOUNT_LISTS_INT worksheet.
- Read the name and description to clarify the data you must add.
- Add one row for each discount list.
- An asterisk (*) in the worksheet means you must include a value in the column. For example, you must include a value in the BATCH_NAME column for each list you import.
- A double asterisk (**) indicates a set of required columns. You must include a value for at least one column in each set. For example, the template uses double asterisks for BUSINESS_UNIT_ID and BUSINESS_UNIT_NAME on sheet QP_DISCOUNT_LISTS_INT to indicate they constitute one set. You must include a value for at least one of these columns.

- Set the values:

Attribute	Value
Batch Name	Enter any alphanumeric value. The import uses the value that you enter as a unique identifier for the data that you import. You select this batch name when you run the Import Discount Lists scheduled process. You can reuse a batch over and over.
Operation Code	Set a value: o CREATE. Create a new record for your discount list. o UPDATE. Update a discount list that already exists. o NO-OP. It means NO OPeration, or don't do any operation. Use it to create a relationship with another worksheet. o For example, assume you have an existing price list and charges for that list. If you want to update only the charges, use No-Op for the price list, and use Update for the charges. o Use NO-OP on the discount list header and CREATE for an entity in the list. For example, use NO-OP on the QP_DISCOUNT_LISTS_INT worksheet for discount list 1001, and use CREATE for items on the QP_DISCOUNT_LIST_ITEMS_INT worksheet for discount list 1001. o You use the same set of operation codes on each worksheet.
SOURCE_DISCOUNT_LIST_ID	Enter any numeric value. Make sure the value is unique across worksheets.
NAME DESCRIPTION	Enter any alphanumeric value. The Pricing Administration work area will display these values.
BUSINESS_UNIT_ID	Enter the Id for the business unit you want to use for the discount list. For example, get the Id for the Vision Operations business unit: a. Go to the Setup and Maintenance work area, then go to the Manage Business Unit Set Assignment task. b. Select Manage Business Unit Set Assignment, set Business Unit to Select and Add, then click **Apply and Go to Task**. c. On the Manage Business Units page, search for Vision Operations. d. Click **View > Columns**, then add a check mark to BusinessUnitId. e. Notice the value in the BusinessUnitId column. Its 204 for Vision Operations.
BUSINESS_UNIT_NAME	Enter the name of the business unit.
CURRENCY_CODE	Get the currency:

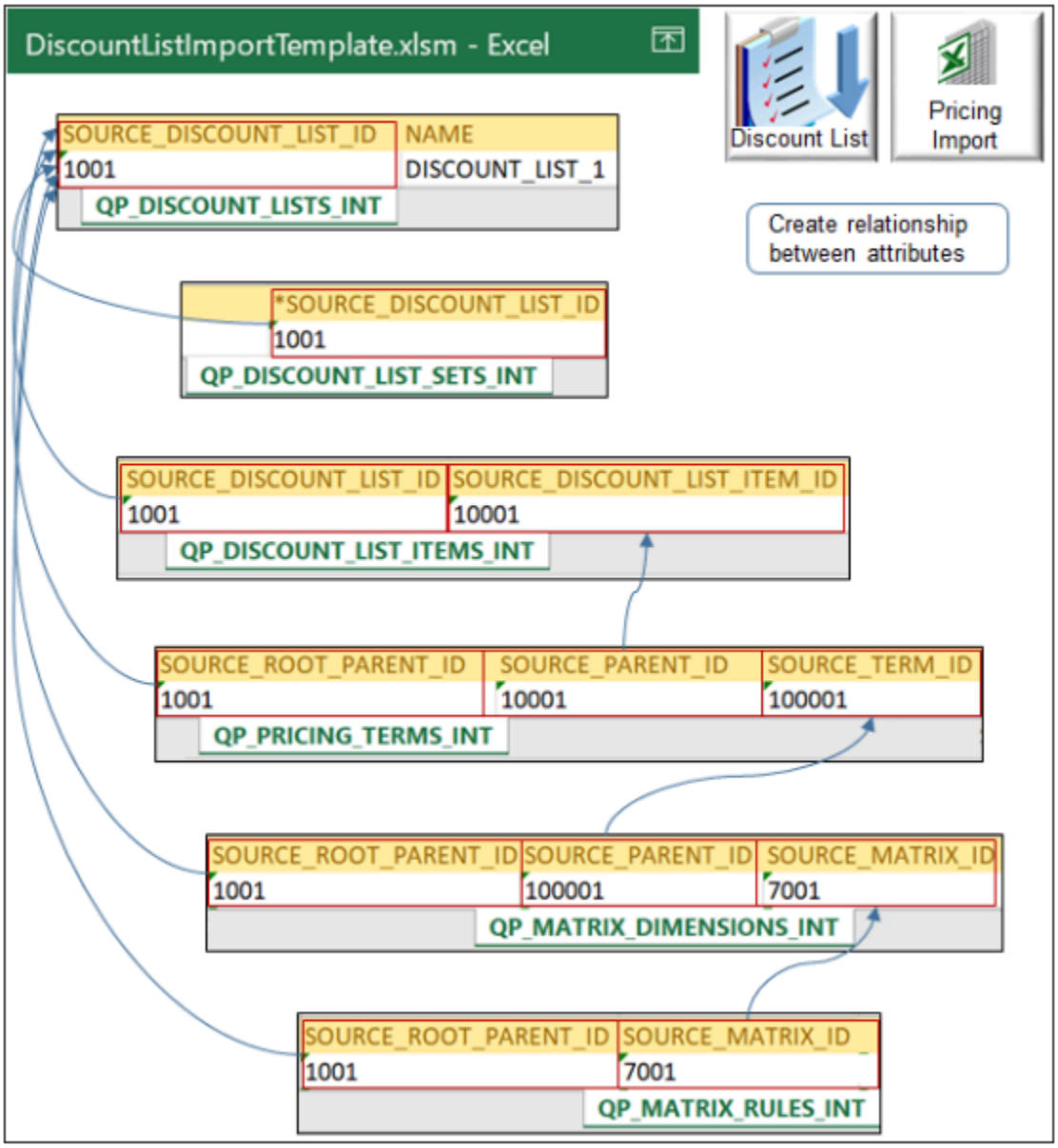
Attribute	Value
	a. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Currencies**. b. On the Manage Currencies page, leave the search attributes empty, then click **Search**. c. Examine the search results. Each row contains the Currency Name, such as US Dollar, and the Currency, such as USD. If necessary, use **View > Columns** to display the columns you need.

START_DATE	Enter a date for each of your discount lists.
	
	`yyyy/mm/dd hh:mm:ss`
	
	For example:
	
	`2019/01/15 10:15:20`
	
	where
	
	o 2019 is the year
	o 01 is the month
	o 15 is the day
	o 10 is the hour, using a 24 hour clock. For example, 10 means 10 AM. 23 means 11 PM.
	o 15 is the minutes
	o 20 is the seconds

STATUS_CODE	You must enter APPROVED or IN_PROGRESS. If you import with APPROVED, then make sure the access set you import exists in Oracle Financials and its assigned to a business unit.
ATTRIBUTE_CATEGORY ATTRIBUTE_CHARx ATTRIBUTE_NUMBERx ATTRIBUTE_DATEx ATTRIBUTE_TIMESTAMPx	Use these columns only if you use a flexfield with your discount list.

Create Relationships Between Entities

Use identifiers to create relationships between the parent entity and each child entity:



Note

- The SOURCE_DISCOUNT_LIST_ID on the QP_DISCOUNT_LISTS_INT worksheet is the root parent of all other entities. We use the term **root** when a hierarchy includes several levels, such as parent, child, and grandchild.
- QP_DISCOUNT_LISTS_INT is the only sheet that's required. You use other sheets depending on the data you import:

This Worksheet	Is a Parent of This Worksheet
QP_DISCOUNT_LISTS_INT	QP_DISCOUNT_LIST_SETS_INT
QP_DISCOUNT_LISTS_INT	QP_DISCOUNT_LIST_ITEMS_INT

This Worksheet	Is a Parent of This Worksheet
QP_DISCOUNT_LIST_ITEMS_INT	QP_PRICING_TERMS_INT
QP_PRICING_TERMS_INT	QP_MATRIX_DIMENSIONS_INT
QP_MATRIX_DIMENSIONS_INT	QP_MATRIX_RULES_INT

Create relationships between entities:

Use Column	In Child Worksheet	To Reference Column	In Parent Worksheet
SOURCE_ROOT_PARENT_ID For example, 1001.	QP_MATRIX_RULES_INT	SOURCE_DISCOUNT_LIST_ID	QP_DISCOUNT_LISTS_INT
SOURCE_MATRIX_ID For example, 7001.	QP_MATRIX_RULES_INT	SOURCE_MATRIX_ID	QP_MATRIX_DIMENSIONS_INT
SOURCE_ROOT_PARENT_ID For example, 1001.	QP_MATRIX_DIMENSIONS_INT	SOURCE_DISCOUNT_LIST_ID	QP_DISCOUNT_LISTS_INT
SOURCE_PARENT_ID For example, 100001.	QP_MATRIX_DIMENSIONS_INT	SOURCE_TERM_ID	QP_PRICING_TERMS_INT
SOURCE_ROOT_PARENT_ID For example, 1001.	QP_PRICING_TERMS_INT	SOURCE_DISCOUNT_LIST_ID	QP_DISCOUNT_LISTS_INT
SOURCE_PARENT_ID For example, 10001.	QP_PRICING_TERMS_INT	SOURCE_DISCOUNT_LIST_ITEM_ID	QP_DISCOUNT_LIST_ITEMS_INT
SOURCE_DISCOUNT_LIST_ID For example, 1001.	QP_DISCOUNT_LIST_ITEMS_INT	SOURCE_DISCOUNT_LIST_ID	QP_DISCOUNT_LISTS_INT
SOURCE_DISCOUNT_LIST_ID For example, 1001.	QP_DISCOUNT_LIST_SETS_INT	SOURCE_DISCOUNT_LIST_ID	QP_DISCOUNT_LISTS_INT

Create a relationship only when necessary. For example:

If You Don't Add Any Rows In	Then You Don't Create a Relationship Between
QP_MATRIX_RULES_INT	QP_MATRIX_RULES_INT and QP_MATRIX_DIMENSIONS_INT QP_MATRIX_RULES_INT and QP_DISCOUNT_LISTS_INT.
QP_MATRIX_DIMENSIONS_INT	QP_MATRIX_DIMENSIONS_INT and QP_PRICING_TERMS_INT QP_MATRIX_DIMENSIONS_INT and QP_DISCOUNT_LISTS_INT

Add Access Sets

Use QP_DISCOUNT_LIST_SETS_INT to specify the access set. In most situation, you use COMMON:

OPERATION_CODE	SET_CODE
CREATE	COMMON

The CREATE command doesn't create a new access set. It adds an access set that already exists to your discount list.

See what other access sets are available:

1. Go to the Pricing Administration work area and create a discount list.
2. Set the business unit.
3. Click **Access Sets**.
4. Click **Actions > Add Row**.
5. Search the Set Code column for the set you need.
6. Use the value in the Set Code column in the work area for the SET_CODE column in the worksheet.

Add Items

- Use the QP_DISCOUNT_LIST_ITEMS_INT worksheet.
- Add one row for each item.
- Set the values.

Tip: Use the Pricing Administration work area to determine the values you can use for some attributes. For example, to determine the values you can use for the item level, create a discount list in the Pricing Administration work area. On the Discount Lines tab, examine the values that are available in the Item Level attribute.

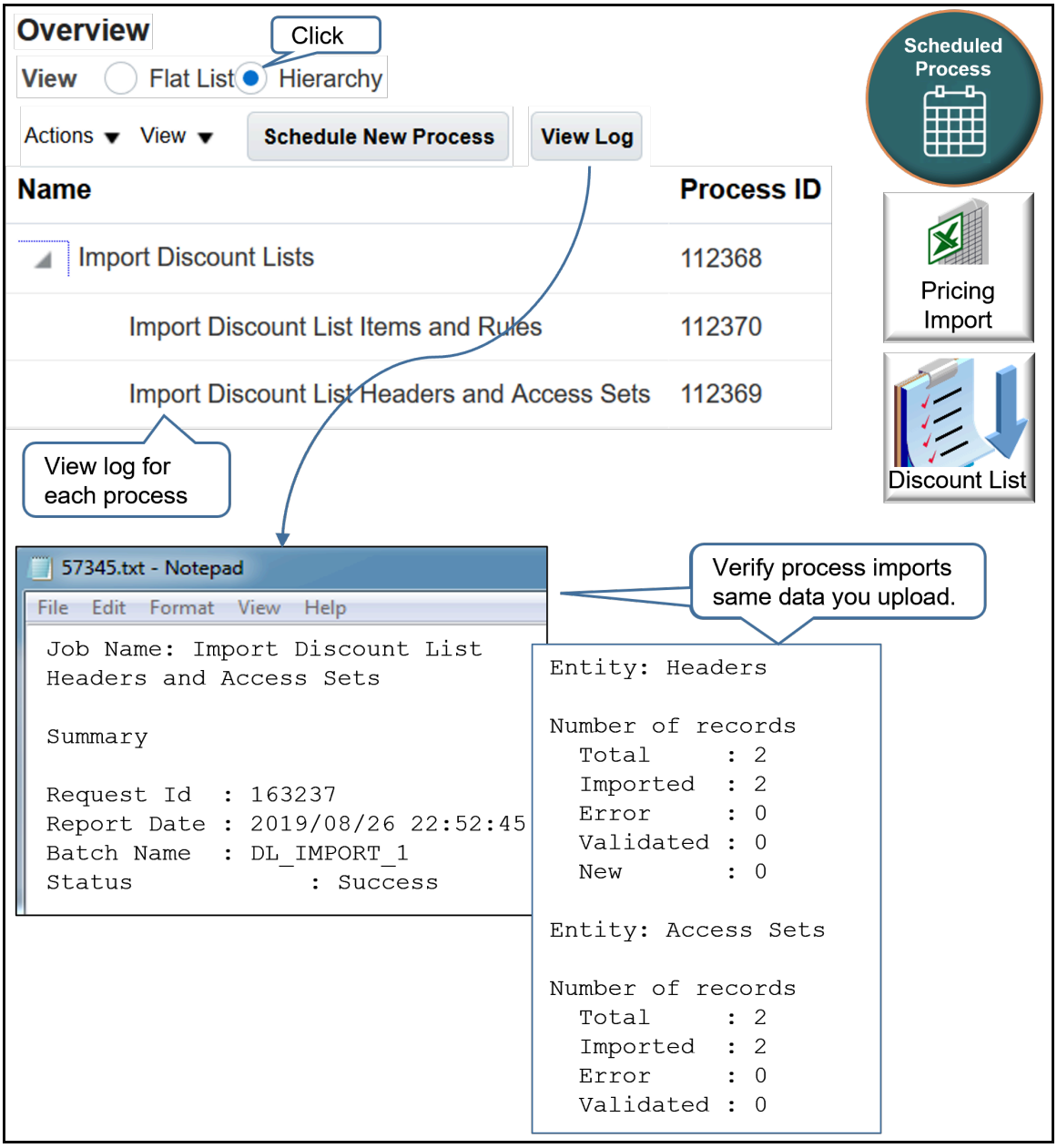
Attribute	Value
SOURCE_ROOT_PARENT_ID	The rule is a grandchild of the parent discount list. Enter the same value that you enter in SOURCE_DISCOUNT_LIST_ID on sheet QP_DISCOUNT_LISTS_INT.
SOURCE_PARENT_ID	The rule is a child of the parent item.

Attribute	Value
	Enter the same value that you enter in SOURCE_DISCOUNT_LIST_ITEM_ID on sheet QP_DISCOUNT_LIST_ITEMS_INT.
SOURCE_TERM_ID	Enter any numeric value. Make sure the value is unique across worksheets.
NAME	Enter any alphanumeric value.
PRICING_RULE_TYPE_CODE	Enter one of: • **SIMPLE**. Create a simple rule. • **ATTRIBUTE_PRICING**. Create a rule that evaluates according to an attribute. These values correspond to the same actions you can select in the Pricing Administration work area when you click Actions > Create in the Discount Rules area.
PRICE_TYPE_CODE	Enter one of: • ALL • ONE_TIME • RECURRING
CHARGE_TYPE_CODE	Try this: 1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Lookups**. 2. On the Manage Pricing Lookups page, search Lookup Type for ORA_QP_CHARGE_TYPES. You can use any value that displays in the Lookup Code column, such as ORA_SALE, ORA_SERVICE, and so on.
CHARGE_SUBTYPE_CODE	On the Manage Pricing Lookups page, search Lookup Type for ORA_QP_CHARGE_SUBTYPES. You can use any value that displays in the Lookup Code column, such as ORA_PRICE, ORA_FEE, and so on.
PRICE_PERIODICITY PRICE_PERIODICITY_CODE	Include a value only if you set PRICE_TYPE_CODE to RECURRING. Try this: 1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Units of Measure**. 2. On the Manage Units of Measure page, search Class Name for TIME. 3. Notice the values in the search results. You can use any value in the: o **UOM Name column for PRICE_PERIODICITY**. For example, Year, Month, Week, Day, Hour, Minute, Second, and so on. o **UOM Code column for PRICE_PERIODICITY_CODE**. For example, YR, MNTH, WK, HR, MIN, SEC, and so on.
ADJUSTMENT_TYPE_CODE	On the Manage Pricing Lookups page, search Lookup Type for ORA_QP_LINE_ADJ_TYPES. You can use any value that displays in the Lookup Code column, such as:

Attribute	Value
	• DISCOUNT_AMOUNT • DISCOUNT_PERCENT • MARKUP_AMOUNT • MARKUP_PERCENT • PRICE_OVERRIDE
ADJUSTMENT_AMOUNT	Enter any numeric value.
ADJUSTMENT_BASIS ADJUSTMENT_BASIS_ID	Identify the values you can select: 1. Create a discount list in the Pricing Administration work area. 2. In the Discount Rules area, click one of: o **Actions > Create > Tier Based Rule** o **Actions > Create > Attribute Based Rule** 3. In the Create Discount Rule dialog, set the values, then, in the Attribute Based Rule area, click **Actions > Add Row**. 4. Set the values in the row, then examine values you can select in the Adjustment Basis attribute. For example, here's one entry from the list. `QP_AdjBasisforListPrc 3000100071623810` Use the basis or the basis ID. For example: • Use QP_AdjBasisforListPrc for ADJUSTMENT_BASIS. or • Use 3000100071623810 for ADJUSTMENT_BASIS_ID.
APPLY_TO_ROLLUP_FLAG	Set a value: • **Y**. Apply rolled up charge. • **N**. Don't apply rolled up charge. For details, see *Set Up Pricing for Configuration Models*.

Verify Your Import

Verify That Your Scheduled Processes Finished Successfully



Note

- Run your scheduled processes, then click **View Log** to examine the log to verify the records you import.
- Select the Hierarchy option to help visualize parent and children.
- Examine the log for the parent Import Discount Lists process and each child process.

- Verify the scheduled processes finished successfully.

  For example, if you import one batch that includes 10 discount lists, and these 10 lists include 1,000 items and 1,000 pricing terms, then the log for the parent should include values like these:

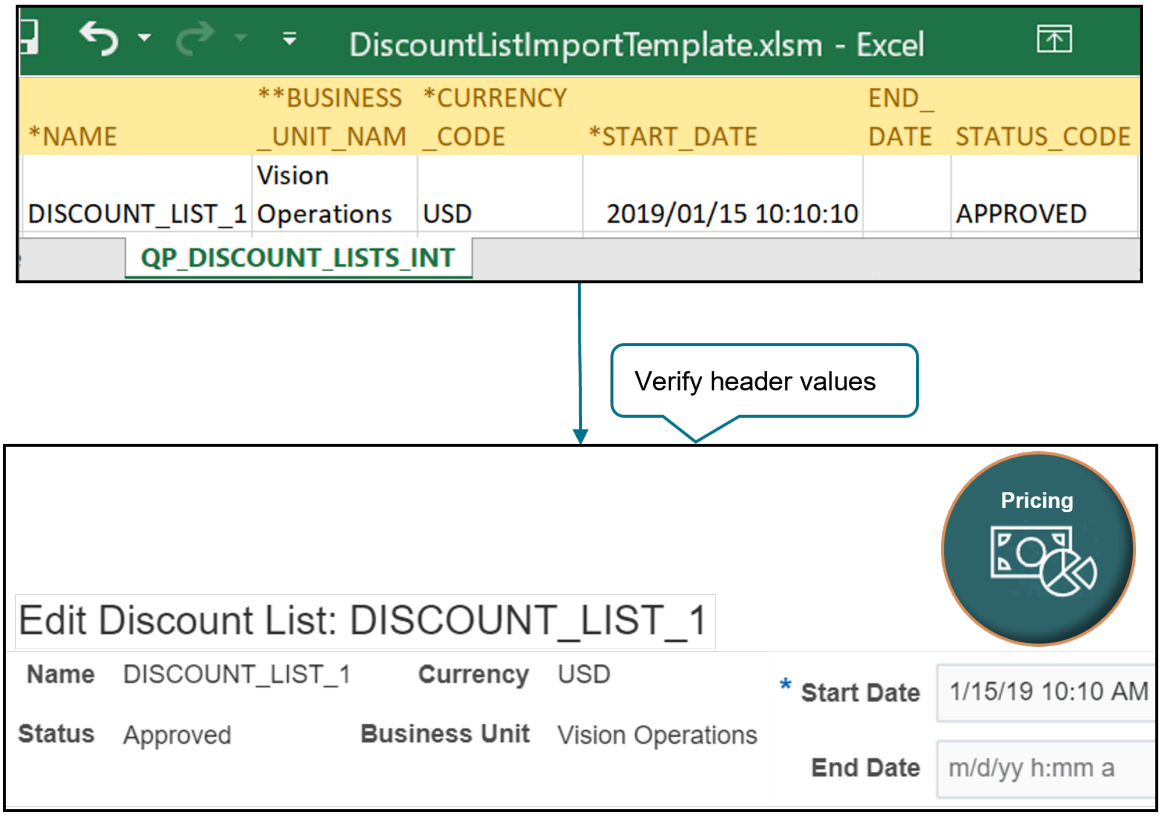
Entity	Values
Headers	Total 10 Imported 10
Items	Total 1000 Imported 1000
Terms	Total 1000 Imported 1000

- The process stamps data with the UTC time zone from the server. You can't change the time zone.
- The process uses English for translation. You can't use more than one language.
- The log includes the total number of records it imports, even records that include a NO-OP operation.
- If the status for a scheduled process is Error, then:
  - Click **Error** in the Status column.
  - In the Log and Output area, next to Attachment, click the **link**, such as ESS_O_112369, then examine the log. Click **more** to examine the log and a text file that includes more detailed instructions.
  - Examine the log for the parent and each child scheduled process. The logs for child processes typically contain more helpful details.

Verify Header Values

Verify you successfully imported header values:

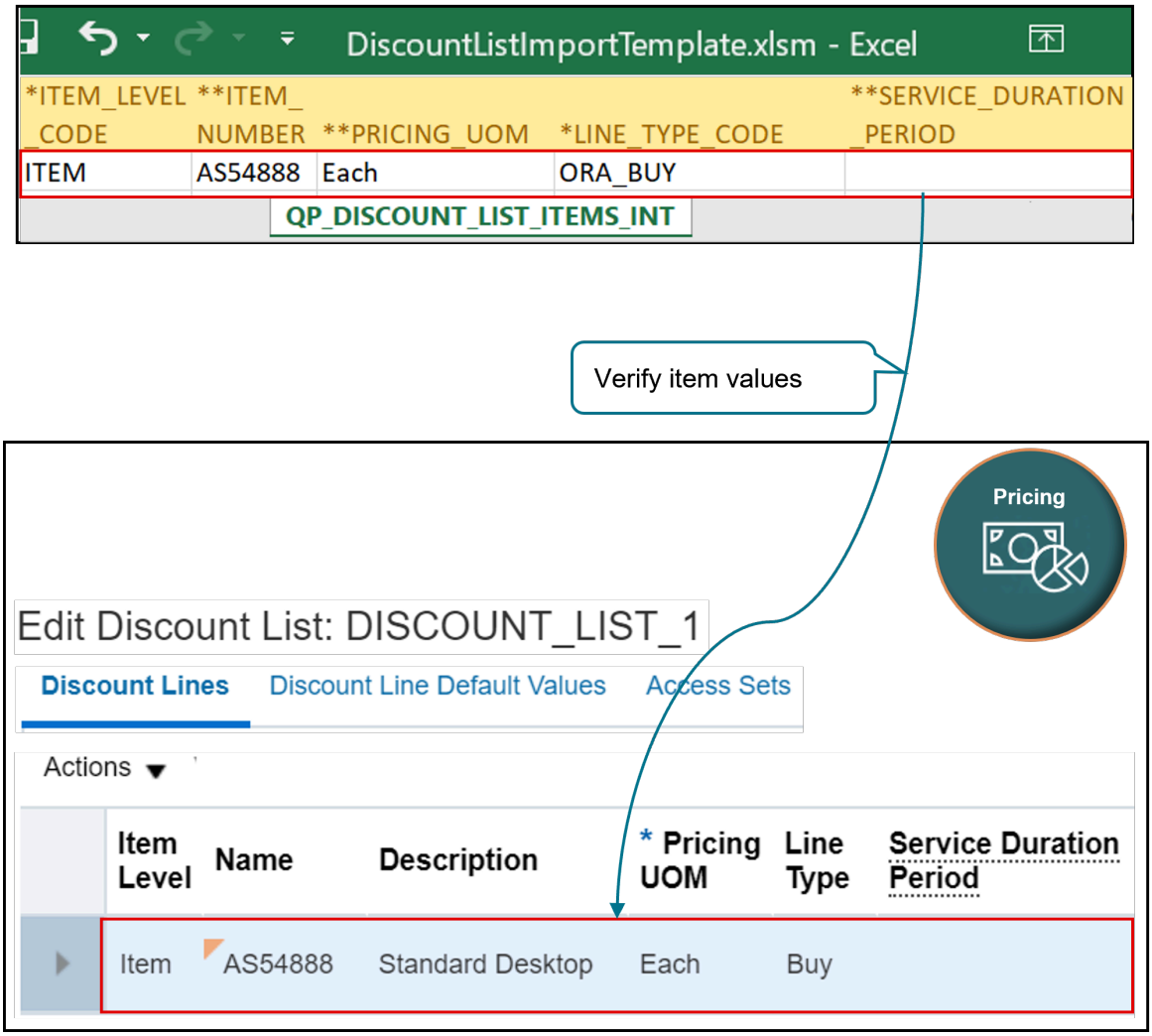




Finish your import, then go to the Pricing Administration work area and verify that the import imported your discount list and header values from the QP_DISCOUNT_LISTS_INT worksheet, such as Name, Status, Currency, Business Unit, Start Date, and so on.

Verify Items

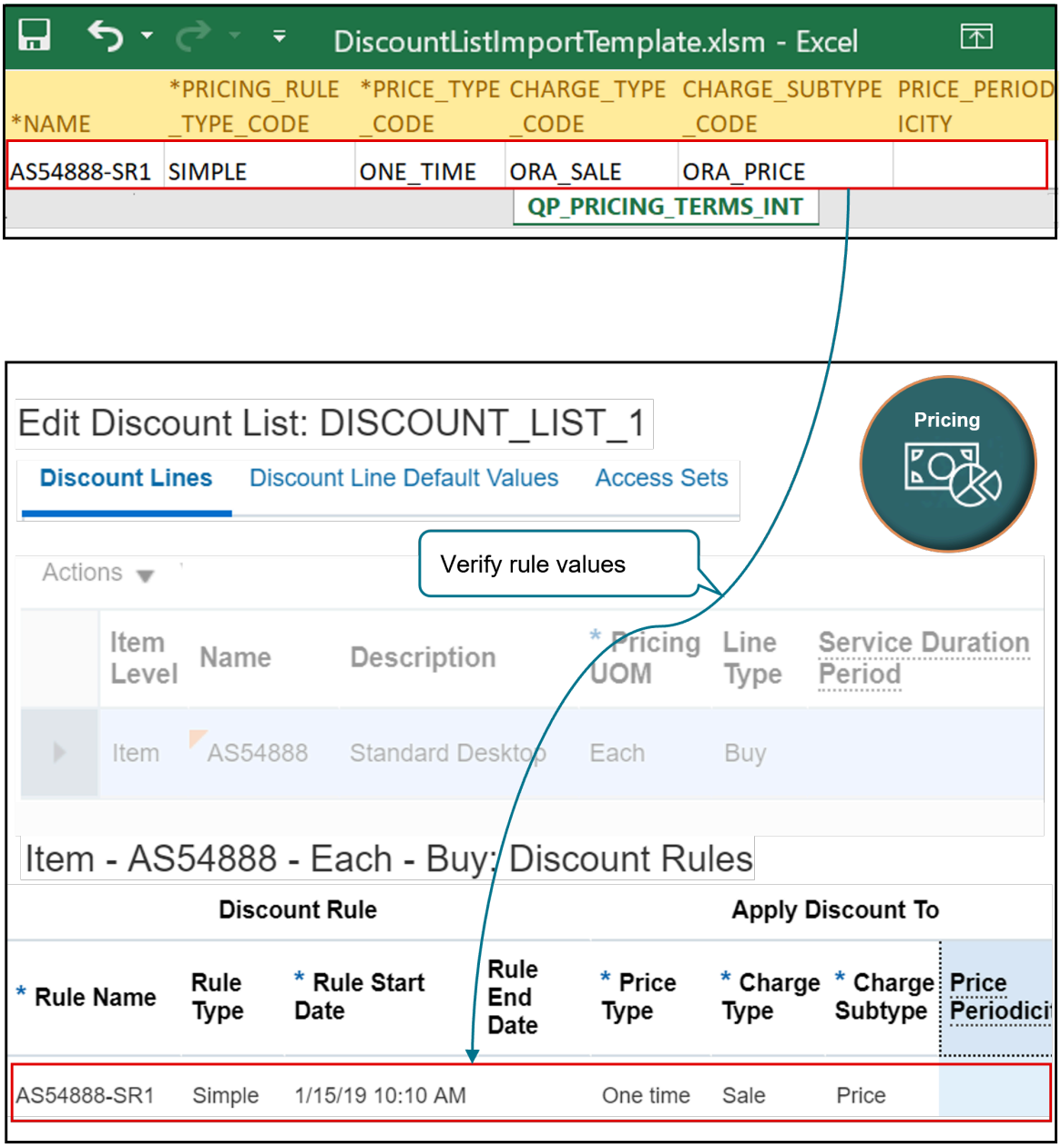




Note

- Search for your item in the Discount Lines area.
- Verify that the import imported values from the QP_DISCOUNT_LIST_ITEMS_INT worksheet, such as Item Level, Name, Pricing UOM, Line Type, and so on.
- Some values don't import but instead come from Product Management, such as Description.
- If you import more than one item, then search for and verify each item.

Verify Rules



Note

- Examine rules that you import in the Discount Rules area.
- Verify that the import imported values from the QP_PRICING_TERMS_INT worksheet, such as Rule Name, Rule Type, Rule Start Date, Price Type, and so on.
- If you import more than one rule, then verify each rule.

Update Your Discount List

Use the UPDATE operation to update a discount list that already exists. You must include values in the BATCH_NAME, SOURCE_DISCOUNT_LIST_ID, and NAME columns.

For example, assume the header on your current discount list includes values:

SOURCE_DISCOUNT_LIST_ID	NAME	DESCRIPTION	BUSINESS_UNIT_ID	BUSINESS_UNIT_NAME	CURRENCY_CODE	START_DATE	END_DATE	STATUS_CODE
1001	DISCOUNT_LIST_1	DISCOUNT_LIST_1	204	Vision Operations	USD	2014/01/15 10:10:10	2020/01/15 10:10:10	APPROVED

If you include a value in the template for the END_DATE attribute but leave the other attributes that currently contain values in the list empty, then UPDATE updates END_DATE, doesn't update any other attributes in the row, and doesn't replace the discount list.

If you include a value for all attributes that the current list contains, then the update replaces the discount list with a new list that includes your new values. In this example, to replace the current list, you would need to add a value for each of these attributes:

- NAME
- DESCRIPTION
- BUSINESS_UNIT_ID
- BUSINESS_UNIT_NAME
- CURRENCY_CODE
- START_DATE
- END_DATE
- STATUS_CODE

Your update or replace would look this in the template:

Type of Update	BATCH_NAME	OPERATIO CODE	SOURCE_DISCOUNT LIST_ID	NAME	DESCRIPTI	BUSINESS_ UNIT_ID	BUSINESS_ UNIT_ NAME	CURRENC CODE	START_ DATE	END_ DATE	STATUS_ CODE
Update	DL_IMPORT_2	UPDATE	1001	DISCOUNT LIST_1	-	-	-	-	-	2020/01/1 10:10:10	-
Replace	DL_IMPORT_2	UPDATE	1001	DISCOUNT LIST_1	DISCOUNT LIST_1	204	Vision Operations	USD	2014/01/15 10:10:10	2020/01/1 10:10:10	APPROVED

Note

- There are some attributes you can't update, such as Business Unit.
- You can update the currency only if the discount list doesn't contain a pricing terms.
- You can update the status code from In Progress to Approved but not from Approved to In Progress.

Attributes You Can Update

Worksheet	Attributes You Can Update
QP_DISCOUNT_LISTS_INT	Status Code Currency Start Date End Date Descriptive Flexfields
QP_DISCOUNT_LIST_ITEMS_INT	Descriptive Flexfields
QP_PRICING_TERMS_INT	Price Type Charge Type Charge Subtype Price Periodicity Price Periodicity Code Adjustment Basis Adjustment Basis Id Adjustment Type Adjustment Amount Start Date End Date Descriptive Flexfields
QP_MATRIX_DIMENSIONS_INT	-
QP_MATRIX_RULES_INT	Value Strings 1 through 10 Start Date End Date

Related Topics

- Set Up Pricing for Configuration Models
- Overview of Data Access Set Security

Import Batches of Discount Lists

Use an Excel spreadsheet to import a batch of discount lists into Oracle Pricing.

In this example, you import two discount lists in one batch. Each list contains one item:

List	Item
DISCOUNT_LIST_1	AS54888
DISCOUNT_LIST_2	CN92777

You use the same procedure that you use when you import a price list, but with a few important differences. For details, see *Import Price Lists*.

Summary of the Setup

1. Prepare your data.
2. Upload your data.
3. Import your data.
4. Verify your import.

You can use FBDI to import pricing details on the Microsoft Windows or on the Apple macOS operating systems.

Prepare Your Data

1. Download the DiscountListImportTemplate.xlsm file:
    - Go to *File-Based Data Import (FBDI) for Oracle SCM*.
    - Select your update and click **Books**.
    - In the Development area, under File Based Data Import for Oracle Supply Chain Management Cloud, click **HTML**.
    - Expand Order Management, then click **Discount List Import**.
    - In the Discount List Import area, click **DiscountListImportTemplate.xlsm**.

    To get a file that already includes the values that you use in this example, go to *Technical Reference for Order Management (Doc ID 2051639.1)*, download the Payloads and Files attachment, open it, then open DiscountListImportTemplate_example_1.xlsm.

    You must download and use DiscountListImportTemplate.xlsm from *File-Based Data Import (FBDI) for Oracle SCM*. Don't use ManageDiscountLists.xlsx that you can download from the Pricing Administration work area.

2. Add discount lists.

   Click **QP_DISCOUNT_LISTS_INT**, then add your discount lists to the template. Add two rows, one row for each discount list:

BATCH_ NAME	OPERATION CODE	SOURCE_ DISCOUNT_ LIST_ID	NAME	DESCRIPTIC	BUSINESS_ UNIT_ID	BUSINESS_ UNIT_ NAME	CURRENCY_ CODE	START_ DATE	STATUS_ CODE
DL_ IMPORT_1	CREATE	1001	DISCOUNT_ LIST_1	DISCOUNT_ LIST_1	204	Vision Operations	USD	1/15/2019 10:10:10 AM	APPROVED
DL_ IMPORT_1	CREATE	1002	DISCOUNT_ LIST_2	DISCOUNT_ LIST_2	204	Vision Operations	USD	1/15/2019 10:10:10 AM	IN_ PROGRESS

   Here's more detail:

Attribute	Value
Batch Name	The example uses DL_IMPORT_1 as the Batch Name in rows 5, and 6, so the batch will include two different discount lists: o DISCOUNT_LIST_1 o DISCOUNT_LIST_2
Operation Code	CREATE
SOURCE_DISCOUNT_LIST_ID	Enter: o `1001 for DISCOUNT_LIST_1` o `1002 for DISCOUNT_LIST_2`
NAME DESCRIPTION	Enter: o `DISCOUNT_LIST_1` o `DISCOUNT_LIST_2` The Pricing Administration work area will display these values.
BUSINESS_UNIT_ID	204 204 is the Business Unit ID for Vision Operations.
BUSINESS_UNIT_NAME	Vision Operations

Attribute	Value
CURRENCY_CODE	USD
START_DATE	2019/01/15 10:15:20
STATUS_CODE	Enter two different values: o `APPROVED for DISCOUNT_LIST_1` o `IN_PROGRESS for DISCOUNT_LIST_2`

Leave all other columns empty.

3. Add access sets.

   Click **QP_DISCOUNT_LIST_SETS_INT**, then add one access set for each discount list:

OPERATION_CODE	SOURCE_DISCOUNT_LIST_ID	SOURCE_DISCOUNT_LIST_SET_ID	SET_ID	SET_CODE
CREATE	1001	100	Leave this empty	COMMON
CREATE	1002	101	Leave this empty	COMMON

4. Add items.

   Click **QP_DISCOUNT_LIST_ITEMS_INT**, then add your items. Add two rows, one row for each item:

OPERATION_CODE	SOURCE_DISCOUNT_LIST_ID	SOURCE_DISCOUNT_LIST_ITEM_ID	ITEM_LEVEL_CODE	ITEM_NUMBER	ITEM_ID	PRICING_UOM	PRICING_UOM_CODE	LINE_TYPE_CODE
CREATE	1001	10001	ITEM	AS54888	Leave this empty	Each	Ea	ORA_BUY
CREATE	1002	10002	ITEM	CN92777	Leave this empty	Each	Ea	ORA_BUY

5. Add a discount rule.

   Click QP_PRICING_TERMS_INT, then add a rule:

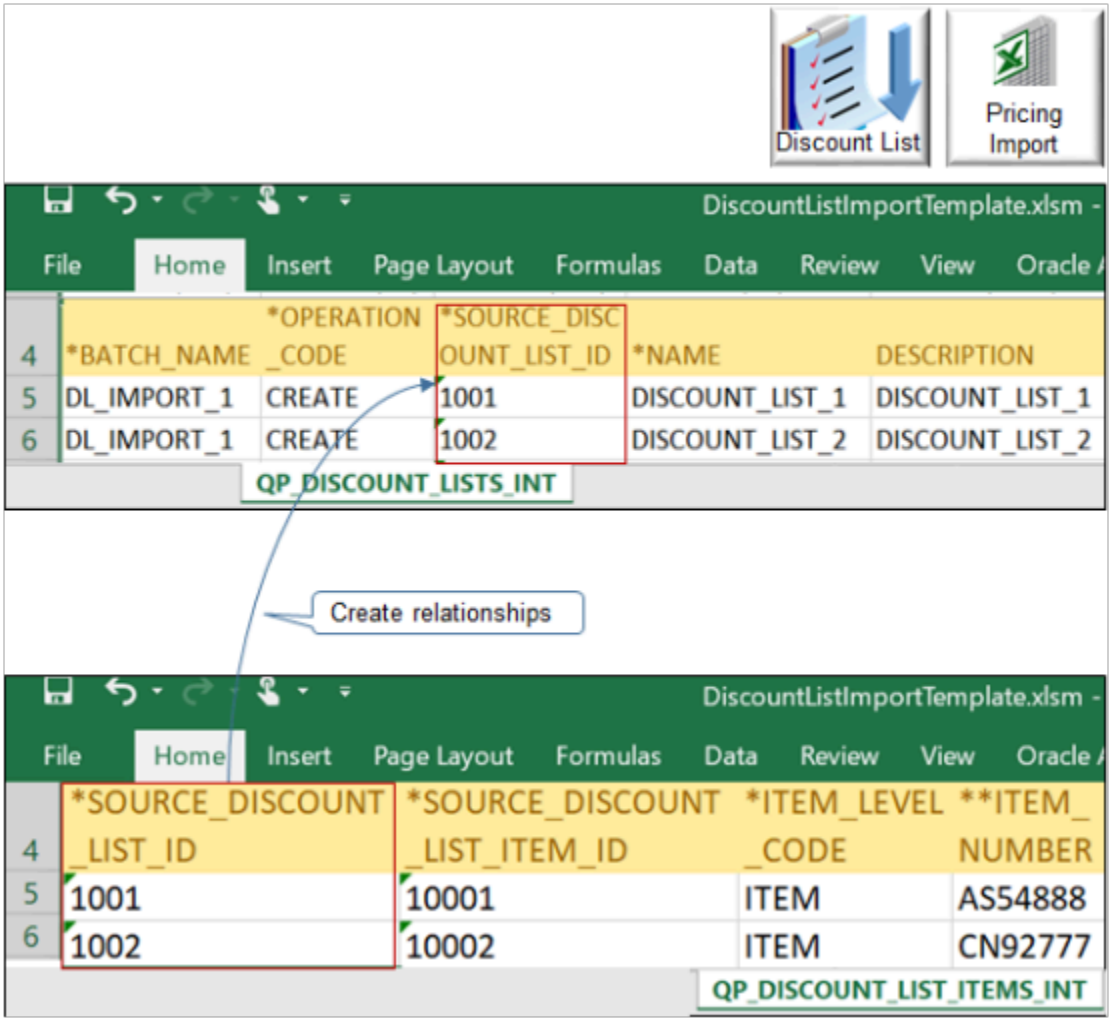
OPERATI CODE	SOURCE_ ROOT_ PARENT_ ID	SOURCE_ PARENT_ ID	SOURCE_ TERM_ ID	NAME	PRICE_ TYPE_ CODE	CHARGE_ TYPE_ CODE	CHARGE_ SUBTYPE CODE	ADJUSTI TYPE_ CODE	ADJUSTI AMOUNT	ADJUSTI BASIS	APPLY_ TO_ ROLLUP_ FLAG	START_ DATE
CREATE	1001	10001	100001	AS54888 SR1	ONE_ TIME	ORA_ SALE	ORA_ PRICE	DISCOUN PERCENT	5	QP_ AdjBasis	N	2019/01/15 10:15:20

You aren't adding a pricing matrix, so leave these worksheets empty:

- QP_MATRIX_DIMENSIONS_INT
- QP_MATRIX_RULES_INT

How to Set the Value for SOURCE_DISCOUNT_LIST_ID

Create a relationship:



Note

- Use SOURCE_DISCOUNT_LIST_ID to create a relationship between the parent discount list and other child entities. For example:
  - Set SOURCE_DISCOUNT_LIST_ID to 1001 on the parent QP_DISCOUNT_LISTS_INT sheet.
  - Set SOURCE_DISCOUNT_LIST_ID to 1001 on the child QP_DISCOUNT_LIST_ITEMS_INT sheet.

A discount list contains entities, such as items. The worksheet uses this reference to create a relationship between each child entity and the parent discount list.

The example includes items AS54888 and CN92777. Each of them reference list 1001. You can use 1001 or whatever numeric value you prefer. It doesn't have to be 1001, but you must use the same number across worksheets. For example, if you use 1001 on QP_DISCOUNT_LISTS_INT, then you must use 1001 on QP_DISCOUNT_LIST_ITEMS_INT.

Upload Your Data

1. Click the **Instructions and CSV Generation** tab, then click **Generate CSV File**.
2. In the dialog that displays, save the file as DiscountListInterface.zip.
3. Sign into Oracle Applications. Make sure you have the Import Price Lists privilege or the Import Approved Price Lists privilege.

4. Go to the Scheduled Processes work area, then run the Load Interface File for Import scheduled process to upload the zip file:

Parameter	Value
Import Process	Import Discount Lists
Data File	Click **Upload a New File**, browse to the DiscountListInterface.zip file that you saved earlier, then click **OK**.

5. Submit, then notice the process number that the dialog displays, such as 163199.
6. Click **Actions > Refresh** on the Pricing Administration page until you can see that the status for your scheduled process is Succeeded.

Import Your Data

1. Click **Schedule New Process**, then run the *Import Discount Lists* scheduled process:

Parameter	Description
Batch Name	Select the same value that you entered in the Batch Name column of sheet QP_DISCOUNT_LISTS_INT in the import template. For example, DL_IMPORT_1.
Commit Point	Optional. Specify at what point to commit data to the transaction tables while the scheduled process processes your records. If you leave this parameter empty, then the scheduled process will automatically commit data after it processes 1,000 records, but you can specify when to do it. For example, if you specify 1 as the commit point, then the scheduled process will commit data after it processes one record. If you specify 50, then the scheduled process doesn't commit data until after it processes the 50th record. Each discount list is one record. Set the commit point to improve performance. Each commit requires a round trip to the database. Each round trip takes time and consumes computing resources. If you import 2,000 records and you set Commit Point to 1, that's 2,000 round trips, but if you set it to 100, that's only 20 round trips. If you set a high commit point, then make sure your server has sufficient cache. The import uses the cache until the next commit point. For example, if you set Commit Point to 2,000, then the import will load data into the cache until it processes 2,000 records. If you set a high commit point and find that the import fails during testing because of a processing error or resources not available error, then reduce the commit point.
Number of Child Processes	Optional. Specify how many child scheduled processes to run for the items and rules that each discount list contains.

Parameter	Description
	The scheduled process automatically creates the optimal number of child scheduled processes that it needs for each batch, but you can specify how many. For example, if you specify 16, then the scheduled process will create no more than 16 child processes. The child processes run in parallel with each other. The typical range of values is 1 through 16. The scheduled process typically uses 4, by default. Assume you need to process 100 records. If you set Number of Child Processes to: o **1**. One child process will process all 100 records consecutively. It takes a lot longer. o **10**. Ten child processes will process 10 records each, and each of the 10 child processes run concurrently. Its a lot faster, but make sure you have sufficient computing power and the servers that you need to accommodate this parallel processing. Adjust Commit Point and Number of Child Processes during testing until you find the optimal balance between speed and your investment in computing resources.

Batch Name is required. The other parameters are optional.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

2. Submit, then notice the process number that the dialog displays.

The parent *Import Discount Lists* scheduled process creates child scheduled processes:

Child Scheduled Process	Description
Import Discount List Headers and Access Sets	Processes the headers and the access set for each discount list.
Import Discount List Items and Rules	Processes the items, pricing terms, and matrix rules for each discount list.

Use the search result area of the Pricing Administration page to refresh the page and monitor your parent and child processes. Continue to refresh until the status for each process is Succeeded.

3. Click **View Log** to examine the log and verify the records that you imported.

Verify Your Import

1. Go to the Pricing Administration work area, search for, then open your discount list:

Attribute	Value
Name	DISCOUNT_LIST_1

2. Verify that the header attributes match the values from your worksheet:

Attribute	Value
Status	Approved
Currency	USD
Business Unit	Vision Operations

3. Verify the item that you imported for the DISCOUNT_LIST_1 list:
   - On the Discount Lines tab, search for the item:

Attribute	Value
Name	AS54888

   - On the Discount Rules area, verify the rule that you imported:

Attribute	Value
Rule Name	AS54888-SR1
Rule Type	Simple
Rule Start Date	1/15/19 10:10 AM
Rule End Date	Empty
Price Type	One Time
Charge Type	Sale
Charge Subtype	Price
Frequency	Empty
Adjustment Type	Discount Percent

Attribute	Value
Adjustment Basis	QP_AdjBasisforBaseListPrc
Adjustment Amount	5

- Click **Access Sets**, then verify the access set that you imported:

Attribute	Value
Set Code	COMMON
Set Name	Common Set

4. Repeat these steps for the DISCOUNT_LIST_2 discount list.

   Verify that the values for DISCOUNT_LIST_2 are the same as DISCOUNT_LIST_1 but with these differences:
   - Status is In Progress.
   - The item is CN92777.
   - The Discount Rules area doesn't contain a rule.

Troubleshoot

See:

- *Guidelines for Importing Batches of Discount Lists*
- *Troubleshoot Importing Batches of Discount Lists*

Related Topics

- Import Price Lists

Import Batches of Discount Lists That Have Pricing Matrices

Add a pricing matrix to a discount list that already exists.

Assume you must import the DISCOUNT_LIST_1 discount list. It already exists, but you must add pricing terms and a pricing matrix to it. You must add the Size condition and the Color condition. For details, see *Import Batches of Discount Lists*.

Summary of the Setup

1. Add your discount list data.

2. Set up matrix class.
3. Upload and verify data.

Add Your Discount List Data

1. Get a copy of the Excel file for this example:
   - Go to *Technical Reference for Oracle Order Management (Doc ID 2051639.1)*.
   - Download the Payloads and Files attachment.
   - Unzip the attachment and open DiscountListImportTemplate_example_2.xlsm. This file already includes the values you use in this example.
2. Open the DiscountListImportTemplate.xlsm file in Excel.
3. Add the discount list header:
   - Click the QP_DISCOUNT_LISTS_INT tab, then set the values:

Attribute	Value
BATCH_NAME	Import Batch 1
OPERATION_CODE	NO-OP The discount lists already exists in the Pricing Administration work area. You aren't adding it, so you use NO-OP (no operation) to indicate to not do an operation, but you still use this row to identify the discount list where you're adding the pricing matrix.
SOURCE_DISCOUNT_LIST_ID	1001 This value identifies the discount list. You use it to create a relationship between the parent list DISCOUNT_LIST_1 on sheet QP_DISCOUNT_LISTS_INT and other entities on other child sheets. For example, you also use 1001 in the SOURCE_DISCOUNT_LIST_ID column of the child QP_DISCOUNT_LIST_ITEMS_INT sheet.

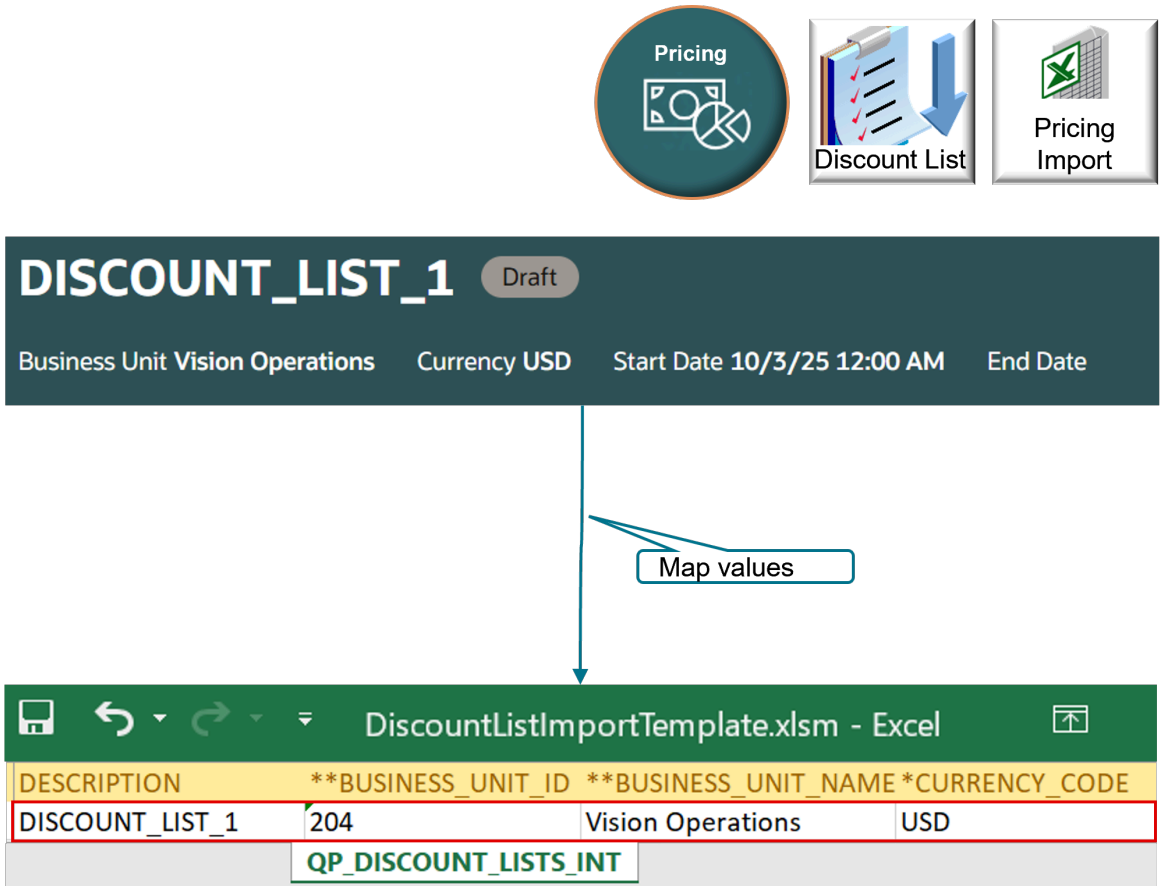
   - Open a web browser then go to **Home > Order Management > Pricing Administration > Discount Lists**.
   - Search for and open the DISCOUNT_LIST_1 discount list.
   - Map values from the discount list's header to the QP_DISCOUNT_LISTS_INT worksheet in Excel:

Attribute	Value in Work Area	Value in QP_DISCOUNT_LISTS_INT
Name	DISCOUNT_LIST_1	Set NAME to DISCOUNT_LIST_1.
Business Unit	Vision Operations	Set BUSINESS_UNIT_NAME to Vision Operations.

Attribute	Value in Work Area	Value in QP_DISCOUNT_LISTS_INT
Currency	USD	Set CURRENCY_CODE to USD.
Start Date	1/15/19 10:10 AM	Set START_DATE to 2019/01/15 10:10:10.
End Date	Empty	Leave END_DATE empty.
Status	Approved	Set STATUS_CODE to Approved.
Description	DISCOUNT_LIST_1	Set DESCRIPTION to DISCOUNT_LIST_1.

Make sure you map the exact value, including upper case and lower case letters. For example, use Vision Operations. Don't use other values, such as Vision operations, vision operations, visionOperations, Vision Ops, and so on.

For example:



4. Ignore the access sets in Excel.

   This discount list doesn't have any access sets other than the common set, so ignore the QP_DISCOUNT_LIST_SETS_INT worksheet.
5. Add your item:
   - Click the QP_DISCOUNT_LIST_ITEMS_INT tab in Excel, then set the values:

Attribute	Value
OPERATION_CODE	NO-OP The item already exists in the Pricing Administration work area. You aren't adding it, so you use NO-OP to indicate to not do an operation. You use this row to identify the item where you're adding the pricing matrix.
SOURCE_DISCOUNT_LIST_ID	1001
SOURCE_DISCOUNT_LIST_ITEM_ID	10001 This value creates a relationship between the parent AS54888 item on the QP_DISCOUNT_LIST_ITEMS_INT worksheet and the: - Child pricing term on worksheet QP_PRICING_TERMS_INT. - Child matrix dimension on worksheet QP_MATRIX_DIMENSIONS_INT.

   - In Pricing Administration, on the on the discount list, on the Discount Lines tab, search for your item:

Attribute	Value
Name	AS54888

   - Map values from the search results in Pricing Administration to the QP_DISCOUNT_LIST_ITEMS_INT worksheet in Excel:

Attribute	Value in Work Area	Column in QP_DISCOUNT_LIST_ITEMS_INT
Item Level	Item	Set ITEM_LEVEL_CODE to ITEM.
Name	AS54888	Set ITEM_NUMBER to AS54888.
-	-	Leave ITEM_ID empty.
Pricing UOM	Each	Set PRICING_UOM to Each.

Attribute	Value in Work Area	Column in QP_DISCOUNT_LIST_ITEMS_INT
-	-	Set PRICING_UOM_CODE to Ea.
Line Type	Buy	Set LINE_TYPE_CODE to ORA_BUY.

Service Duration Period, Service Duration, and Associated Items are empty in Pricing Administration, so leave all other columns in the worksheet empty, such as SERVICE_DURATION_PERIOD, SERVICE_DURATION_PERIOD_CODE, SERVICE_DURATION, ATTRIBUTE_CATEGORY, ATTRIBUTE_CHAR1, and so on.

6. Create a new pricing term:
    - Click the **QP_PRICING_TERMS_INT** tab in Excel, then set the values:

Attribute	Value
OPERATION_CODE	CREATE
SOURCE_ROOT_PARENT_ID	1001 This value creates a relationship to the DISCOUNT_LIST_1 discount list that you specify in column SOURCE_DISCOUNT_LIST_ID on sheet QP_DISCOUNT_LISTS_INT. The discount list is the grandparent, and the AS54888 item is the parent of the pricing term.
SOURCE_PARENT_ID	10001 This value creates a relationship to the parent AS54888 item that you specify in column SOURCE_DISCOUNT_LIST_ITEM_ID on sheet QP_DISCOUNT_LIST_ITEMS_INT.
SOURCE_TERM_ID	100001 The pricing term is a parent of the pricing matrix, so you will use this value in the next step to create a relationship between term and matrix.
NAME	AS54888-AR1
PRICING_RULE_TYPE_CODE	ATTRIBUTE_PRICING
PRICE_TYPE_CODE	ONE_TIME

Attribute	Value
CHARGE_TYPE_CODE	ORA_SALE
CHARGE_SUBTYPE_CODE	ORA_PRICE
APPLY_TO_ROLLUP_FLAG	N
START_DATE	2019/01/15 10:10:10

Leave all other values in the sheet empty.

7. Create new matrix dimensions:
    - Click the **QP_MATRIX_DIMENSIONS_INT** tab, then set the values:

OPERATION_CODE	SOURCE_ROOT_PARENT_ID	SOURCE_PARENT_ID	SOURCE_MATRIX_ID	DIMENSION_NAME	DIMENSION_TYPE	MAP_TO_RULE_COLUMN
CREATE	1001	100001	7001	Color	Condition	VALUE_STRING1
CREATE	1001	100001	7001	Size	Condition	VALUE_STRING2
CREATE	1001	100001	7001	Adjustment Type	Result	VALUE_STRING3
CREATE	1001	100001	7001	Adjustment Amount	Result	VALUE_STRING4
CREATE	1001	100001	7001	Adjustment Basis	Result	VALUE_STRING5

Note

- Use a separate row for each CREATE action.
- SOURCE_ROOT_PARENT_ID maps to the SOURCE_DISCOUNT_LIST_ID column on the QP_DISCOUNT_LISTS_INT sheet. It identifies DISCOUNT_LIST_1 as the root parent of each matrix dimension.

- SOURCE_PARENT_ID maps to the SOURCE_TERM_ID column on the QP_PRICING_TERMS_INT sheet. It identifies AS54888-AR1 as the parent pricing term of each matrix dimension.
- SOURCE_MATRIX_ID is a unique value you add to identify the pricing matrix.
- DIMENSION_NAME and DIMENSION_TYPE must equal the same value you set up in the matrix class. You will verify these later in this topic.
- Use a different value for each row in MAP_TO_RULE_COLUMN. You must use the name of a VALUE_STRING column from sheet QP_MATRIX_RULES_INT. You will see why next.

8. Create new matrix rules:

   - Click the QP_MATRIX_RULES_INT tab, then set the values:

OPERATION_CODE	SOURCE_ROOT_PARENT_ID	SOURCE_MATRIX_ID	SOURCE_RULE_ID
CREATE	1001	7001	1500001
CREATE	1001	7001	1500002

Note

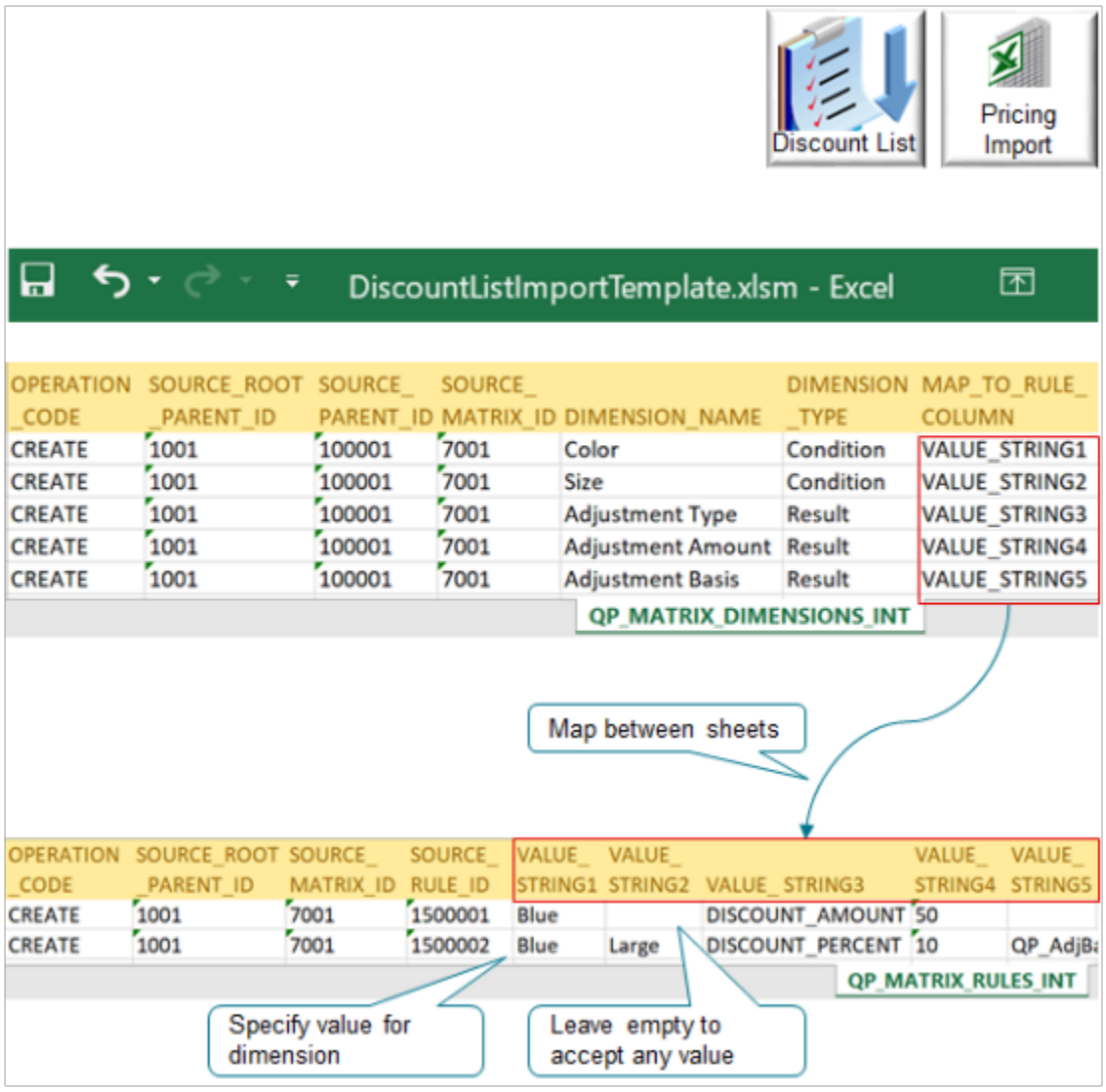
- Use a separate row for each CREATE action.
- SOURCE_ROOT_PARENT_ID maps to the SOURCE_DISCOUNT_LIST_ID column on the QP_DISCOUNT_LISTS_INT sheet. It identifies DISCOUNT_LIST_1 as the root parent of each matrix rule.
- SOURCE_MATRIX_ID maps to the SOURCE_ MATRIX_ID column on the QP_MATRIX_DIMENSIONS_INT sheet. It identifies 7001 as the parent dimension for the matrix rule.
- SOURCE_RULE_ID is a unique value you add to identify the rule.

Here's a continuation of the same rows. You add the value strings:

VALUE_STRING1	VALUE_STRING2	VALUE_STRING3	VALUE_STRING4	VALUE_STRING5
Blue	Leave empty	DISCOUNT_AMOUNT	50	Leave empty
Blue	Large	DISCOUNT_PERCENT	10	QP_ AdjBasisforBaseListPrc

VALUE_STRING1	VALUE_STRING2	VALUE_STRING3	VALUE_STRING4	VALUE_STRING5

Use the value strings columns to create relationships between QP_MATRIX_DIMENSIONS_INT and QP_MATRIX_RULES_INT:



Note

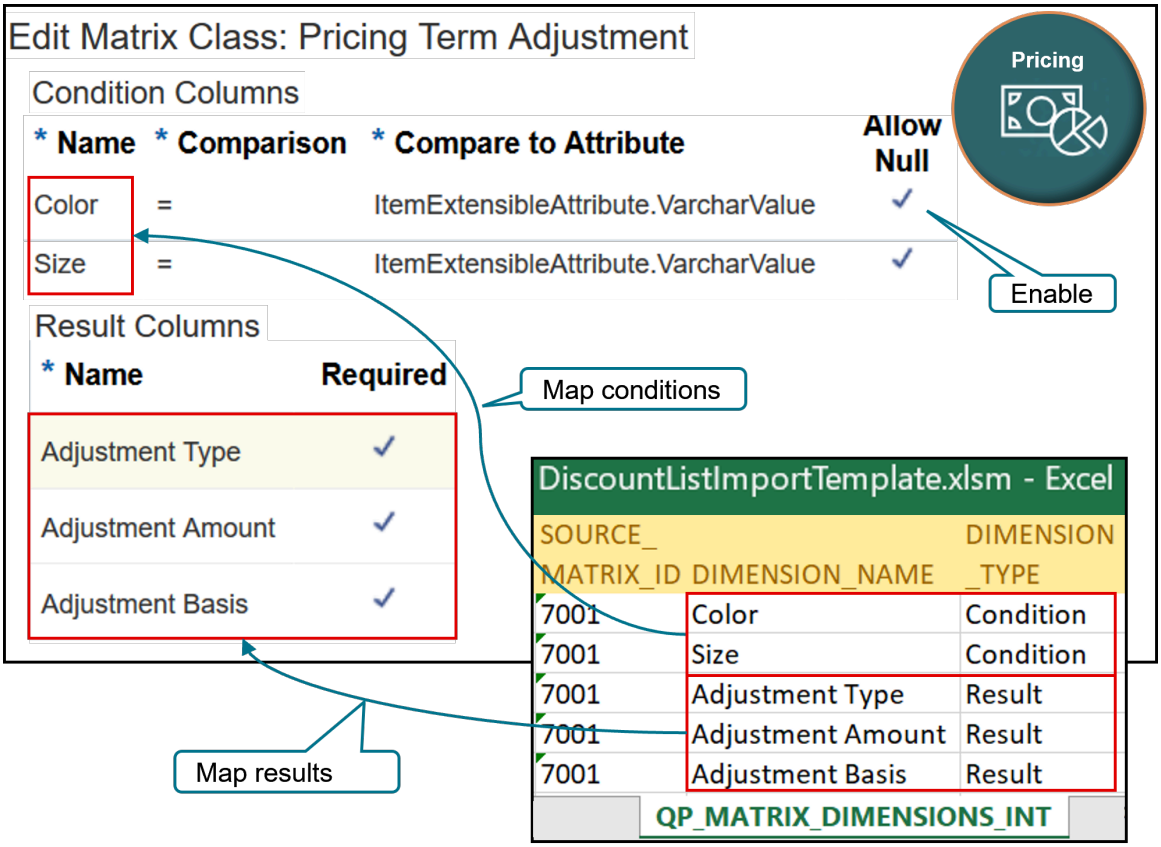
- You map value strings from the QP_MATRIX_DIMENSIONS_INT sheet to the QP_MATRIX_RULES_INT sheet.
- The MAP_TO_RULE_COLUMN on sheet QP_MATRIX_DIMENSIONS_INT identifies the column on sheet QP_MATRIX_RULES_INT.
- For example, VALUE_STRING1 in column MAP_TO_RULE_COLUMN maps to column VALUE_STRING1 on sheet QP_MATRIX_RULES_INT. The dimension on

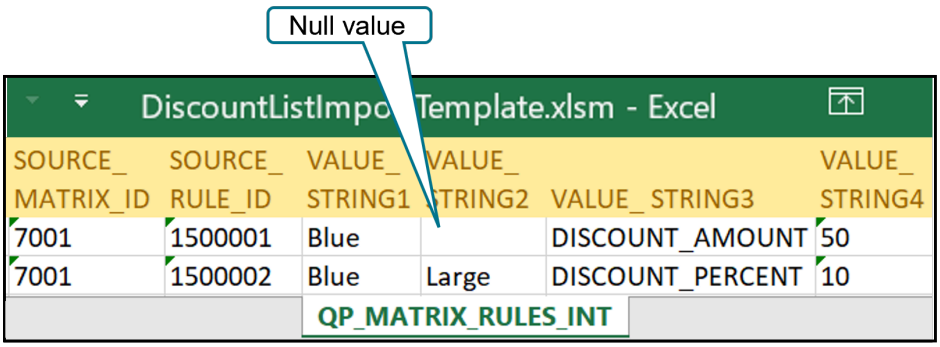
sheet QP_MATRIX_DIMENSIONS_INT is Color, and it maps to VALUE_STRING1 on QP_MATRIX_RULES_INT, which contain Blue.

- The value string column on QP_MATRIX_RULES_INT identifies the value you will be able to select in Pricing Administration. If you leave the value empty in a value string column on QP_MATRIX_RULES_INT, then you can select any value Pricing Administration. If you leave it empty, make sure you enable Allow Null for the matrix class, which you will do later in this topic.
- The value for VALUE_ STRING5 is QP_AdjBasisforBaseListPrc. Its chopped off in the screen print.

Set Up Matrix Class

You must set up the class so it supports the pricing matrix that you're adding to the discount list. You must set it up before you upload your data:





Note

- Add condition columns in the matrix class that support the conditions you add on the QP_MATRIX_DIMENSIONS_INT sheet.
- Add result columns in the matrix class that support the results you add on the QP_MATRIX_DIMENSIONS_INT sheet.
- Enable the Allow Null attribute on the condition in the matrix class so you can use an empty value in the QP_MATRIX_RULES_INT sheet. Use null so you can enter any value when you set up the discount list in the Pricing Administration work area. If you don't use null, then you must select from the values that you specify on the sheet.
- Do this set up only if your import includes conditions and results that are different from the matrix class in the Pricing Administration work area.

For this example, you add your own attributes to the predefined Pricing Term Adjustment matrix class.

Try it:

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, in the Name column, click **Pricing Term Adjustment**.
3. On the Edit Matrix Class page, in the Condition Columns area, add rows:

Name	Comparison	Compare to Attribute	Allow Null
Size	=	itemExtensibleAttribute.VarcharV	Contains a check mark
Color	=	itemExtensibleAttribute.VarcharV	Contains a check mark

   Recall that you added these in the worksheet and specified them as Condition dimension types.

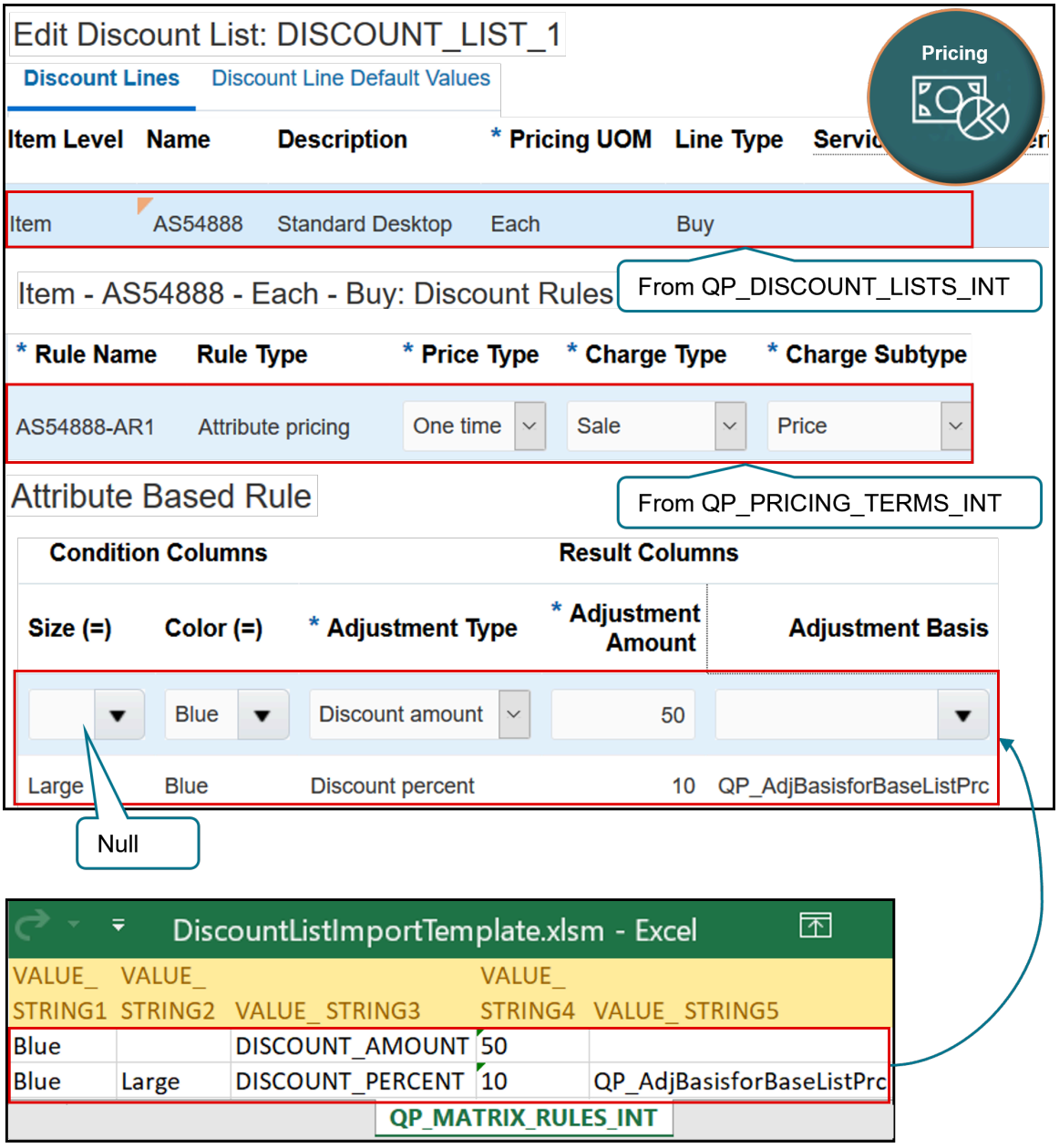
   Learn how to set the compare to attribute and domain in the condition and result. For details, see *Add Your Own Attributes to Items in Pricing*.
4. Verify that the results support your new pricing matrix.

   In the Result Columns area, if these rows don't exist, add them. Recall that you added these in the worksheet and specified them as Result dimension types. Notice that they're all required:

Name	Required	Allow Null
Adjustment Type	Contains a check mark	Empty
Adjustment Amount	Contains a check mark	Empty
Adjustment Basis	Contains a check mark	Contains a check mark

5. Save and close.

Upload and Verify Data



Do it:

1. Create the zip file, upload your data, and run the scheduled processes. For details, see *Import Batches of Discount Lists*.
2. In the Pricing Administration work area, click **Tasks > Manage Discount Lists**.
3. Search for and DISCOUNT_LIST_1 and open it for editing.

4. Verify that the header contains the values you specified on sheet QP_DISCOUNT_LISTS_INT:

Attribute	Value
Name	DISCOUNT_LIST_1
Currency	USD
Business Unit	Vision Operations

5. In the Discount Lines area, search for the item you added from sheet QP_DISCOUNT_LIST_ITEMS_INT:

Attribute	Value
Name	AS54888

6. Verify that the search results contain the values you specified on sheet QP_DISCOUNT_LIST_ITEMS_INT:

Attribute	Value
Pricing UOM	Each
Line Type	Buy

Note that Pricing gets other values for the item from Product Management, such as Standard Desktop in the Description.

7. Expand section **Item AS54888 Each Buy: Discount Rules**, then verify that it contains the values you specified in sheet QP_PRICING_TERMS_INT:

Attribute	Value
Rule Name	AS54888-AR1
Rule Type	Attribute Pricing
Price Type	One Time
Charge Type	Sale
Charge Subtype	Price

Attribute	Value

8. Expand section **Attribute Based Rule**, then verify that it contains the values you specified in sheet QP_MATRIX_RULES_INT:

Size	Color	Adjustment Type	Adjustment Amount	Adjustment Basis
Empty	Blue	Discount Amount	50	Empty
Large	Blue	Discount Percent	10	QP_ AdjBasisforBaseListPrc

Related Topics

- Import Batches of Discount Lists
- Add Your Own Attributes to Items in Pricing

Troubleshoot Importing Batches of Discount Lists

Troubleshoot problems that happen when you import a batch of discount lists.

Trouble	Description
I don't see my batch in the Batch Name attribute on the Process Details dialog when I try to run the *Import Discount Lists* scheduled process.	Different problems might cause this trouble: • You must successfully upload your batch before you run the scheduled process. • If you rerun Import Discount Lists, then you must first rerun the Load Interface File for Import scheduled process before you rerun Import Discount Lists.

I encounter an error in the scheduled process log.
`The discount list entity you attempted to append doesn't exist or is expired.`
`The value of the attribute set_id is not valid.`
	Make sure the attribute contains a value and that your value is the correct data type.
	For example, the data type for SET_ID is Number(18), so make sure it contains numeric data with precision 18. For details, go to *SQL Language Reference*, then expand **Basic Elements of Oracle SQL > Data Types**.

I examine the error logs but they seem incomplete.	Wait for all processes to reach their "terminal" state. Processing continues to add details to the log even after it sets the process status to Error. To determine whether the process is done, click Error in the status column, then verify that the dialog displays this text. `The request has completed and is in a terminal state.`

Trouble	Description
	Also, each process creates two files, one with a log extension and the other with the .txt extension. To view both, click more in the link next to Attachment of the log area in the Scheduled Processes work area.
I encounter an error in the scheduled process log. `1002: A discount list with the name DISCOUNT_LIST_2 already exists. Enter a unique name.`	This error happens when you set the action to CREATE for an entity, such as a discount list, but the entity already exists. Change the name in template, or delete the entity in Pricing Administration, then import again. If you import again with the same template, make sure you remove records that the import successfully processed. The scheduled process imports records sequentially. For example, assume your template includes DISCOUNT_LIST_1 and DISCOUNT_LIST_2, the import successfully imports DISCOUNT_LIST_1 but fails while attempting to import DISCOUNT_LIST_2. You correct the error in the template for DISCOUNT_LIST_2 but don't remove DISCOUNT_LIST_1. You upload and import again, but this time you encounter the duplication error for DISCOUNT_LIST_1. This happens because the first import already imported DISCOUNT_LIST_1 into the database.
I encounter an error in the scheduled process log. `1001: QP_PRCLT_IMPORT_APROVE_ACC_SET`	Try this: • Make sure you specify an access set on the QP_DISCOUNT_LIST_SETS_INT sheet. • Specify an access set for each discount list you create. Troubleshoot depending on the command you use on QP_DISCOUNT_LIST_SETS_INT: • **I used CREATE**. Make sure the access set doesn't already exist. • **I didn't use CREATE**. Make sure the access set does already exist.

Note

- If you delete data in the Pricing Administration work area to fix a duplication problem, it might not be necessary to run the Load Interface File for Import scheduled process again because you haven't changed data in the interface tables. Try running only the Import Discount Lists scheduled process. See *Load Interface File for Import Process*.

Import Cost Lists

Guidelines for Importing Cost Lists

Use file-based data import to import a large number of cost lists.

- Create headers for your cost lists, access sets, items on cost lists, and charges.
- Use a cost list to set up cost plus pricing, or to calculate part of the profit margin for a charge.
- Add access sets, items, and charges to a cost list that already exists.

You might find this feature useful when you maintain the costs that you use to determine price or margin outside of Oracle Pricing. You can import costs into a cost list, then use them to determine cost plus pricing or to calculate margin in the charge breakdown on the order line in Oracle Order Management:

- Efficiently create a large number of cost lists.
- Process all your import data quickly with a single import.
- You can create new cost lists, cost list items, and cost list charges.
- You can't use this feature with configured items, coverages, or subscriptions.

The concepts, guidelines, and procedures that you use to import costs list are similar to the concepts, guidelines, and procedures that you use to import batches of price lists and discount lists. Here are some important differences:

- Make sure you have these privileges:
  - Import Cost Lists (QP_COST_LIST_IMPORT_PRIV). Import cost lists that are in progress.
  - Import Approved Cost Lists (QP_COST_LIST_APPROVED_IMPORT_PRIV). Import cost lists that are approved.
- Download the CostListsImportTemplate.xls file. Go to *File-Based Data Import (FBDI) for Oracle SCM*, Order Management > Cost List.
- Add your cost lists, items, and charges to the template.

For more, see *Import Batches of Discount Lists*.

Click *here* to view a demonstration that illustrates how to use the cost list template. The demonstration starts at 01:52.

Update a Large Number of Cost Lists

You can also use file-based data import to update a large number of cost lists:

- Update headers for your cost lists, access sets, items on cost lists, and charges.
- Use a single action to end date charges on cost list items, and then create new charges.
- View and fix validation errors that happen during import.
- Delete cost list headers, access sets, items, and charges from interface tables after the scheduled process is done running.

Note

- Download and use the Cost Lists Import template.
- The Maximum Number of Child Processes parameter comes with a default value of 20. You can update it, as necessary. For details, see *Manage Pricing Parameters*.
- You can't use this feature with configured items, coverages, or subscriptions.

Export

Export Price Lists

Export your price lists from Oracle Pricing into a spreadsheet so you can migrate, convert, and maintain them.

For example, migrate price lists between a test environment and production environment.

You can export entities from each price list that you specify, including the list header, items on the list, charges for items, pricing tiers, matrix rules, and so on. For more, see:

- *Manage Price Lists*
- *Import Price Lists*

Try It

Assume you need to export the Corporate Segment Price List.

1. Go to the Scheduled Processes work area, search for the Export Price Lists scheduled process, set the Price List Name parameter to Corporate Segment Price List, then submit.
2. Monitor the search results area until the status of your instance is Succeeded.
3. In the search results area, click the **row** that has your process instance.
4. In the Process Details area, where it says Attachment, click **1 More**.
5. In the Attachments dialog, in the File Name column, click the **name** that ends in zip, such as PriceList_export_20220318061320.zip.
6. Save the zip file to your local drive, then unzip the file.
7. Notice that the zip file contains a number of separate CSV files:
    - MatrixDimensionsInterface.csv
    - MatrixRulesInterface.csv
    - PLComponentItemsInterface.csv
    - PLCoveredItemsInterface.csv
    - PriceListChargesInterface.csv
    - PriceListItemsInterface.csv
    - PriceListsInterface.csv
    - PriceListsSetsInterface.csv
    - TierHeadersInterface.csv
    - TierLinesInterface.csv

Note

- PriceListsInterface.csv contains the price list header for the Corporate Segment Price List, such as the list name, its ID, end date, and so on. All the other files contain details about the entity that the Corporate Segment Price List references. For example, PriceListItemsInterface contains all the items on the price list, such as the AS54888 Desktop Computer.
- You can use the data that you get from the Export Price List scheduled process to populate values in an FBDI worksheet (File Based Data Import). If you do, then you must make sure you correctly populate the worksheet. You might need to manually edit the data and add values in the worksheet before you run the Import Price List Batch scheduled process.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

Parameters

Use these parameters to filter the data that the scheduled process will look at:

Parameter	Description
Price List Name	Use the same value that you see in the Name attribute on the Price Lists page in Administering Pricing. For example, Corporate Segment Price List.
Price List Type	Select a value: • Ceiling Price List • Floor Price List • GSA Price List • Segment Price List The scheduled process will export all price lists that are of the type that you select. For example, assume Pricing Administration has two segment price lists, the Corporate Segment Price List and the Computer Service and Rentals price list. If you select Segment Price List, then the process will export these two price lists.
Status	Select a value: • Approved • In Progress The scheduled process will export all price lists that are in the status that you select.

Note

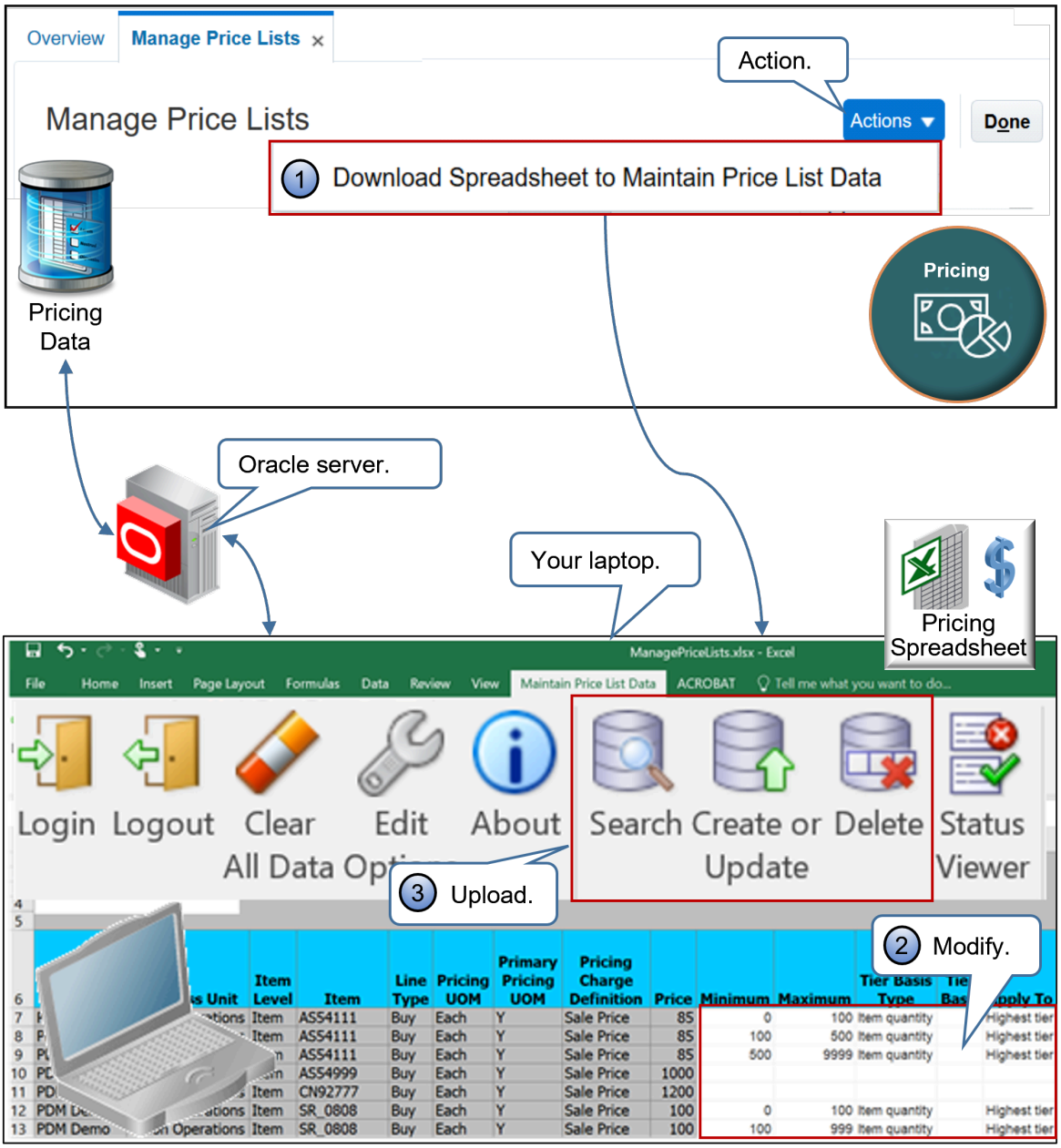
- Each parameter is optional, but you must set a value in at least one parameter. If you don't, you can still submit the process, but it's status will be stuck at Retrying.
- Use these parameters together to filter the price lists that you export. For example, if you set the Price List Type parameter to Segment Price List, and the Status parameter to Approved, then the scheduled process will only export segment price lists that are approved.

For important details, see *Guidelines for Using Scheduled Processes in Order Management*.

Remote

Overview of Using Spreadsheets to Manage Pricing

Search and download pricing data from Pricing Administration, then manage it in a spreadsheet in Microsoft Excel.



Use a Microsoft Excel workbook to modify price lists, discount lists, or customer pricing profiles to reflect changes to your pricing policies and profitability targets. Apply these changes to more than one object, such as all of your price lists, in Pricing Administration:

1. Download a spreadsheet from the Pricing Administration work area.
2. Use the worksheet to do search, download, create, read, update, and delete actions on pricing data in Excel.
3. Upload your changes to the Oracle server.

   The server displays your changes in the Pricing Administration work area.

Use the workbook to:

- Manage price list data, including items, charges, tier adjustments, and matrix adjustments.
- Manage pricing data across more than one price list and more than one business unit.

- Do an operation on items, service items, coverage items, or configured items, or do an operation on all items.
- These pricing spreadsheets use Application Development Framework Desktop Integration (ADFDI). You can use them to manage a low to moderate amount of data. If you have a large amount of data or if you must do a massive update, don't use ADFDI. Use File-Based Data Import instead.
- Manage lists and profiles:

Manage	Description
Manage Price Lists and Discount Lists	○ Download charges or discounts from the Pricing Administration work area to an Excel workbook. ○ Create and update base list price and list price across more than one price list in Excel. ○ Create and update discounts across more than one discount list in Excel. ○ Search for and adjust pricing charges or discounts according to attribute, such as Price List Description or Discount List Description, Pricing Strategy, Business Unit, Currency, Status, Item Number, Description, Charge Definition, Price Type, Effectivity Date, and so on. ○ Upload your changes to the Oracle server.
Manage Customer Pricing Profiles	○ Download profiles from Pricing to an Excel workbook. ○ Edit or create new profiles in Excel. ○ Search for and adjust profiles according to attribute, such as Customer Name, Revenue Potential, Cost to Serve, Customer Value, Customer Rating, Customer Size, Effectivity Date, and so on. ○ Upload your changes.

Get Started

1. Go to **Home > Tools > Download Desktop Integration** to install the ADF Desktop Integration.
2. Go to the Pricing Administration work area, navigate to the page that you need, then download the workbook. You use a different workbook for each of:
   - Price list
   - Discount list
   - Customer pricing profile

Related Topics

- Use Spreadsheets to Manage Pricing
- Manage Discount Lists
- Pricing Rules
- Manage Price Lists
- Guidelines for Using Pricing Spreadsheets

Guidelines for Using Pricing Spreadsheets

Follow these guidelines when you use spreadsheets to manage pricing.

Manage Your Workbook and Search Your Data

- Download the workbook only one time. For subsequent uses, open the `xls` file directly on your local computer.
- The workbook that you download doesn't contain price lists, discount lists, or customer pricing profiles. Its an empty workbook that provides an alternative to using the Pricing Administration work area to administer changes across more than one object.
- Save and move the workbook file to any location on your laptop, or copy it to another computer.
- Don't use the same workbook in different environments. Each workbook contains address details, such as the URL, so it can connect to a specific environment.
- Don't open more than one workbook at the same time. For example, if you must manage price lists and discount lists, then open the workbook for price lists, do all your price list tasks, and close the workbook. Then open the workbook for discount lists. Opening more than one workbook might cause problems in the cache, clipboard, and other areas.
- Be aware of session time out settings. For example, if your session times out after 15 minutes and you're still working in the workbook, you might not be able interact with the server. Commands to the server won't respond. Close the workbook, open the workbook, and sign in.
- If you get stuck, clear the data and start over.
- Enable the Active Charges option to search for all charges that are active according to date.
- If you have more than one price list that have similar names, then make sure you search for the entire name. Assume you have a price list named Vision Price List A and you need to update it. Assume you also have another price list named Vision Price List B, and each list contains 1,000 records. If you search for the string Vision Price List, then the spreadsheet will download all 2,000 records, and any updates that you make will update all the records for lists A and B. To avoid this problem, search for Vision Price List A.
- Keep your Desktop Integration add-in up to date. Oracle periodically modifies the add-in. Download the latest add-in with each update.
- If you encounter an error, fix the error and then retry your upload. Don't upload again without fixing the error first.
- Use the Active Charges option to filter your search results. This will help to improve performance and focus your update so it only considers the charges that you're currently using.
- Enter the entire name of the price list when you search. For example, if you have 10 price list names that all start with Price List, but you only want the Price List for Vision Operations price list, then enter Price List for Vision Operations. Don't enter only Price List. This will help to avoid downloading a bunch of price lists that you don't want.
- Each workbook contains address details, such as the URL, so it can connect to a specific environment. If you use more than one development environment, then don't use the same workbook in these environments. Instead, download a separate workbook into each environment.

Manage Your Data

- • If you must process more than 200 records, then don't use ADF Desktop Integration. Use PriceListsImportBatchTemplate.xlsm instead. For details, see *Import Batches of Price Lists*
- To make sure the data in the workbook is up to date with the data on the server, refresh the search each time after the workbook finishes processing your action.
- Keep your ADF Desktop Integration plug-in up to date. The workbook usually informs you when an update is available. If the plug-in isn't up to date, then your results won't be correct.

- Use ADF Desktop Integration to maintain your data and to do mass updates. Use the Price List Import Template to do a one time import for one price list. For details, see *Import Price Lists*.

Use Microsoft Excel

- Rearrange, resize, or hide columns to meet your preferences.
- Don't delete any column. Don't copy and paste any column. If you delete a column, or copy and paste a column, then you might encounter an error.
- Only update attributes that display with a white background. Don't update attributes that display with a grey background. For example, here are the attributes that you can update on the Manage Pricing Charges worksheet:
    - Calculation Method
    - Price
    - Start Date
    - End Date
    - Allow Manual Adjustment
    - Calculation Type
    - Cost Calculation Amount

  Don't update any other attributes on the Manage Pricing Charges worksheet.
- To create a new record, add data to an empty row. Add values to attributes that display with a white background or a grey background when you create a new record.
- Copy data from another worksheet and paste it into the Oracle worksheet that you download from the Pricing Administration work area.

  For example, assume you maintain your own spreadsheet that includes data about the items that you sell. Copy this data into the Oracle worksheet, then upload it to the Oracle server:
    - Make sure you copy only data rows.
    - Don't copy header rows.
    - Don't overwrite header rows in the Oracle worksheet.
    - Don't import an entire file or a very large set of data. Use file-based data import instead. For details, see *Import Batches of Price Lists*.
- If Excel prompts you to save the file, then don't save changes in the spreadsheet because this data might become old compared to the data on the server. Instead, download data from the server immediately before each action.
- Don't use CTRL+S, and don't use Save in the File menu in Excel.
- You can't use the Undo command in Excel to undo changes while you're in Connected mode.
- You can't select a row and then click Delete to delete it while in Connected mode. Instead, use the Delete command on the ribbon.
- Don't use the Clear or Clear All command in Excel. Instead, use the Clear All Data command that displays in the ribbon after you install ADF Desktop Integration.
- You must use Microsoft Excel. You can't use any other spreadsheet application.

Use Spreadsheet Elements

Spreadsheet Element	Description
Login Logout	Sign into or sign out of a session with the Oracle server.
Clear All Data	Click Clear All Data at the end of each session to clear data from the spreadsheet. This action helps to make sure the spreadsheet is up to date with data on the server during the next session.
Edit Options	This value references the server address. You can modify it to use the spreadsheet with a different server. In most situations, don't modify the value in the Edit Options dialog. Instead, use a different spreadsheet for each server.
Search	Search data on the Oracle server.
Create or Update	Upload data that you created or updated in the Oracle spreadsheet. You upload to the Oracle server.
Delete	Select one or more data rows in the spreadsheet, then click Delete to delete them from the server. This action happens immediately while in Connected mode and you can't undo it.
Status Viewer	View messages that describe the result of your actions. For example, if the Status Viewer displays **`No error`**, then your action was successful. If an error happens, for example you don't include a required attribute, then the Status Viewer displays an error message, such as. **`The attribute Adjustment Type is required.`**
Changed column	Displays an icon that indicates you modified a value in the row but haven't yet uploaded your modification to the server.
Flagged column	Double-click in the Flagged column to delete the row. The Flagged column displays an icon that indicates you plan to delete the row but haven't yet uploaded your deletion to the server. Click **Delete** in the ribbon to upload your deletion.
Status column	Displays the status of your action. Scan the Status column after each action to make sure the server successfully processed your request. If the Status is empty after a request or displays Row Updated Successfully, then the update was successful. If an error happens, then the Status column displays a summary of the error. Use the Status Viewer to view error details.
Key column	Displays a value that uniquely identifies the record. You can't modify this value. Make sure you don't select it when you copy a row. You can hide the Key column.

Use the Manage Pricing Charges Worksheet

- Create or update items and pricing charges.
- You must create a price list and associate it with a business unit before you set up pricing and charges for the item.
- Update charge attributes Price, Start Date, End Date, or Allow Manual Adjustment.
- You can't delete a pricing charge. Instead, use the End Date to set the date when Pricing must no longer use the pricing charge.
- The Updated Pricing Rules area of the Pricing Administration work area doesn't display changes that you make in the spreadsheet.

Use the Manage Tiered Adjustment Rules Worksheet

- Create, update, or delete tier rules for a pricing charge.
- This worksheet processes all tier lines for each charge together in a group. It doesn't process each row individually.
- You can update all tier header attributes except for Tier Basis Type, Application Method, and Enforce Adjustment. For details about the tier header, see *Add Tiers to Pricing Rules*.
- Update all tier line attributes.
- Update the minimum value and the maximum value for tier line attributes only in Connected mode.
- If you use this worksheet to delete the last tier rule of any charge, then this worksheet will also delete the tier header.
- You must set up the charge for the item in a price list before you manage the tiered adjustment rules that reference the item.
- Make sure the discount rule attributes that your rule specifies are identical for each row in the tier group.
- Make sure the tier header includes all required values.
- If you set the tier basis type to an amount, then make sure you set the application method to Per Unit.

Use the Manage Matrix Adjustment Rules Worksheet

- Create, update, or delete matrix rules or attribute rules for a pricing charge.
- You must set up the charge for the item in a price list before you manage a matrix adjustment rule that references the item.
- Do update or delete operations only in Connected mode.
- You must use the `yyyy-mm-dd` format for date data.
- You must use the `yyyy-mm-dd hh:mm:ss` format for time stamp data.
- If you must add a condition column or a result column, then you must first add it to the matrix class in Pricing Administration, then download the workbook from Pricing Administration.

Use Connected Mode or Deferred Mode

The spreadsheet works in Connected mode or Deferred mode, but these modes are transparent to you. You don't need to take an action to use one or the other. For example, if you download data to the spreadsheet from the server, edit a row, then upload the edit, then the spreadsheet works in Connected mode. It maintains an active connection to the server.

An active connection doesn't exist in Deferred mode. For example, assume you open your spreadsheet but don't connect to the server. You intentionally cancel out of the sign in dialog that displays. Instead, you copy data into the Oracle spreadsheet from some other source, such as your own spreadsheet file. You sign into the server and are now in Connected mode. You click Create or Update to upload your changes.

You can create or update data without having to identify the action as a create or update. The spreadsheet automatically determines to update or create during upload whether you use Connected mode or Deferred mode:

- **Update**. It finds a record that matches the data you provide, so it updates the existing record.
- **Create**. It can't find a match given the data that you provide, so it creates a new record. For example, assume you add a new row for a pricing charge for item AS54888 in the Corporate Segment Price List. The spreadsheet examines the Oracle database during the upload. It finds item AS54888 in Product Management but doesn't find AS54888 in the Corporate Segment Price List in Pricing Administration, so it adds AS54888 to Corporate Segment Price List.

Troubleshoot

- If an error happens, then the Status column displays a summary of the error status. Use it to read the summary.
- Use the Status Viewer to read the error message.
- If an error happens, then the spreadsheet displays the Download dialog. Notice the text `Do you want to discard the pending changes?`. Click a value:
  - **Yes**. Discard the change that causes the error. The server doesn't update data and the spreadsheet doesn't display details about the error.
  - **No**. The server doesn't update data, but the spreadsheet does display details about the error in the Status column and in the Status Viewer.
- Use the Search command to restore the values in your spreadsheet after you encounter an error.

Here are some more troubleshooting details:

Trouble	Shoot
You receive a message. `Unable to execute the command Create or Update while a cell is in edit mode.`	Step out of the cell you're editing, then retry the update.
You receive a message. `Maximum: cannot convert the input value to the expected data type (BigDecimal).` Some messages in the Status Viewer includes details: • A prefix that identifies the attribute name. In this example, the error happens in the Maximum attribute. • A parenthetical that suggests a correction. In this example, (BigDecimal) is the parenthetical, and	Make sure the value in the Maximum attribute contains a BigDecimal data type.

Trouble	Shoot
it suggests that you use BigDecimal as the data type value for the Maximum attribute.	
You receive a message. `Required property UniqueAttribute missing.`	Make sure each required attribute contains a value.
You receive a message. `View row with key Oracle x is not found.`	Refresh your data. You might encounter this error if you perform an action, don't refresh your data, then do another action. For example, assume you. 1. Download data to the Manage Tiered Adjustment Rules worksheet. 2. Download data to the Manage Pricing Charges worksheet. 3. Make, then upload a change on the Manage Pricing Charges worksheet. 4. Navigate back to the Manage Tiered Adjustment Rules worksheet but don't refresh your data. 5. Make, then upload a change on the Manage Tiered Adjustment Rules worksheet. To fix this problem, refresh your data after you navigate back to the Manage Tiered Adjustment Rules worksheet, then make and upload your changes.
You receive a message. `A tiered pricing rule with overlapping values already exists for the value range 50 - 500.` You specified a value in the Minimum attribute or Maximum attribute on the Manage Tiered Adjustment Rules worksheet.	Make sure your values don't overlap across tiers. For example, if the values for tier one are 0 through 99, then make sure the minimum value for tier two is 100, not 99 or less.
You receive a message. `Pricing could not finish the upload because the item does not exist or is not valid.`	If you add an item in the spreadsheet, and if you receive an error that the item doesn't exist, then you must make sure Product Management contains the item. For example, assume you add a new row on the Manage Pricing Charges worksheet and specify a value of AS54999 in the Item column. If Product Management specifies the AS54998 item but not the AS54999 item, then the item doesn't exist.
The workbook doesn't display the Maintain Price List Data tab. If you close the workbook, open it, then click No in the Connect dialog, then the workbook won't display the Maintain Price List Data tab.	Click Yes in the Connect dialog.

Related Topics

- Use Spreadsheets to Manage Pricing
- Manage Discount Lists
- Pricing Rules
- Manage Price Lists

Use Spreadsheets to Manage Pricing

Use a Microsoft Excel workbook to manage Oracle Pricing.

Summary of the Steps

1. Install the ADF Desktop Integration.
2. Download your workbook.
3. Search and download your data.
4. Modify and upload your data.
5. End your session.

Install the ADF Desktop Integration

1. Close all instances of Microsoft Excel on your local computer.
2. Sign into Oracle Pricing with administrative privileges.
3. Go to the Home page, then under Tools, click **Download Desktop Integration**.
4. In the status bar at the bottom of your browser, allow the download, then wait for the adfdi-excel-addin-installer.exe file to finish downloading. ADFDI means Application Development Framework Desktop Integration.
5. Open adfdi-excel-addin-installer.exe on your local computer.
6. In the ADF Desktop Integration Installer dialog, click **Developer Options**.
7. In the Developer Options dialog, click **Enabled**, click **Install**, wait for the installation to finish, then click **Close**.
8. Open the Control Panel on your local computer, click **Programs and Features**, then verify the list that displays includes your new installation.

   For example, make sure it displays `Oracle ADF 11g Desktop Integration Add-In for Excel`.
9. Open Excel, then, in `Microsoft Office Customization Installer`, click **Install**.
10. Click **File > Add-Ins**, then verify that the Add-Ins submenu contains ADF Desktop Integration.

    In some Excel versions, you might need to enable the add-in. For details about installing add-ins, see the documentation for Microsoft Excel at https://www.microsoft.com.
11. Close Excel.

Download Your Workbook

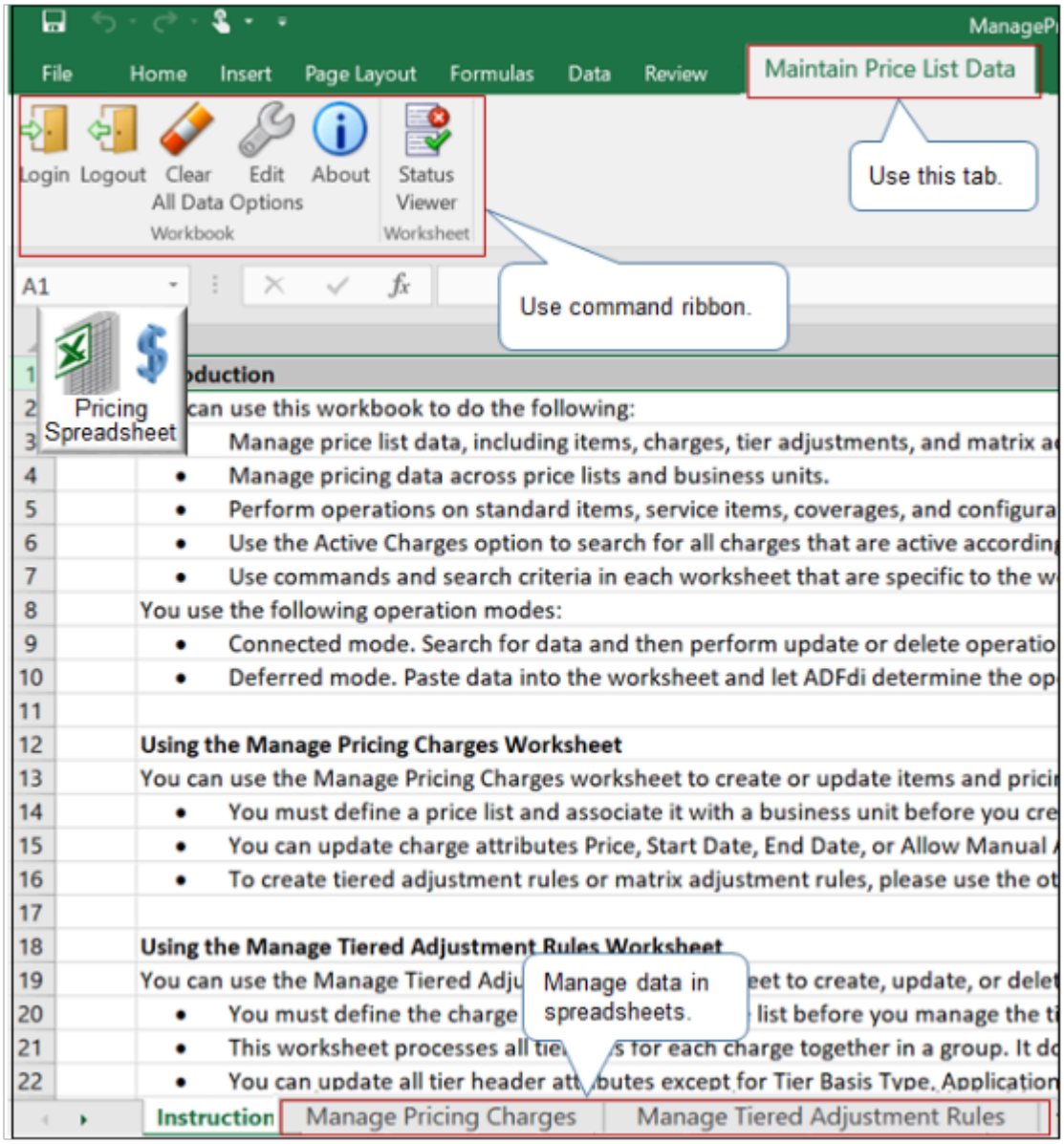
1. Go to the classic Pricing Administration work area.
2. Use one of these tasks, depending on the data that you must download:
   - Manage Price Lists
   - Manage Discount Lists
   - Manage Customer Pricing Profiles

   For this example, navigate to Manage Price Lists.
3. Click **Actions > Download Spreadsheet to Maintain Price List Data**.
4. In the Opening ManagePriceLists.xlsx dialog, set **Open With** to Microsoft Excel, then click **OK**.
5. In Excel, in the Connect dialog, click **Yes**.

   Use this dialog to connect to the server that hosts your Oracle application.
6. In the Login dialog, enter your user name and password, then click **Sign In**.

Use the same user name and password that you use when you sign into Oracle Pricing with administrative privileges.

Your workbook opens:



Note

- Use the Maintain Price List Data tab.
- Use the command ribbon that ADF Desktop Integration installed, such as `Login` and `Logout`, to interact with the server.
- Use worksheets to manage your data, such as Manage Pricing Charges.
- Download the workbook only one time.

Item Class

If you need to use the Item Class attribute in your spreadsheet, then you must enable a profile option before you download the spreadsheet. This option might affect performance. Enable it only if you need it:

1. Go to **Home > My Enterprise > Setup and Maintenance > Manage Profile Options**, then create this profile option:

Attribute	Value
Profile Option Code	QP_ITEM_SEARCH_ENABLED_ADFDI
Profile Display Name	QP_ITEM_SEARCH_ENABLED_ADFDI
Application	Pricing
Module	Pricing Administration Use advanced search. Search for **`User Module Name contains Pricing Administration`**, then select the row in the search results.
Site	Enabled
User	Enabled

2. Go to the Manage Administrator Profile Values task, search the Profile Option Code attribute for QP_ITEM_SEARCH_ENABLED_ADFDI, then add a new profile value:

Attribute	Value
Profile Level	Site
Profile Value	Y

Search and Download Your Data

Assume you're introducing a new line of computer items that use the AS prefix, and you must discontinue the old AS line. You will set the End Date for all AS items:

1. Click the **Manage Pricing Charges** tab, then wait a moment for the worksheet to display column headings:
    - It might take a few seconds for the worksheet to display column headings the first time you open a spreadsheet, depending on network speed and other factors.
    - Each column in the spreadsheet displays an attribute. These are the same attributes you can access On the Price Lists page in the Pricing Administration work area.
2. In the ribbon, click **Search**.
3. In the Search Pricing Charges dialog, set the values, click **OK**, then wait for the spreadsheet to download and display your items:

Attribute	Value
Price List	Corporate Segment Price List

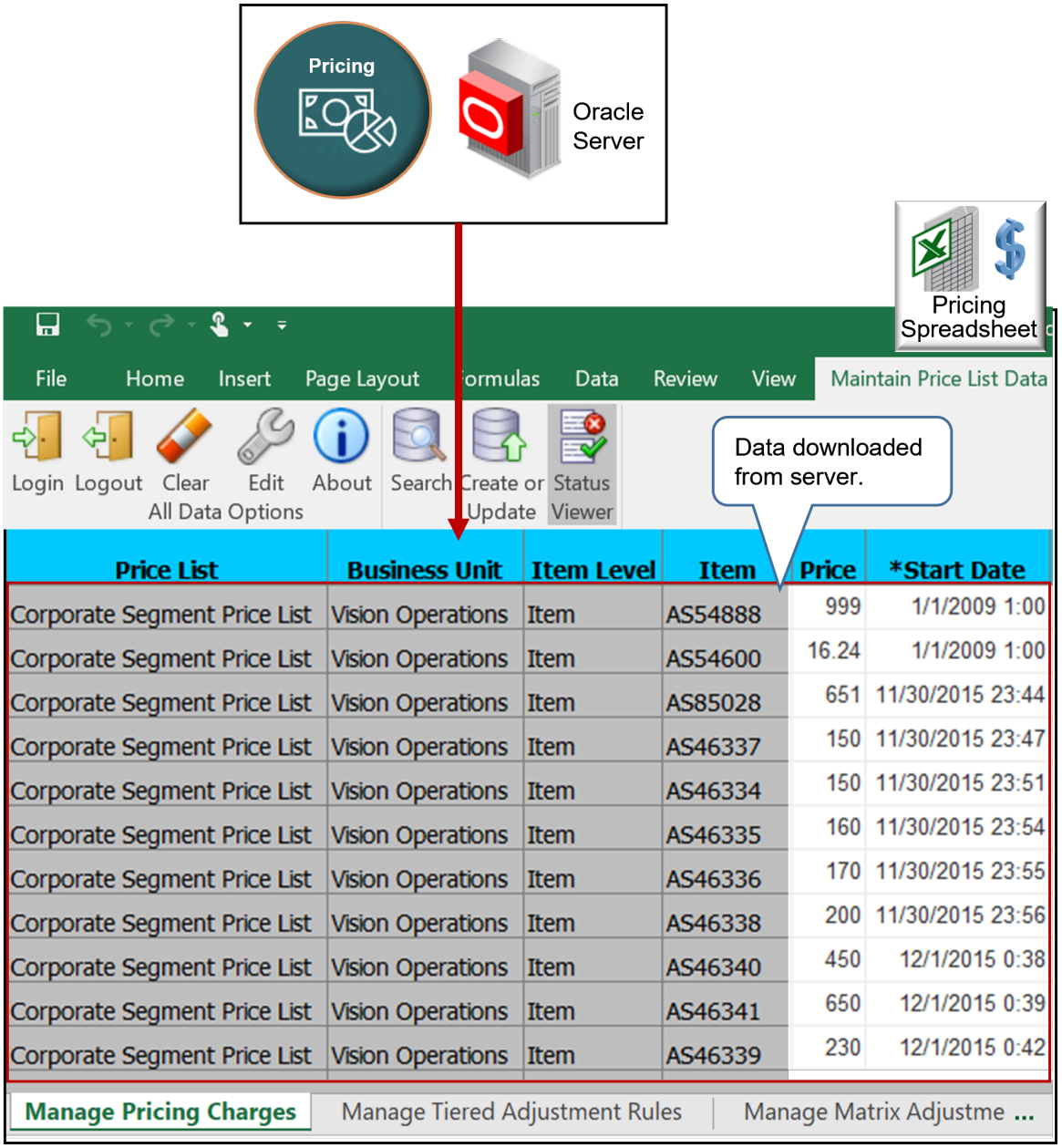
Attribute	Value
Business Unit	Vision Operations
Item	AS

Note

- You must include a value in at least one of the attributes that the dialog displays with a double asterisk (**).
- To improve performance, search for, filter, and download only the data you need for the subsequent action. For example, specify the Price List, Business Unit, and the beginning characters of the Item, instead of only the Business Unit.

4. Wait for the spreadsheet to download data from the server.

   The spreadsheet downloads data that meets your search criteria:



Modify and Upload Your Data

1. Change the value in the End Date column to the current date, such as 12/12/2017 9:00:00 AM, for all rows.

   You can copy and paste values just like you do in any Excel spreadsheet.
2. In the ribbon, click **Create or Update**.

3. Verify your upload:
    - In the Pricing Administration work area, On the Price Lists page, search for Corporate Segment Price List.
    - In the search results, click **Corporate Segment Price List**.
    - On the Price List page, in the Search area, search for the value:

Attribute	Value
Item	AS

    - In the search results, click the first **row**, and then in the Dates area, verify that the End Date contains the date you set in the spreadsheet, such as 12/12/2017 9:00:00 AM.
      The hour might display a different value, depending on the time zone that the server uses.
      For example, the Pacific time zone is one hour behind the Mountain time zone. If your local computer is in the Pacific time zone, and if you set the date to 12/12/2017 9:00:00 AM in the spreadsheet, and if the server uses the Mountain time zone, then the server might display 12/12/2017 10:00:00 AM in the Pricing Administration work area.

End Your Session

1. In Excel, click **Clear All Data**.
2. Click **File > Close**.
3. In the Microsoft Excel dialog that displays the text `Want to save changes?`, click **Don't Save**.
4. To start another session, in Excel, click **File > Open**, then click **ManagePriceLists.xlsx**.

Related Topics

- Manage Discount Lists
- Pricing Rules
- Manage Price Lists
- Overview of Using Spreadsheets to Manage Pricing
- Add Tiers to Pricing Rules

Examples of Using Spreadsheets to Manage Pricing

Examine some examples to learn how you can use spreadsheets to manage pricing.

Manage Tiered Adjustment Rules

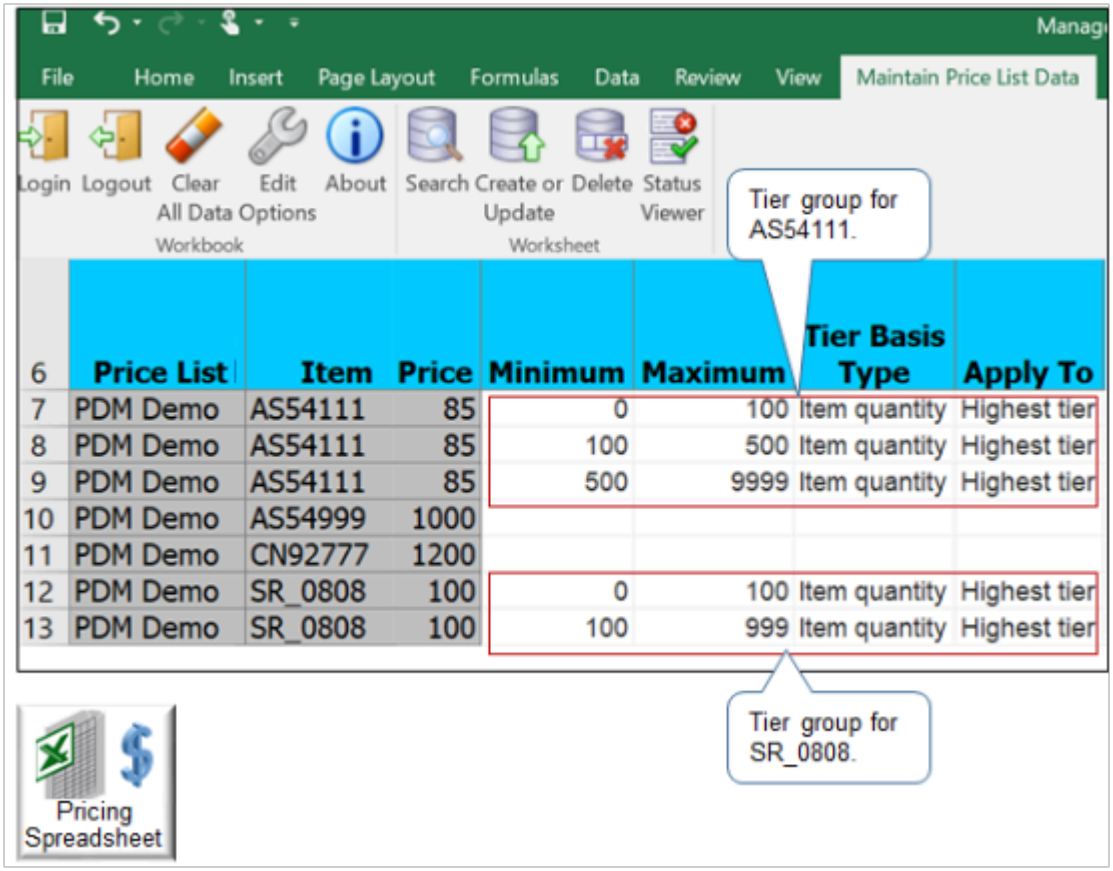
In this example, assume you used the Pricing Administration work area to create a price list named PDM Demo, you added seven items to it, and you use this price list for items that require tier pricing. You will modify tier pricing for item AS54111. For details, see *Tier Pricing*:

1. Open the ManagePriceLists.xlsx file that you downloaded earlier in this topic.
2. In the Connect dialog, click **Yes**, then, in the Login dialog, enter your user name and password.

3. Click the **Manage Tiered Adjustment Rules** tab, then click **Search**.
4. In the Search Tiered Adjustment Rules dialog, set the value, then click **OK**:

Attribute	Value
Price List	PDM Demo

5. Wait for the download to finish, then notice the results that the spreadsheet displays:



	Price List	Item	Price	Minimum	Maximum	Tier Basis Type	Apply To
7	PDM Demo	AS54111	85	0	100	Item quantity	Highest tier
8	PDM Demo	AS54111	85	100	500	Item quantity	Highest tier
9	PDM Demo	AS54111	85	500	9999	Item quantity	Highest tier
10	PDM Demo	AS54999	1000				
11	PDM Demo	CN92777	1200				
12	PDM Demo	SR_0808	100	0	100	Item quantity	Highest tier
13	PDM Demo	SR_0808	100	100	999	Item quantity	Highest tier

Note

- The price list defines two tier groups. One tier group for item AS54111, and one tier group for item SR_0808.
- The spreadsheet includes attributes for tiered pricing, such as Minimum, Maximum, Tier Basis Type, and Apply To. These are the same attributes you set for tier pricing in the Pricing Administration work area.
- The spreadsheet applies the same rules and constraints that the Pricing Administration work area applies for tiered pricing.

6. Modify the Minimum and Maximum values for item AS54111:

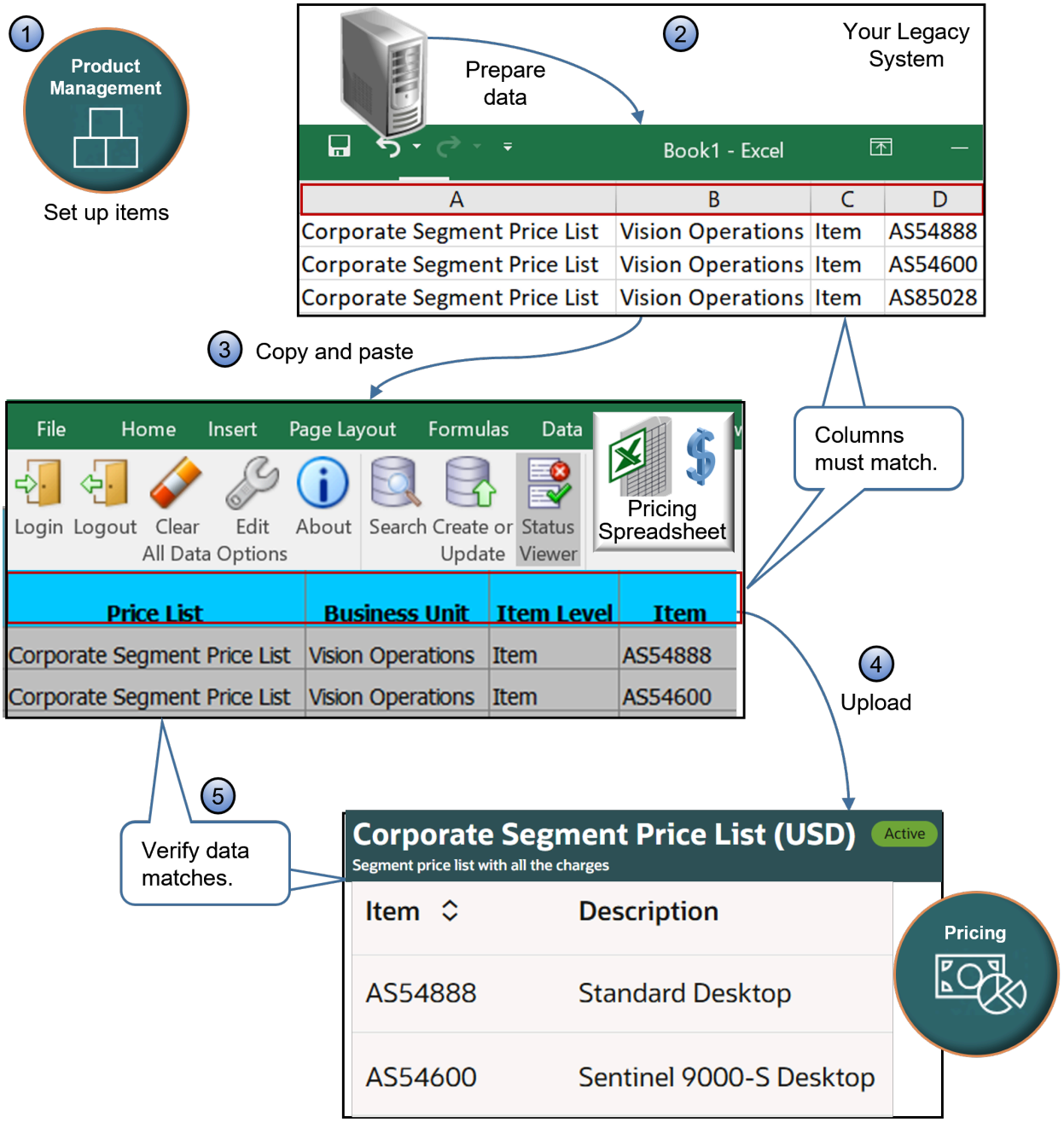
Item	Price	Minimum	Maximum
AS54111	85	0	99

Item	Price	Minimum	Maximum
AS54111	85	100	499
AS54111	85	500	9999

7. Set the adjustment amount and basis for each tier.
8. Upload your changes, navigate to the Pricing Administration work area, then verify that the work area displays your changes.

Migrate Price Lists from Your Legacy System

Assume you must migrate price list headers, items, charges, tier adjustments, and attribute adjustments:



Note

1. Use the Product Management work area to set up each item you will migrate, and assign the item to the validation organization for the business unit.

2. Prepare your data:
    - Use a spreadsheet application to prepare the price list data you're migrating. Make sure the data in your source file matches exactly the columns and sequence of columns that the Manage Pricing Charges spreadsheet uses. For example:

	Column 1	Column 2	Column 3	Column 4
Columns in Source File	Price List	Business Unit	Item Level	Item
Columns in Pricing Spreadsheet	Price List	Business Unit	Item Level	Item

    - Make sure the price list header exists in the Pricing Administration work area. For example, if you migrate data for the Corporate Segment Price List, then make sure at least the header for the Corporate Segment Price List exists in Pricing Administration.
3. Migrate data:
    - Open your pricing workbook and navigate to the Manage Pricing Charges spreadsheet.
    - Copy data from your legacy system, then paste it into the Manage Pricing Charges spreadsheet.
4. Upload:
    - Click **Create or Update** on the command ribbon.
    - Notice that the Changed column indicates the rows that changed.
5. Verify:
    - Examine your records for errors. Use the error message to help correct problems.
    - Make sure price list data in the Pricing Administration work area matches data in the worksheet.

Update Charges Across Price Lists and Business Units

1. Sign into Oracle Pricing with administrative privileges, go to the Pricing Administration work area, then make sure the price list, items, and charges exist.
2. Open your pricing worksheet and navigate to the Manage Pricing Charges worksheet.
3. Click **Search** on the command ribbon.
4. Enter values for the Business Unit, Price List, Item, and so on, then click **OK**.

   The workbook downloads the records from Pricing Administration.
5. Use the Base Price column to update the charges.
6. Make sure the Changed column indicates you changed the record.
7. Click **Create or Update** on the command ribbon.
8. Examine your records for errors. Use the error message to help correct problems.
9. Make sure price list data in the Pricing Administration work area matches data in the worksheet.

Upload New Charges with New Dates

Use the same procedure, except update the end date on each charge you download, or create new charges with new dates.

Related Topics

- Manage Discount Lists
- Pricing Rules
- Manage Price Lists
- Overview of Using Spreadsheets to Manage Pricing
- Add Tiers to Pricing Rules

8 Integrate

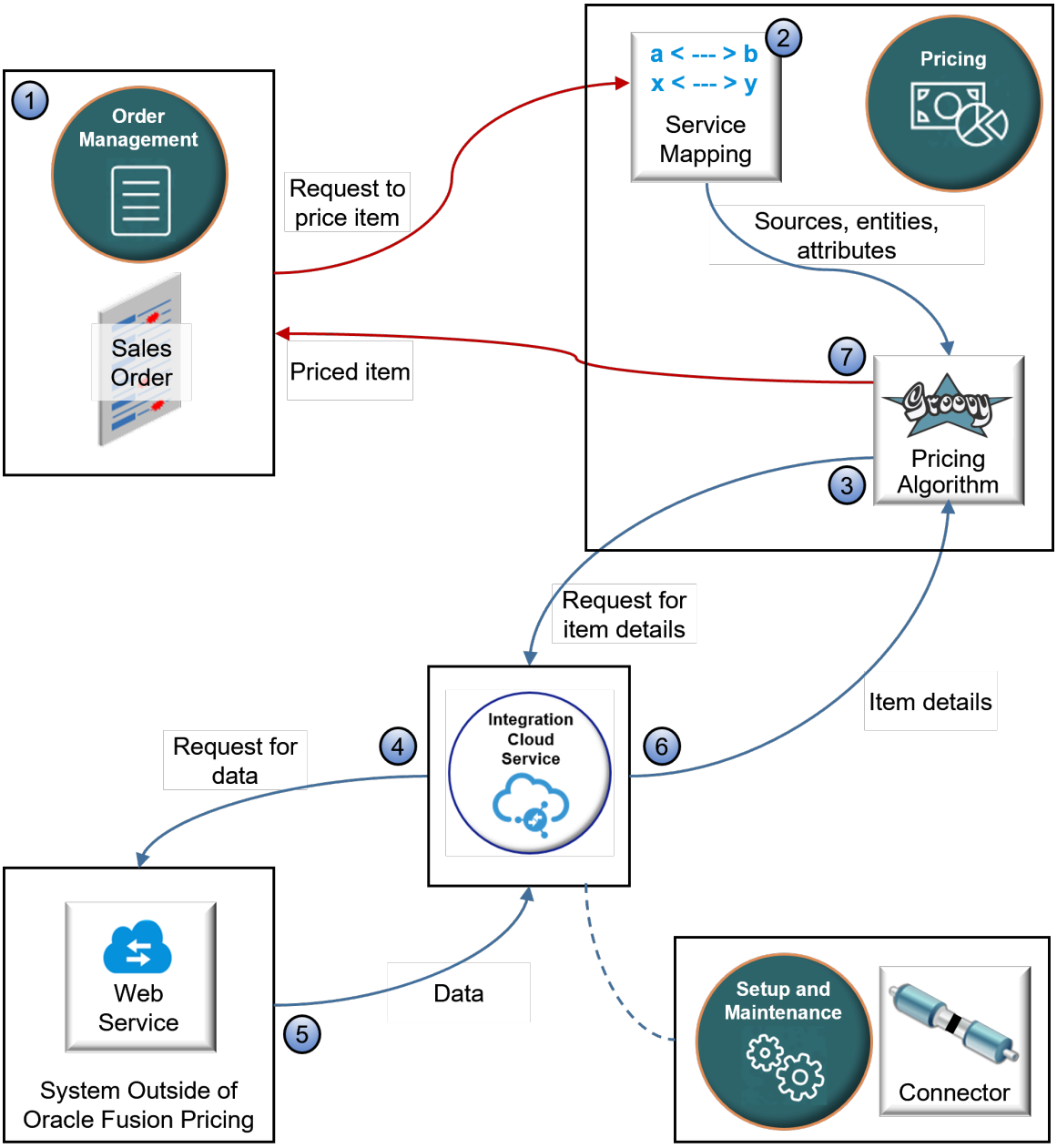
Get Price Details from Cost Accounting

Set up a pricing algorithm to get cost details from Oracle Cost Accounting instead storing these details on a cost list in Oracle Pricing.

Set up a pricing algorithm to get cost details from Oracle Cost Accounting instead storing these details on a cost list in Oracle Pricing. You can use the *Item Cost* SOAP API to periodically call Cost Accounting and get the latest charge data.

> **Note:** In this example, Pricing uses Oracle Integration Cloud Service to establish an integration between Oracle Pricing and Oracle Cost Accounting. You might use a different integration technology. You can use this feature to integrate only with Oracle Cost Accounting. You can't use it to integrate with any other application.

How it works:



Note

1. Create a sales order in Oracle Order Management. Order Management then sends a request to Pricing to price your item.
2. Create a service mapping that specifies sources, entities, and mapping between entities and attributes so Pricing can create the structure for the output SDO it will send to Order Management.
3. Create a pricing algorithm that specifies when to call the web service, what details to get from the web service, and how to process the details it receives from the web service.
4. Set up Oracle Integration Cloud Service so it gets connector details, then sends a SOAP request to the web service endpoint that resides on the system outside of Pricing.

   Use the Manage Connector Details page to specify how to call the web service, including the URL that locates the end point, and the user and password that the end point requires to sign in.
5. The web service sends the reply in a SOAP payload to Integration Cloud Service.

6. Integration Cloud Service sends details to the pricing algorithm.
7. The pricing algorithm uses the details that it receives from the web service to calculate the price, including for pricing rules in Oracle Pricing, then sends price details to Order Management.

You can also use the Price Request Service to determine and validate sales transactions. For details, go to *SOAP Web Services for Oracle Cloud SCM*, then search for Price Request Service.

Related Topics

- Get Cost Details from Oracle Costing
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Service Mapping
- Manage Pricing Algorithms

Get Cost Details from Oracle Costing

Use a web service to get costs from Oracle Costing, then mark it up or down according to rules on the price list.

Oracle Pricing comes predefined to support cost plus pricing through costs that you set up on the Manage Cost Lists page in the Pricing Administration work area. Instead, you can get cost from Oracle Costing and avoid having to duplicate your costing setup.

You modify a pricing algorithm so it calls a SOAP web service. You use Oracle Integration Cloud to route the call.

This topic assumes:

- You have a license to use Oracle Costing.
- You already set up Pricing. For details, see *Roadmap to Manage Oracle Pricing*.
- Customer Computer Service and Rentals is part of the Corporate Segment pricing segment and the Corporate Pricing Strategy.
- You get costs from the costing service according to Inventory Organization.
- If you use Oracle Global Order Promising to promise your sales orders, then you must set up your orchestration process so Promising returns a default value for the warehouse before the process calls Oracle Pricing. For background, see *Guidelines for Setting Up Orchestration Processes*.

If you need help in troubleshooting the setup that you do in this topic, go to *Technical Reference for Oracle Pricing (KB171412)*, then download the Troubleshoot the Get Cost from External System Integration attachment.

The web service is secure, so use a secure tool, such as SOAP UI, to send and receive your payloads.

Summary of the Set Up

A. Test the costing service.

B. Set up your integration.

C. Test the soap service.

D. Set up the connector in Oracle Applications.

E. Edit the service mapping.

F. Troubleshoot.

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

A. Test the Costing Service

Call the costing SOAP web service to make sure it works:

1. Create an input test payload.

   Here's an example input payload you can use.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/scm/costing/itemCosts/itemCostServiceV2/types/" xmlns:item="http://
xmlns.oracle.com/apps/scm/costing/itemCosts/itemCostServiceV2/">
<soapenv:Header/>
 <soapenv:Body>
 <typ:retrieveItemCost>
 <!--Zero or more repetitions:-->
 <typ:costparams>
 <item:ItemId>149</item:ItemId>
 <item:InventoryOrganizationId>204</item:InventoryOrganizationId>
 <item:UnitOfMeasure>Ea</item:UnitOfMeasure>
 </typ:costparams>
 <typ:costparams>
 <item:ItemId>151</item:ItemId>
 <item:InventoryOrganizationId>204</item:InventoryOrganizationId>
 <item:UnitOfMeasure>Ea</item:UnitOfMeasure>
 </typ:costparams>
 </typ:retrieveItemCost>
 </soapenv:Body>
</soapenv:Envelope>
```

2. Send your input payload to the costing service.

   Use the WSDL URL for your instance of Oracle Pricing. Here's an example.

```
http://myServer.myCompany.com:10663/fscmService/ItemCostServiceV2?WSDL
```

3. Examine the response payload that the costing service sends to you.

   Here's an example of the response payload.

```
<env:Envelope 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/costing/itemCosts/service/
types/">
 <env:Header>
 <wsa:Action>http://xmlns.oracle.com/apps/scm/costing/itemCosts/service//ItemCostService/
retrieveItemCostResponse</wsa:Action>
 <wsa:MessageID>urn:uuid:1b2f8967-12ef-4740-8ba0-1b02b85d10c4</wsa:MessageID>
 </env:Header>
 <env:Body>
 <ns0:retrieveItemCostResponse xmlns:ns0="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/
types/">
 <ns2:result xsi:type="ns1:ItemCostOutputResult" xmlns:ns2="http://xmlns.oracle.com/apps/scm/costing/
itemCosts/service/types/" xmlns:ns1="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/"
 xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/" xmlns:ns0="http://xmlns.oracle.com/adf/svc/types/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <ns1:Value>
 <ns1:ItemCost>53.77873553551955</ns1:ItemCost>
 <ns1:CurrencyCode>USD</ns1:CurrencyCode>
```

```
<ns1:UOMCode>Ea</ns1:UOMCode>
<ns1:ErrorFlag>false</ns1:ErrorFlag>
<ns1:ItemID>149</ns1:ItemID>
<ns1:InvOrgID>204</ns1:InvOrgID>
<ns1:SubInvCode xsi:nil="true"/>
<ns1:Locator xsi:nil="true"/>
<ns1:LotNum xsi:nil="true"/>
<ns1:SerialNum xsi:nil="true"/>
<ns1:Grade xsi:nil="true"/>
<ns1:KindOfCost xsi:nil="true"/>
<ns1:KindOfCostDetail xsi:nil="true"/>
<ns1:RefDate>2019-07-15T00:00:00.0Z</ns1:RefDate>
<ns1:ItemNumber xsi:nil="true"/>
<ns1:InvOrgCode xsi:nil="true"/>
<ns1:UnitOfMeasure xsi:nil="true"/>
</ns1:Value>
<ns1:Value>
<ns1:ItemCost>-1.0</ns1:ItemCost>
<ns1:CurrencyCode xsi:nil="true"/>
<ns1:UOMCode>Ea</ns1:UOMCode>
<ns1:ErrorFlag>true</ns1:ErrorFlag>
<ns1:ItemID>151</ns1:ItemID>
<ns1:InvOrgID>204</ns1:InvOrgID>
<ns1:SubInvCode xsi:nil="true"/>
<ns1:Locator xsi:nil="true"/>
<ns1:LotNum xsi:nil="true"/>
<ns1:SerialNum xsi:nil="true"/>
<ns1:Grade xsi:nil="true"/>
<ns1:KindOfCost xsi:nil="true"/>
<ns1:KindOfCostDetail xsi:nil="true"/>
<ns1:RefDate xsi:nil="true"/>
<ns1:ItemNumber xsi:nil="true"/>
<ns1:InvOrgCode xsi:nil="true"/>
<ns1:UnitOfMeasure xsi:nil="true"/>
</ns1:Value>
</ns2:result>
</ns0:retrieveItemCostResponse>
</env:Body>
</env:Envelope>
```

4. Verify the response:
    - Make sure the value under the ItemCost is correct and that its in the correct currency.
    - Make sure each instance of ErrorFlag in the response is false.

B. Set Up Your Integration

Use Integration Cloud Service to route your call from your pricing algorithm to the costing service.

Summary of the Set Up

1. Create WSDL.
2. Create your source connector.
3. Create your target connector.
4. Import a predefined integration.

Learn how to use Integration Cloud Service, including the URL you use when you sign in. For details see *Starting Integration Cloud Service*.

Create WSDL

Copy this code into a text editor, such as Notepad ++, then save it to your local computer with file name ItemCostService.wsdl.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!--
This is an interface WSDL that you can use to configure a connection in Integration Cloud Service. Its a
 proxy for the PriceRequestValidation service. This WSDL contains only XSD definitions, SOAP messages, and
 bindings. It doesn't include services for port operations. This WSDL matches one of the context consumers
 that the integration uses for payload exchanges.
-->

<wsdl:definitions name="ItemCostService"
 targetNamespace="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/"
 xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
 xmlns:errors="http://xmlns.oracle.com/adf/svc/errors/"
 xmlns:tns="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:types="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/types/">
 <wsdl:types>
 <xsd:schema elementFormDefault="qualified"
 targetNamespace="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/"
 sdoJava:package="oracle.apps.scm.costing.itemCosts.service"
 xmlns:sdoJava="commonj.sdo/java"
 xmlns="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"></xsd:schema>
 <xsd:schema elementFormDefault="qualified"
 targetNamespace="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/types/"
 xmlns:ns0="http://xmlns.oracle.com/adf/svc/errors/"
 xmlns="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/types/"
 xmlns:tns="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/types/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="retrieveItemCost">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="CostParams" type="ItemCostInput" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:complexType name="ItemCostInput">
 <xsd:sequence>
 <xsd:element minOccurs="0" name="ItemID" nillable="true" type="xsd:long"/>
 <xsd:element minOccurs="0" name="InvOrgID" nillable="true" type="xsd:long"/>
 <xsd:element minOccurs="0" name="UnitOfMeasureCode" nillable="true" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="retrieveItemCostResponse">
 <xsd:complexType>

 <xsd:sequence>
 <xsd:element name="Result" type="ItemCostOutputResult"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="ItemCostOutputResult">
 <xsd:sequence>
 <xsd:element maxOccurs="unbounded" minOccurs="0" name="Value" type="ItemCostOutput"/>
 </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="ItemCostOutput">
 <xsd:sequence>
```

```
<xsd:element minOccurs="0" name="ItemCost" nillable="true" type="xsd:double"/>
<xsd:element minOccurs="0" name="CurrencyCode" nillable="true" type="xsd:string"/>
<xsd:element minOccurs="0" name="UnitOfMeasureCode" nillable="true" type="xsd:string"/>
<xsd:element minOccurs="0" name="ErrorFlag" nillable="true" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="ItemID" nillable="true" type="xsd:long"/>
<xsd:element minOccurs="0" name="InvOrgID" nillable="true" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

<schema elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/adf/svc/errors/"
sdoJava:package="oracle.jbo.service.errors"
xmlns:sdoJava="commonj.sdo/java"
xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/"
xmlns="http://www.w3.org/2001/XMLSchema">
<element name="ServiceErrorMessage" type="tns:ServiceErrorMessage"/>
<complexType name="ServiceMessage">
<sequence>
<element maxOccurs="1" minOccurs="0" name="code" type="string"/>
<element maxOccurs="1" minOccurs="0" name="message" type="string"/>
<element maxOccurs="1" minOccurs="0" name="severity" type="string"/>
<element maxOccurs="unbounded" minOccurs="0" name="detail"
type="tns:ServiceMessage"/>
</sequence>
</complexType>
<complexType name="ServiceErrorMessage">
<complexContent>
<extension base="tns:ServiceMessage">
<sequence>
<element maxOccurs="1" minOccurs="0" name="sdoObject" type="anyType"/>
<element maxOccurs="1" minOccurs="0" name="exceptionClassName"
type="string"/>
</sequence>
</extension>
</complexContent>
</complexType>
</schema>
</wsdl:types>

<wsdl:message name="ServiceException">
<wsdl:part name="ServiceErrorMessage" element="errors:ServiceErrorMessage"/>
</wsdl:message>

<wsdl:message name="ItemCostService_retrieveItemCost">
<wsdl:part name="parameters" element="types:retrieveItemCost"/>
</wsdl:message>

<wsdl:message name="ItemCostService_retrieveItemCostResponse">
<wsdl:part name="parameters" element="types:retrieveItemCostResponse"/>
</wsdl:message>

<wsdl:portType name="ItemCostService">
<wsdl:documentation/>
<wsdl:operation name="retrieveItemCost">
<wsdl:input message="tns:ItemCostService_retrieveItemCost"/>
<wsdl:output message="tns:ItemCostService_retrieveItemCostResponse"/>
<wsdl:fault name="ServiceException" message="tns:ServiceException"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ItemCostServiceSoapHttp"
type="tns:ItemCostService">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="retrieveItemCost">
<soap:operation soapAction="http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/retrieveItemCost"/>
```

```
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<wsdl:fault name="ServiceException">
<soap:fault name="ServiceException" use="literal"/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

Create Your Source Connector

1. Sign into Integration Cloud Service.
2. On the Welcome page, under Start Here, click **Connections**.
3. On the Connections page, click **Create**.
4. On the Create Connection Select Adapter dialog, select **SOAP**.

   You can also select REST, but for this example, we use SOAP.
5. In the Create New Connection dialog, set the values, then click **Create**:

Attribute	Value
Name	Enter any text. For this example, enter `Connect to Cost Source`.
Identifier	CONNECT_TO_COST_SOURCE
Role	Trigger and Invoke
Description	Connect to the source we use when pricing items outside of Pricing Administration.

6. On the Connect to Cost Source page, click **Configure Connectivity**.
7. In the Connection Properties dialog, set the values, then click **OK**:

Connection Property	Value
WSDL URL	ItemCostService.wsdl Upload the WSDL you saved to your local computer earlier in this procedure.
Target Server's TLS Version	TLSv1.1
Suppress Insertion of Timestamp Into the Request	No

Connection Property	Value
Ignore Timestamp in the Response Message	Y
Enable Two Way SSL for Outbound Connections	Empty
Identify Keystore Alias Name	Empty

8. Click **Configure Security**, set the value, then click **OK**:

Attribute	Value
Security Policy	No Security Policy

9. At the top of the page, click **Save**.
10. Click **Test > Validate and Test**, confirm that your integration displays a message at the top of the page.

```
Connection Connect to Cost Source was tested successfully.
```

11. Click **Close > Close**.

Create Your Target Connector

1. On the Connections page, click **Create**.
2. On the Create Connection Select Adapter dialog, select **SOAP**.
3. In the Create New Connection dialog, set the values, then click **Create**:

Attribute	Value
Name	Enter any text. For this example, enter `Connect to Cost Target`.
Identifier	CONNECT_TO_COST_TARGET
Role	Trigger and Invoke
Description	Connect to the target we use when pricing items outside of Pricing Administration.

4. On the Connect to Cost Target page, click **Configure Connectivity**.

5. In the Connection Properties dialog, set the values, then click **OK**:

Connection Property	Value
WSDL URL	http://myServer.myCompany.com:10663/fscmService/ItemCostServiceV2?WSDL
Target Server's TLS Version	TLSv1.1
Suppress Insertion of Timestamp Into the Request	No
Ignore Timestamp in the Response Message	Y
Enable Two Way SSL for Outbound Connections	Empty
Identify Keystore Alias Name	Empty

6. Click **Configure Security**, set the values, then click **OK**:

Attribute	Value
Security Policy	Username Password Token
Username	Enter the user you use when you administer Oracle Pricing.
Password	Enter the password you use when you administer Oracle Pricing.

7. At the top of the page, click **Save**.
8. Click **Test > Validate and Test**, confirm that your integration displays a message at the top of the page.

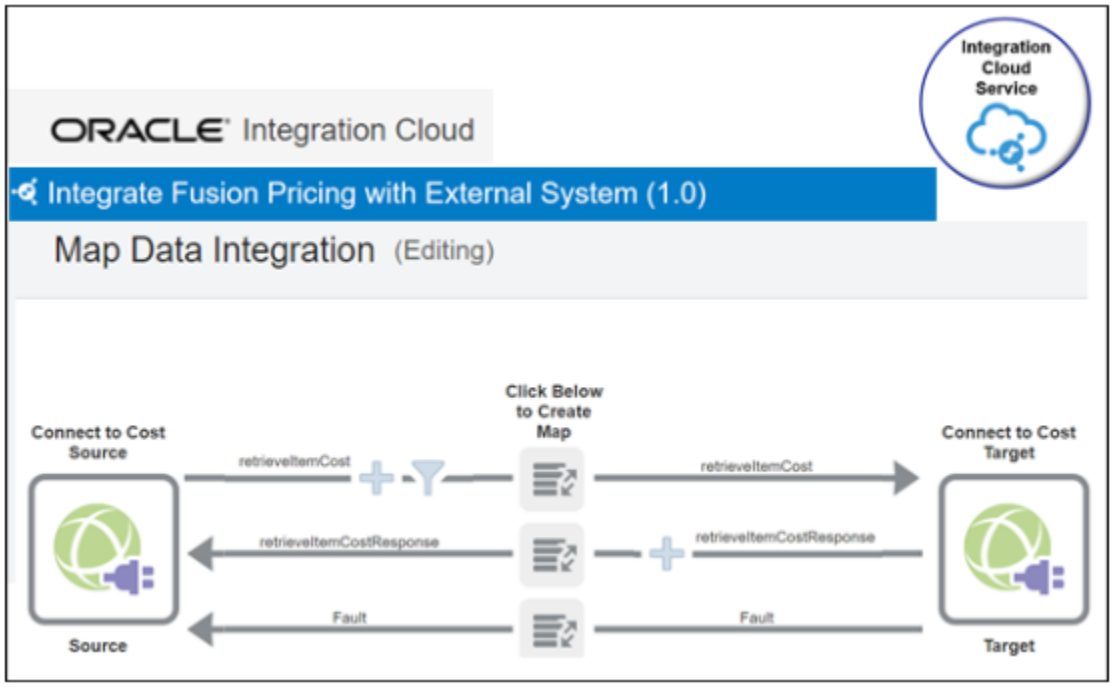
```
Connection Connect to Cost Source was tested successfully.
```

9. Click **Close > Close**.

1. Go to *Technical Reference for Oracle Pricing (KB171412)*, then download the Files That Support Pricing Examples attachment to your laptop.

   The attachment contains the INTEG_FUSIO_PRICI_WITH_EXTER_SYS_01.00.0000.iar file. Save this file to a folder on your laptop.
2. Go to Integration Cloud Service.
3. On the Welcome page, click the **Integrations** icon.
4. On the Integrations page, click **Import**.

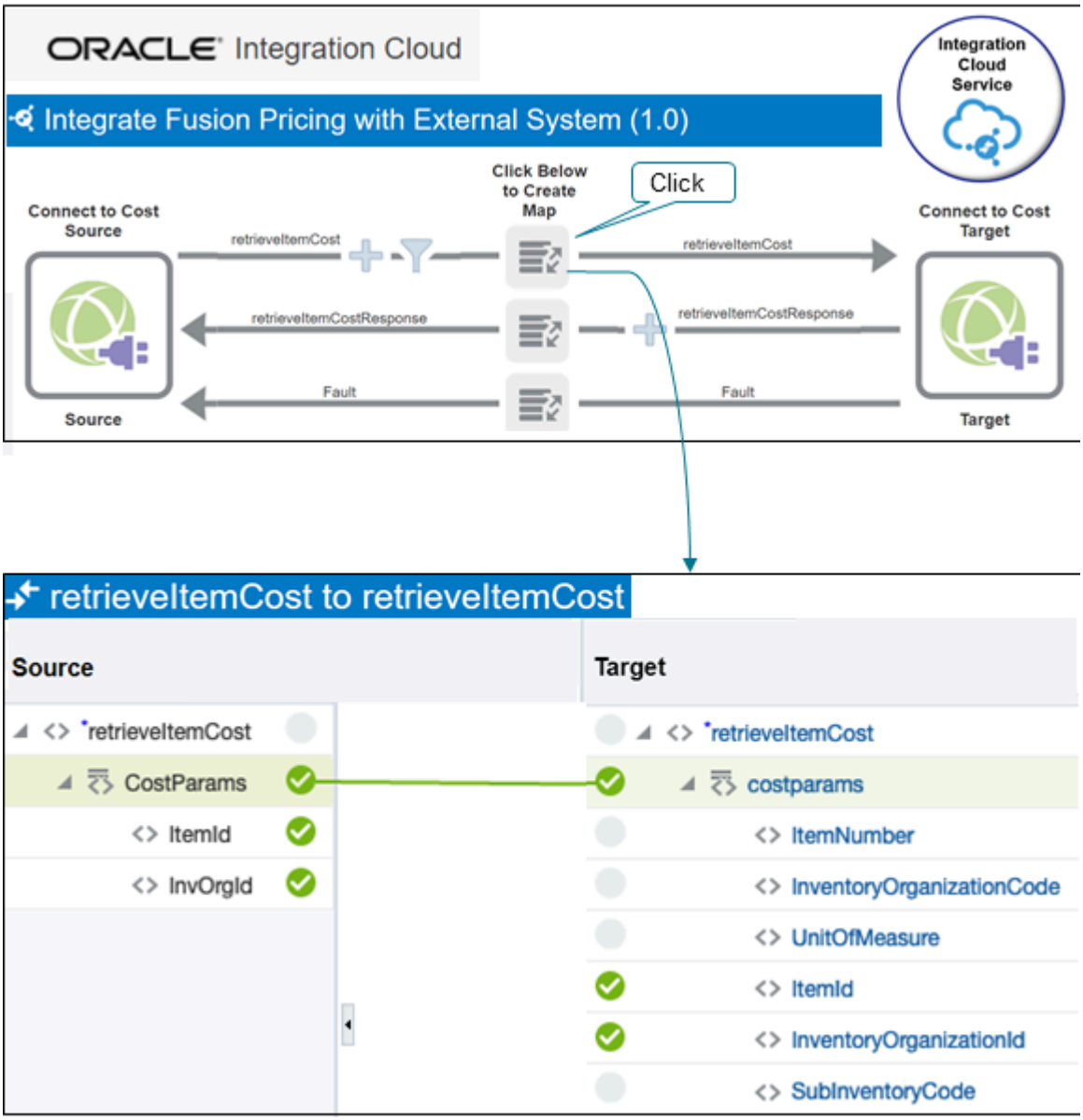
5. In the Import Integration dialog, browse to the `.iar` file that you downloaded, then click **Open**.
6. Click **Import**.
7. On the Integrations page, click **Integrate Fusion Pricing with External System**, which is the integration you just imported.
8. Examine the integration:
    - Notice that the integration connects the source to the target:



Some of the text might be difficult to read in this screen print. Here's the same detail but in a table:

Attribute	Value	
Operation	Operation	Direction
retreiveItemCost	retreiveItemCost	Toward target
retreiveItemCostResponse	retreiveItemCostResponse	Toward source
Fault	Fault	Toward source

- Click the **Request Mapping** icon, then notice the attributes that the integration maps from source to target:

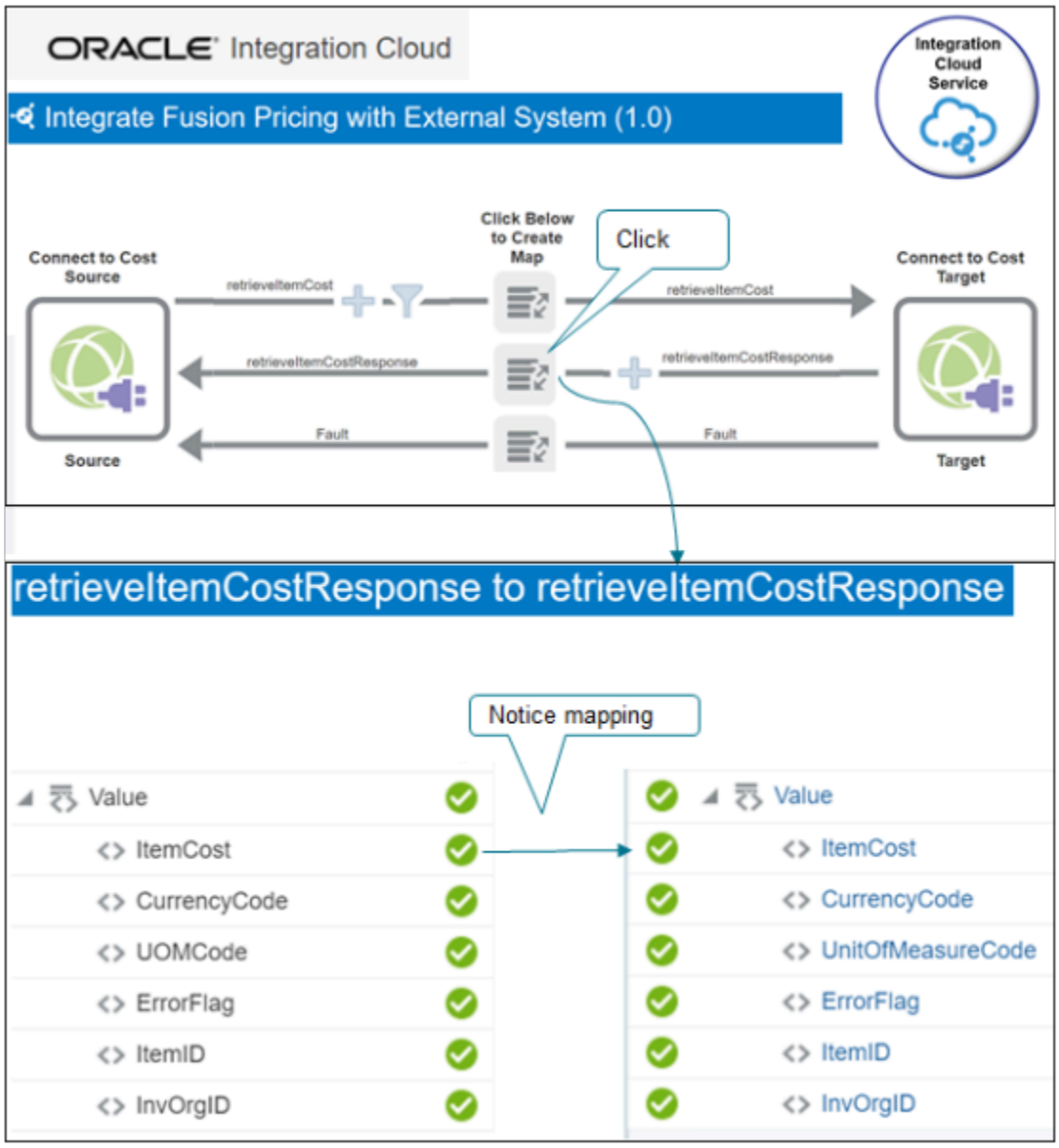


For example:

Attribute in Source	Attribute in Target
itemID	itemID
InvOrgID	InvOrgID

Attribute in Source	Attribute in Target

- Click the **Response Mapping** icon, then notice the attributes that the integration maps from target to source:



For example:

Attribute in Source	Attribute in Target
Value	Value

Attribute in Source	Attribute in Target
itemCost	itemCost
CurrencyCode	CurrencyCode
UOMCode	UnitOfMeasureCode
ErrorFlag	ErrorFlag
ItemId	ItemId
InvOrgId	InventoryOrgId

- Click the **Fault Mapping** icon, then notice how the integration maps attributes to handle any faults that might happen at run time:

Attribute	Value
Fault	ServiceErrorMessage
Route To	ServiceErrorMessage

Activate the Integration

1. On the Integrations page, on the row that contains your integration, click **Actions > Edit**.
2. On the Map Data Integration page, click **Actions > Tracking**.
3. On the Business Identifiers for Tracking dialog, set the value, then click **Done**:

Primary	Tracking Field
Contains a check mark.	ItemID

4. On the Map Data Integration page, on the status bar near the top of the page, verify that the status displays 100%.

   The status displays next to the Last Modified attribute.
5. Click **Save**, then click **Close**.
6. On the Integrations page, on the row that contains your integration, click **Actions > Activate**.

7. On the Activate Integrations dialog, set the values, then click **Activate**:

Option	Value
Enable Tracing	Contains a check mark.
Include Payload	Contains a check mark.

C. Test the SOAP Service

Test your call to the SOAP service:

- Determine the input payloads you need.
- Use SOAP UI to do your test.
- Get the URL for the SOAP web service that you need to do the test from Integration Cloud Service. For example:
  `https://ssotest.oracle.com/integration/flowsvc/soap/COSTING_API_PRICING_INTEG/v01/`
- Examine the response payloads.
- Test for more than one item to make sure the service returns the correct cost for different types of calculations.
- Include error cases during your test to make sure the costing service returns errors correctly through the integration.

Here's an example input payload you can use to send to the SOAP service during the test. Example address to the WSDL.

```
https://server.integration.us2.oraclecloud.com/integration/flowsvc/soap/CGB_INTE_ORAC_FUS_PRIC_WITH_ANTH/
v01/?wsdl
```

Example payload.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://
xmlns.oracle.com/apps/scm/costing/itemCosts/service/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-772E6BAE6F9F39812416003717946724">
 <wsu:Created>2020-09-17T19:43:14.670Z</wsu:Created>
 <wsu:Expires>2020-09-17T19:44:14.670Z</wsu:Expires>
 </wsu:Timestamp>
 <wsse:UsernameToken wsu:Id="UsernameToken-772E6BAE6F9F398124160037142746 53">
 <wsse:Username>shailendra.pitre@oracle.com</wsse:Username>
 <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">WedAug05@0@0</wsse:Password>
 <wsse:Nonce EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-
security-1.0#Base64Binary">zFzlHdJthMM1ZS5uXnmCcQ==</wsse:Nonce>
 <wsu:Created>2020-09-17T19:37:07.465Z</wsu:Created>
 </wsse:UsernameToken>
 </wsse:Security>
 </soapenv:Header>
 <soapenv:Body>
 <typ:retrieveItemCost>
 <typ:CostParams>
 <!--Optional:-->
```

```
 <typ:ItemID>149</typ:ItemID>
 <!--Optional:-->
 <typ:InvOrgID>204</typ:InvOrgID>
 <!--Optional:-->
 <typ:UnitOfMeasureCode>Ea</typ:UnitOfMeasureCode>
 </typ:CostParams>
 </typ:retrieveItemCost>
 </soapenv:Body>
</soapenv:Envelope>
```

Here's an example response payload that the SOAP service sends to you. Examine it to make sure ItemCost contains the amount you expect for ItemID and InvOrgID.

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
 <wss:costServiceResponse xmlns:wss="http://www.boomi.com/connector/wss">
 <Value>
 <ItemCost>12484.78</ItemCost>
 <CurrencyCode>GBP</CurrencyCode>
 <UnitOfMeasureCode>Ea</UnitOfMeasureCode>
 <ErrorFlag>false</ErrorFlag>
 <ItemID>100000001582378</ItemID>
 <InvOrgID>300000037424187</InvOrgID>
 </Value>
 <Value>
 <ItemCost>11404.41</ItemCost>
 <CurrencyCode>GBP</CurrencyCode>
 <UnitOfMeasureCode>Ea</UnitOfMeasureCode>
 <ErrorFlag>false</ErrorFlag>
 <ItemID>100000001582379</ItemID>
 <InvOrgID>300000037424187</InvOrgID>
 </Value>
 </wss:costServiceResponse>
 </S:Body>
</S:Envelope>
```

D. Set Up the Connector in Oracle Applications

1. Sign into Oracle Applications.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage External Service Details for Algorithms**.

   This example assumes you're setting up the Order Management offering. If you use a different offering, then click it instead of Order Management.
3. On the Manage Connector Details page, create a connection to the system that hosts the web service that your pricing algorithm calls. For this example, assume it calls the Oracle Costing Service:

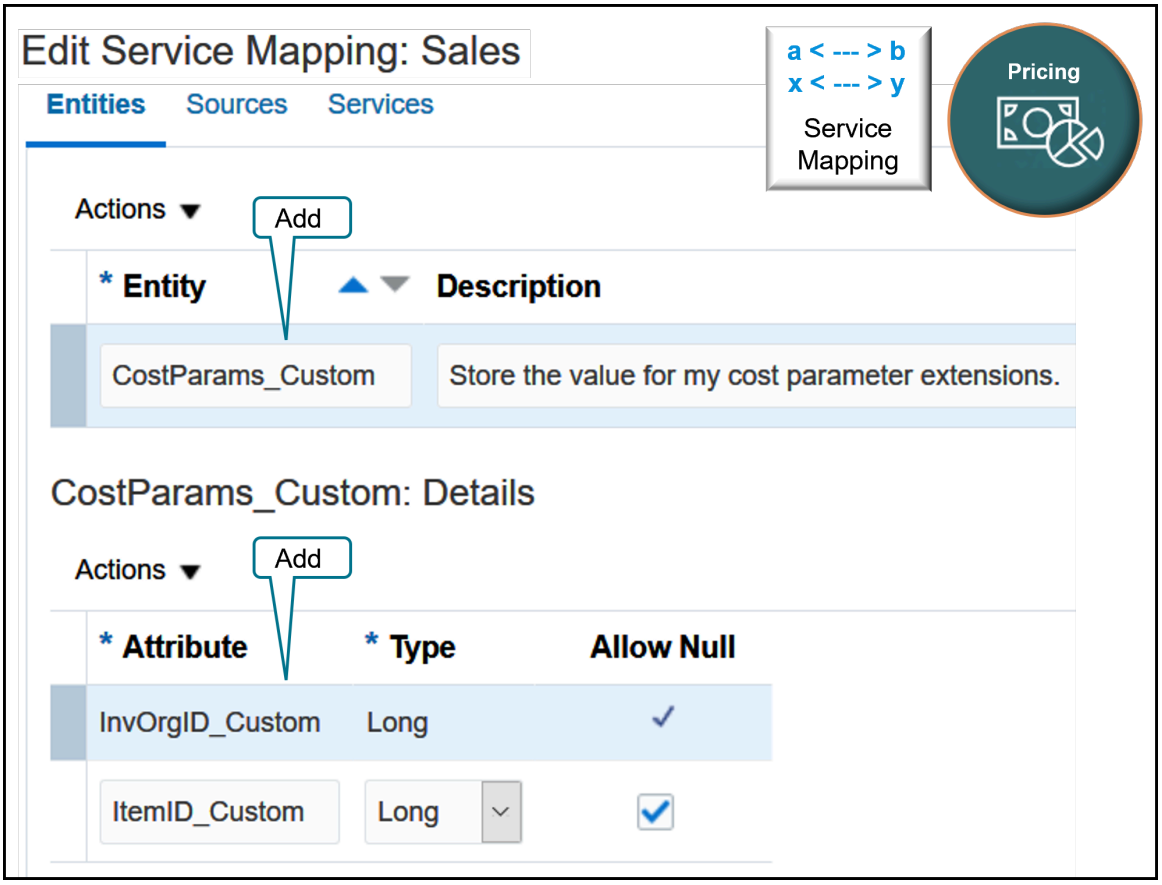
Attribute	Value
Target System	Fusion_HCM You must first use the Setup and Maintenance work area to specify this system as a trading community source system.
Connector Name	QP_FUSION_COSTING_ICS_SRVC

Attribute	Value
	You can use any value, but it must match the value that the pricing algorithm sends to the executeSOAPService function.
Connector URL	Enter the URL that locates the server and port that your web service uses as the end point. For example: `https://ssotest. oracle.com/integration/flowsvc/soap/COSTING_API_PRICING_INTEG/v01/`
User Name	Enter the user name required to access your web service.
Password	Enter the password required to access your web service.
Invocation Mode	Synchronous

For details, see *Connect Order Management to Your Fulfillment System*.

E. Edit the Service Mapping

Add entities to the Sales service mapping. Here's the first one you add:



Make sure you use the exact attribute name and values. If you don't, the mapping will fail.

Try it:

1. Sign into Oracle Pricing with administrative privileges.
2. Go to the Sandboxes work area.
3. On the Sandboxes, page, open and enter your sandbox, or create a new one.
4. Go to the Pricing Administration work area, click **Tasks**, then under Pricing Configuration, click **Manage Service Mappings**.
5. On the Manage Service Mappings page, click **Sales**.
6. On the Edit Service Mapping page, on the Entities tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Entity	CostParams_Custom
Description	Store the value for my cost parameter extensions.

7. In the Details area, add two attributes. Make sure Allow Null contains a check mark for each attribute:

Attribute	Value
InvOrgID_Custom	Long
ItemID_Custom	Long

8. Add another entity. On the Entities tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Entity	Value_Custom
Description	Store the value of the CostParams input element.

9. In the Details area, add attributes to the Value_Custom entity. These attributes will contain the details that you communicate to and from a system that resides outside of Oracle Pricing. Make sure Allow Null contains a check mark for each attribute:

Attribute	Value
CurrencyCode_Custom	String
ErrorFlag_Custom	Boolean
InvOrgID_Custom	Long
ItemCost_Custom	Decimal
ItemID_Custom	Long
UOMCode_Custom	String

10. Add another entity that copies the definition that the web service uses. On the Entities tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Entity	Result_Custom

Attribute	Value
Description	Store the value of the output element.

11. On the Entities tab, click the **row** that contains Header in the Entity column.
12. In the Details area, click Actions > Add Row, then set the values:

Attribute	Value
Entity	FulfillOrgId_Custom If you use a list of values in a search dialog to specify the item price, then you must use a pretransformation to populate the value in the FulfillOrgId attribute.
Type	Long

13. Click **Save**.

Add the Source to the Service Mapping

1. Click **Sources**.
2. Add an attribute to the OrderCatalogLine source:
    - On the Sources tab, click the **row** that contains OrderCatalogLine in the Source column.
    - In the Details area, on the Entity Mappings tab, click the **row** that contains Header in the Entity column.
    - In the Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FulfillOrgId_Custom
View Object Attribute	FulfillOrgId

3. Add an attribute to the OrderHeader source:
    - On the Sources tab, click the **row** that contains OrderHeader in the Source column.
    - In the Details area, on the Entity Mappings tab, click the **row** that contains Header in the Entity column.
    - In the Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FulfillOrgId_Custom

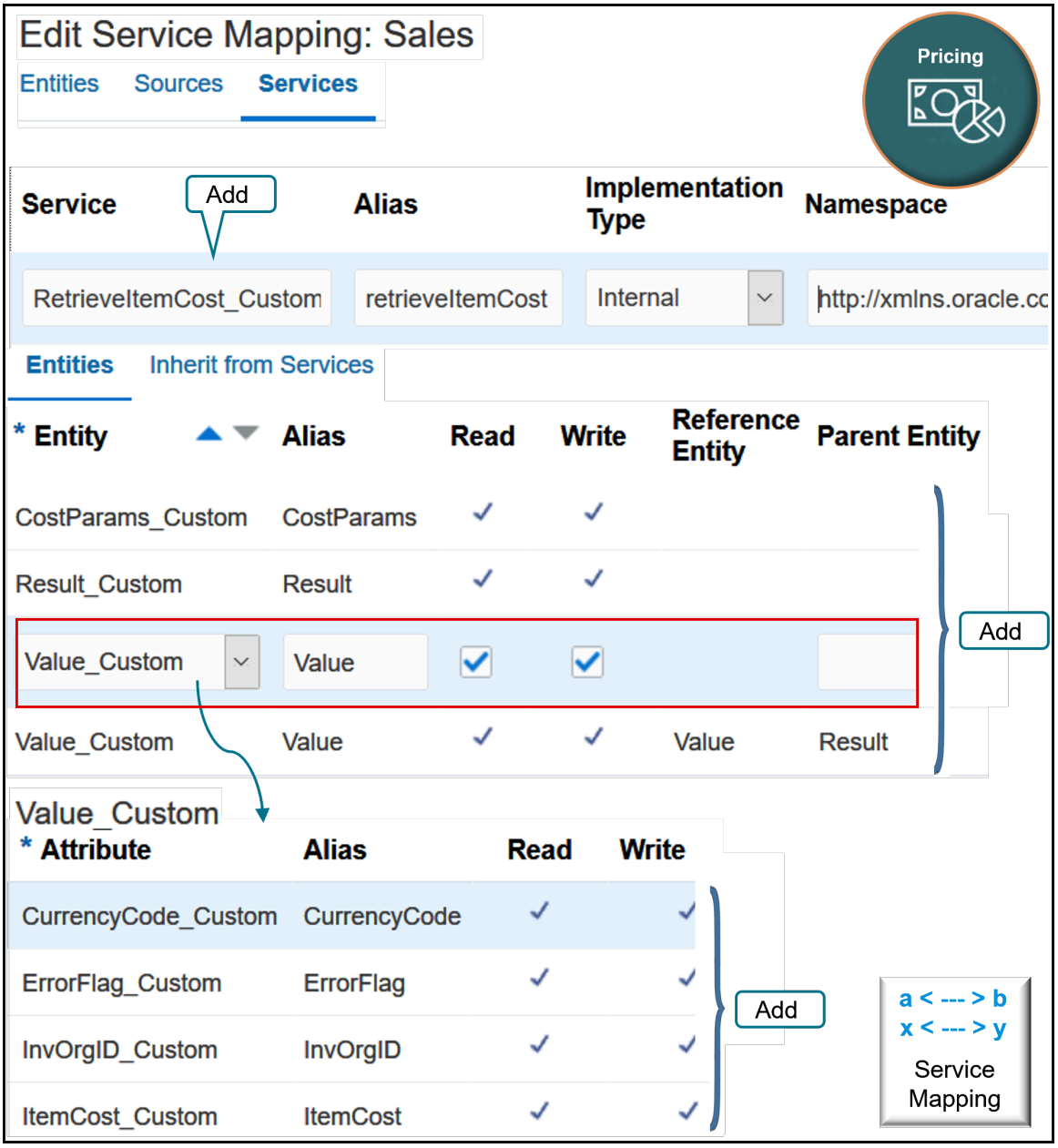
Attribute	Value
View Object Attribute	FulfillOrgId

4. Add an attribute to the OrderLine source:
    - On the Sources tab, click the **row** that contains OrderLine in the Source column.
    - In the Details area, on the Entity Mappings tab, click the **row** that contains Header in the Entity column.
    - In the Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FulfillOrgId_Custom
View Object Attribute	FulfillOrgId

Add the Service to the Service Mapping

Specify the end point that locates the system that resides outside of Oracle Pricing, and specify the entities that you will use to communicate details with the service:



The screen capture includes:

- The service you add.
- The entities you add to the service.
- The Value_Custom entity that doesn't have a value in the Reference Entity attribute is selected.
- The attributes for the first Value_Custom entity.

Try it:

1. On the Edit Service Mapping page, click **Services**.
2. Click **View > Columns**, then add a check mark to Alias and Namespace.
3. Click **Actions**, click **Add Row**, then set the values:

Attribute	Value
Service	RetrieveItemCost_Custom You can enter any text. Describe what the service does.
Alias	retrieveItemCost
Implementation Type	Internal
Namespace	Enter the URL that locates the end point on the system that resides outside of Oracle Pricing, and that will communicate the SOAP payloads with Pricing. For example: `http://xmlns.oracle.com/apps/scm/costing/itemCosts/service/types/` Don't use `http://my_server.com:7012`.

4. In the Details area, on the Entities tab, add the entities. Make sure the Read attribute and the Write attribute for each entity contains a check mark:

Entity	Alias	Parent
CostParams_Custom	CostParams	Leave empty.
Result_Custom	Result	Leave empty.
Value_Custom	Value	Leave empty.

5. Click **Save**.
6. In the Details area, on the Entities tab, click **View > Columns**, then add a check mark to Reference Entity.
7. In the Details area, on the Entities tab, add another entity. Make sure the Read attribute and the Write attribute each contain a check mark:

Entity	Alias	Reference Entity	Parent Entity
Value_Custom	Value	Value	Result

The definition of the Oracle Costing Service includes a three level hierarchy.

```
Root
 CostParams and Result
```

```
Value
```

To create this hierarchy, you:

- Add the Value_Custom entity and leave the Parent Entity empty.
- Add another Value_Custom entity, specify the reference entity as Value, and specify the Parent Entity as Result.

8. In the Entity column, click **CostParams_Custom**.
9. In the CostParams_Custom Entities area, add attributes. Make sure the Read attribute and the Write attribute for each attribute contains a check mark. Leave other attributes empty:

Attribute	Alias
InvOrgID_Custom	InvOrgID
ItemID_Custom	ItemID

The pricing algorithm uses these aliases, and they're case sensitive.

10. In the Details area, on the Entities tab, click the Value_Custom **row** that doesn't contain a value in the Reference Entity attribute.
11. In the Value_Custom Entities area, add the entities. Make sure the Read attribute and the Write attribute for each entity contains a check mark. Leave other attributes empty:

Attribute	Alias
CurrencyCode_Custom	CurrencyCode
ErrorFlag_Custom	ErrorFlag
InvOrgID_Custom	InvOrgID
ItemCost_Custom	ItemCost
ItemID_Custom	ItemID
UOMCode_Custom	UOMCode

12. Click **Save**.

It isn't necessary to use the Sources tab to create the source because you typically use the Sources tab to specify a source when a virtual object populates the web service. However, in this example, you use a pricing algorithm and call a web service to get the details that the web service needs.

Update the PriceSalesTransaction Service

1. On the Services tab, click Query By Example, then query for the value:

Attribute	Value
Service	PriceSalesTransaction

2. In the Details area, on the Entities tab, click Query By Example, then query for the value:

Attribute	Value
Entity	Header

3. In the Entities area, click **Action > Add Row**, set the values, then click **Save and Close**:

Attribute	Value
Attribute	FulfillOrgId_Custom
Read	Contains a check mark
Write	Contains a check mark

F. Troubleshoot

Here are some tips:

- If your pricing algorithm reports a 401 Unauthorized exception while calling the costing service, then use a secured service link in the service registration.
- If the charge breakdown in Order Management doesn't display your cost charge components, then you must add them. For details, see *Manage Order Line's Price Details*.
- Assess your extended algorithms to determine whether you must modify them when you update. For details, see *Performing Your Quarterly Update (Doc ID 2337485.1)*.

Call a Different Service from the Pricing Algorithm

This topic describes how to call the Costing SOAP service from a pricing algorithm, but you can call a different service to meet your specific requirements:

1. Determine whether the schema for your WSDL service must use Integration Cloud Service. Most SOAP services do require that you use Integration Cloud Service.
2. If you don't need Integration Cloud Service, then skip sections B and C.

3. In section D, If you:
    - Don't need Integration Cloud Service, set up the connector so it connects directly to the SOAP service.
    - Do need Integration Cloud Service, set up the connector so it connects directly to Integration Cloud Service.
4. In Step E, set up the service mappings for each request schema and response schema of the WSDL.
5. In Step F, set up the algorithm according to your deployment requirements:
    - Identify the pricing algorithms that you must modify. For details, see *Pricing Process* and the content starting at *Example Pricing Algorithm, Part 1, Get Reference Attributes*.
    - Identify the step in each algorithm where you must call the service.
    - Determine the input values that you must use to so you can interface with the service.
    - Determine how you will process the output from the service, including how you will use the algorithm variables to process the output.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Create a Sandbox So You Can Edit Service Mappings
- Manage Order Line's Price Details
- Pricing Process
- Connect Order Management to Your Fulfillment System

Do Cost Plus Pricing with Oracle Costing

Set up Oracle Pricing to get cost details from Oracle Costing.

Summary of the Set Up

1. Do the set up described in *Get Cost Details from Oracle Costing*.
2. Create a pricing algorithm.
3. Modify a predefined pricing algorithm.

1. Create a Pricing Algorithm

Create a pricing algorithm that gets pricing from the cost service:

1. Download the algorithm:
    - Go to *Technical Reference for Oracle Pricing (KB171412)*.
    - Download the attachment named Files That Support Pricing Examples.
    - Open the attachment, then save the GetPricefromExternalApplicationCustom.zip file to your computer.
2. Sign into Oracle Pricing with administrative privileges.
3. Go to the Pricing Administration work area, the click **Tasks > Manage Algorithms** under Pricing Configuration.
4. On the Manage Algorithms page, click **Actions > Import**.
5. In the dialog that displays, browse to the file you saved, GetPricefromExternalApplicationCustom.zip, then click **OK**.

6. On the Manage Algorithms page, query for the Get Price from External Application Custom algorithm, then open it for editing.
7. On the Edit Algorithm page, notice that the algorithm includes three steps:

Step	Value
Create Costing Input	Iterates over the ChargeCandidate variable to identify candidates that need cost. It includes data sets for ChargeCandidate and each candidate line. You can add data sets, as necessary, but the predefined data should meet most of your requirements. The default action uses InventoryItemId to create an entry in the CostingInput variable for each item that needs a cost plus calculation. It also uses InventoryOrganizationId from the line for ChargeCandidate. If you must price according to the warehouse, then remove InventoryOrganizationId, and replace it with ShipFromPartyId.

Invoke Costing Service	The predefined script in this step should meet most of your requirements.
	
	If necessary, you can modify it.
	
	`executeSOAPService('QP_FUSION_COSTING_ICS_SERVICE', //Service Name CostingInput, //input [operation:'retrieveItemCost', inContext:'Sales.retrieveItemCost_Custom', outContext:'Sales.retrieveItemCost_Custom'])`
	
	To make the connection when you modify it, use the connector that you set up in section D Set Up the Connector in Oracle Applications. For details, see *Get Cost Details from Oracle Costing*.
	
	Next, this script gets the output from the costing service.
	
	`CostingOutput = serviceOutputMap.get("outputSDO")`
	
	Note that the integration automatically adds the CostingInput entity and the CostingOutput entity when it calls the Get Price from External Application Custom pricing algorithm.

Process Costing Output	Iterates over the ChargeCandidate entries that need costs. The CostOutput data set corresponds to the item on the line. The actions determine whether the service successfully returned the costs, and updates the CostValue on the ChargeCandidate. As an option, if the service doesn't return a cost for an item, then use a PricingMessage to mark the corresponding line in error.

8. Click **Test**, then click **Actions > Add Row** to add a test case.

9. On the Test Input tab, notice that the test case includes a row for the PriceRequest variable and some code in the Variable Value column.

   The code doesn't contain any values, so you will replace it with code that does include example values.
10. In the Variable Value column, click the **pencil**.
11. In the Edit Variable dialog, delete all the code, paste this code into the dialog, then click **OK**.

```
<
? xml version = "1.0"
encoding = "UTF-8" ? >
 <
 PriceRequestInternal : PriceRequestInternalType xmlns: ns0 = "http://xmlns.oracle.com/adf/svc/types/"
xmlns: PriceRequestInternal = "http://xmlns.oracle.com/apps/scm/pricing/priceExecution/pricingProcesses/
pricingInternal/PricingInternal"
xmlns: xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi: type = "PriceRequestInternal:PriceRequestInternalType" >

 <
 PriceRequestInternal: Header >
 <
 PriceRequestInternal: CustomerId > 1000 < /PriceRequestInternal:CustomerId> <
 PriceRequestInternal: HeaderId > 1 < /
PriceRequestInternal:HeaderId><PriceRequestInternal:CalculatePricingChargesFlag>true</
PriceRequestInternal: CalculatePricingChargesFlag > < PriceRequestInternal: CalculateShippingChargesFlag
 > false < /PriceRequestInternal:CalculateShippingChargesFlag> <
 PriceRequestInternal: CalculateTaxFlag > false < /PriceRequestInternal:CalculateTaxFlag> <
 PriceRequestInternal: SellingBusinessUnitId > 204 < /PriceRequestInternal:SellingBusinessUnitId> <
 PriceRequestInternal: SellingLegalEntityId > 204 < /PriceRequestInternal:SellingLegalEntityId> <
 PriceRequestInternal: TransactionTypeCode > ORA_SALES_ORDER < /
PriceRequestInternal:TransactionTypeCode> <
 /PriceRequestInternal:Header> <
 PriceRequestInternal: PricingServiceParameter >
 <
 PriceRequestInternal: PricingContext > SALES < /PriceRequestInternal:PricingContext> <
 /PriceRequestInternal:PricingServiceParameter> <
 PriceRequestInternal: Line >
 <
 PriceRequestInternal: ShipFromPartyId > 300000037424187 < /PriceRequestInternal:ShipFromPartyId> <
 PriceRequestInternal: HeaderId > 1 < /PriceRequestInternal:HeaderId> <
 PriceRequestInternal: InventoryItemId > 100000001582379 < /PriceRequestInternal:InventoryItemId> <
 PriceRequestInternal: InventoryOrganizationId > 300000037424187 < /
PriceRequestInternal:InventoryOrganizationId> <
 PriceRequestInternal: LineId > 1 < /PriceRequestInternal:LineId> <
 PriceRequestInternal: LineCategoryCode > ORDER < /PriceRequestInternal:LineCategoryCode> <
 PriceRequestInternal: LineQuantity unitCode = "Ea"
xmlns: tns = "http://xmlns.oracle.com/adf/svc/errors/" > 2 < /PriceRequestInternal:LineQuantity> <
 PriceRequestInternal: LineQuantityUOMCode > Ea < /PriceRequestInternal:LineQuantityUOMCode> <
 PriceRequestInternal: LineTypeCode > ORA_BUY < /PriceRequestInternal:LineTypeCode> <
 /PriceRequestInternal:Line>

 <
 PriceRequestInternal: ChargeCandidate >
 <
 PriceRequestInternal: BasePrice > < /PriceRequestInternal:BasePrice> <
 PriceRequestInternal: CalculateMarginFlag > Y < /PriceRequestInternal:CalculateMarginFlag> <
 PriceRequestInternal: CalculationMethod > COST < /PriceRequestInternal:CalculationMethod> <
 PriceRequestInternal: CalculationTypeCode > MARKUP_PERCENT < /PriceRequestInternal:CalculationTypeCode>
 <
 PriceRequestInternal: CostCalculationAmount > 30 < /PriceRequestInternal:CostCalculationAmount>

 <
 PriceRequestInternal: CanAdjustFlag > Y < /PriceRequestInternal:CanAdjustFlag> <
 PriceRequestInternal: ChargeAppliesTo > PRICE < /PriceRequestInternal:ChargeAppliesTo> <
```

```
 PriceRequestInternal: ChargeDefinitionCode > QP_SALE_PRICE < /
PriceRequestInternal:ChargeDefinitionCode> <
 PriceRequestInternal: ChargeDefinitionId > 300100070841552 < /PriceRequestInternal:ChargeDefinitionId>
 <
 PriceRequestInternal: ChargeId > 1 < /PriceRequestInternal:ChargeId> <
 PriceRequestInternal: ChargeSubtypeCode > ORA_PRICE < /PriceRequestInternal:ChargeSubtypeCode> <
 PriceRequestInternal: ChargeTypeCode > ORA_SALE < /PriceRequestInternal:ChargeTypeCode> <
 PriceRequestInternal: CurrencyCode > USD < /PriceRequestInternal:CurrencyCode> <
 PriceRequestInternal: DenormDistanceNum xsi: nil = "true" / >
 <
 PriceRequestInternal: ItemId > 149 < /PriceRequestInternal:ItemId> <
 PriceRequestInternal: ItemLevelCode > ITEM < /PriceRequestInternal:ItemLevelCode> <
 PriceRequestInternal: ItemLevelPrecedence > 1 < /PriceRequestInternal:ItemLevelPrecedence> <
 PriceRequestInternal: ItemType > STANDARD < /PriceRequestInternal:ItemType> <
 PriceRequestInternal: LineTypeCode > ORA_BUY < /PriceRequestInternal:LineTypeCode> <
 PriceRequestInternal: ParentEntityCode > LINE < /PriceRequestInternal:ParentEntityCode> <
 PriceRequestInternal: ParentEntityId > 1 < /PriceRequestInternal:ParentEntityId> <
 PriceRequestInternal: PriceListChargeId > 300100071623860 < /PriceRequestInternal:PriceListChargeId> <
 PriceRequestInternal: PriceTypeCode > ONE_TIME < /PriceRequestInternal:PriceTypeCode> <
 PriceRequestInternal: PricingUomCode > Ea < /PriceRequestInternal:PricingUomCode> <
 PriceRequestInternal: PrimaryPricingUomFlag > Y < /PriceRequestInternal:PrimaryPricingUomFlag> <
 PriceRequestInternal: StartDate > 2009 - 01 - 01 T09: 00: 00.0 Z < /PriceRequestInternal:StartDate> <
 PriceRequestInternal: Type > SEGMENT_PRICE < /PriceRequestInternal:Type> <
 PriceRequestInternal: NeedsCost > true < /PriceRequestInternal:NeedsCost> <
 /PriceRequestInternal:ChargeCandidate>

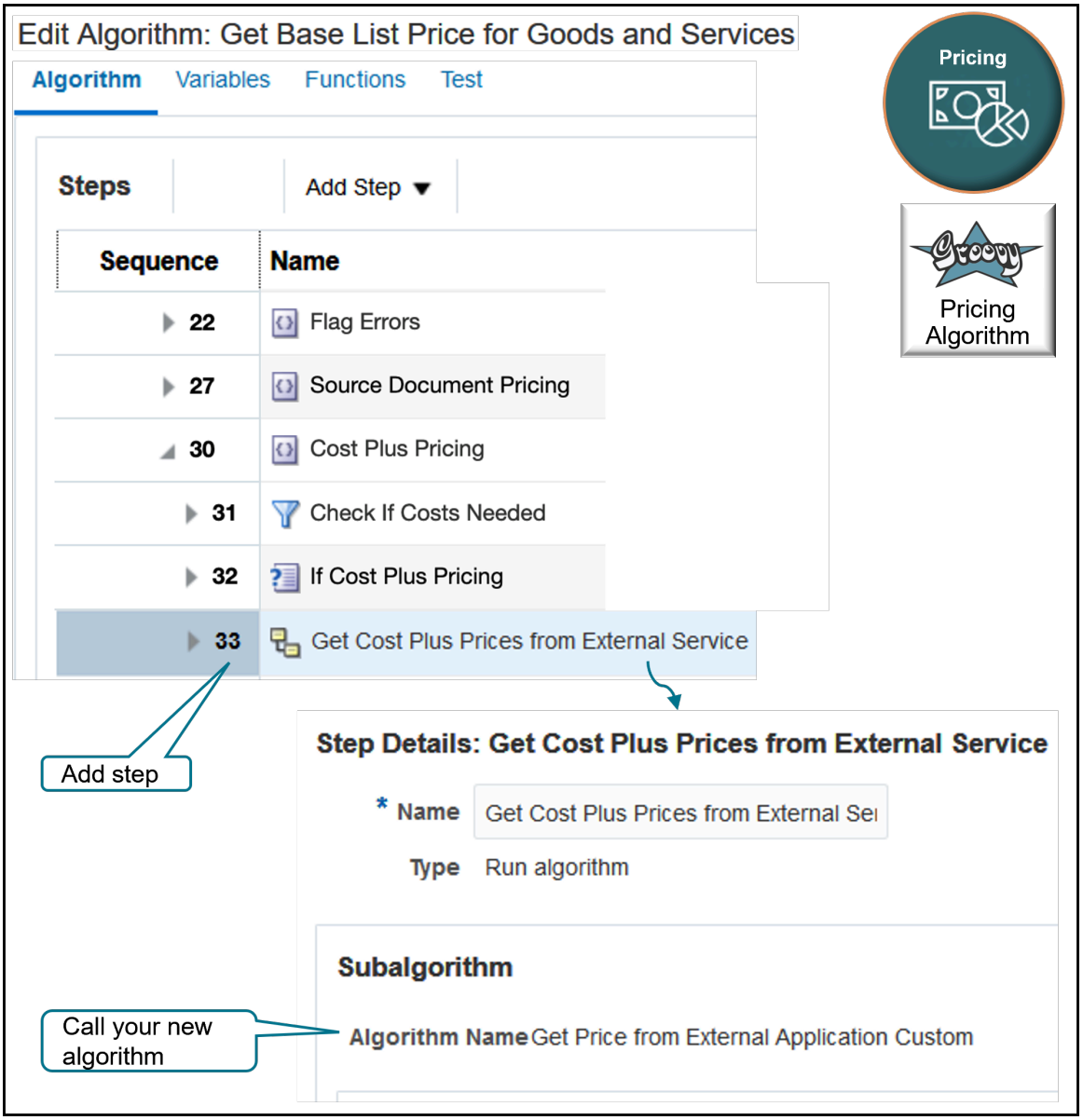
 <
 PriceRequestInternal: ChangeSummary logging = "false"
xmlns: sdo = "commonj.sdo" / >
 <
 /PriceRequestInternal:PriceRequestInternalType>
```

12. You're getting costs directly according to items on the line, so remove any references to ChargeCandidate in the Create Costing Input step and the Process Costing Output step. Make sure the primary set is Line.
13. Click **Save**.
14. Click **Run Test**, then verify that the test finishes without error.

    Order Management doesn't determine the actual warehouse for the sales order until you submit on the sales order. So, if you use ShipFromPartyId to get costs according to the warehouse, then you must set up a defaulting rule that sets the default value for the warehouse on the order header and the order line. If you don't have a defaulting rule, then Pricing won't calculate the cost. For details, see *Overview of Using Business Rules With Order Management*.
15. Click **Save and Close**.
16. On the Manage Algorithms page, click **Actions > Publish**, then verify that the Status attribute contains Published for the Get Price from External Application Custom algorithm.

2. Modify a Predefined Pricing Algorithm

Modify the Get Base List Price for Goods and Services algorithm. This algorithm calculates cost plus pricing:



Try it:

1. Create a version of the pricing algorithm that you need to edit:
    - Click **Tasks > Manage Algorithms** under Pricing Configuration.
    - On the Manage Algorithms page, search for the pricing algorithm:

Attribute	Value
Name	Get Base List Price for Goods and Services

Attribute	Value

- Click **Actions > Create Version**, locate the version that has this status, then open it for editing:

Attribute	Value
Status	In Progress

2. Modify the test that you use to determine whether to process cost plus pricing:
   - On the Edit Algorithm page, in the Sequence column, expand step **15** in the tree, expand the Cost Plus Pricing step, then click the step that has the value:

Attribute	Value
Name	If Cost Plus Pricing

   - Delete the content in the Condition window and replace it with this code:

Attribute	Value
Condition	`false && NeedsCostPlus && finer('==== Invoking Cost-Plus Pricing ====') == null`

   - Click **Save**.
3. Modify cost plus pricing:
   - Click the Cost Plus Pricing step.
   - Click **Add Step > Composite Step > If**.
   - Click the new step you just added and set the values:

Attribute	Value
Name	If External Costing Plus Pricing
Condition	`NeedsCostPlus && finer('==== Invoking Cost-Plus Pricing ====') == null`

   - Click **Save**.

4. Add the subalgorithm:
    - In the tree, click the triangle on the step you just added, If External Costing Plus Pricing.

      This helps to make sure that the subalgorithm you add is a child of If External Costing Plus Pricing.
    - Click **Add Step > Subalgorithm**.
    - In the Step Details area, set the values:

Attribute	Value
Name	Compute Cost from External Systems
Algorithm Name	Get Price From External Application Custom This is the algorithm that you created in the previous section in this topic.

    - In the Input Variables area, set the values:

Attribute	Value
Variable	PriceRequest
Assignment Value	PriceRequest

    - In the Output Variables area, set the values:

Attribute	Value
Variable	PriceRequest
Assignment Value	PriceRequest

5. Click **Save and Close**, then publish the highest version of Get Base List Price for Goods and Services.

Notes

- When you process the output from the costing service, you use the cost value to create a cost charge component for each item that the algorithm successfully calculates. Use the Write Costs step in the Calculate Cost algorithm as an example.
- If you price a configurable item, then call the Rollup and Aggregate Charge Components subalgorithm to roll up and aggregate the costs. Call the new custom algorithm that you created from the algorithm that calculates cost.

Implement Cost Plus Pricing with Systems That Are Part of Oracle

If you do the steps described in the sections earlier in this topic, then the pricing algorithms already calculated cost on the ChargeComponents. Here's what you need to do:

- Create a new version of the algorithm that calculates costs.
- Disable the steps that get cost from the cost lists in Oracle Pricing.
- Add a step that iterates over the ChargeCandidate entries that need the cost plus markup. Get the CostValue from the ChargeCandidate and create a Cost Charge Component.
- Use the Write Costs algorithm step in the Calculate Cost algorithm or the Calculate Margin algorithm as an example to create the cost charge components.
- If you price a configurable item, then call the Rollup and Aggregate Charge Components subalgorithm to roll up and aggregate the costs.
- Save and publish your changes.

Calculate Margin

As an option, you can calculate margin according to costs from Oracle Costing.

The Calculate Cost pricing algorithm calculates cost, and the Calculate Margin step in the Calculate Pricing Charges algorithm calculates the margin. This section describes some guidelines you can follow for this use case.

Related Topics

- Get Price Details from Cost Accounting
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Get Cost Details from Oracle Costing
- Pricing Algorithms
- Overview of Using Business Rules With Order Management

Manage Price Lists That Have Rate Plans

Set up a rate plan in a price list so you can create and manage charges for each of the subscriptions that you have in Oracle Subscription Management.

Set up usage charges for your rate plans so you can bill your customers according to usage.

A usage charge is the rate that you pay according to how much you consume. The price that you pay for a subscription or coverage often has fixed charges and variable charges. For example, a service plan for home internet might charge you a one time $150 installation fee and a $100 monthly subscription fee.

You multiply the usage charge with the actual usage to determine the billable amount. If a subscription has a $100 monthly usage charge, then the yearly rate is $100 multiplied by 12 months, or $1,200.

The charge might also be variable. You might charge $1.00 for each minute of cell phone usage for zero to 300 minutes of usage during the month, and $0.50 for each minute over 300.

Other examples include utilities, phone charges, home maintenance, and so on. In some cases, you have to pay the monthly fee whether you use the service or not, or you might have to pay the same fee even though you use the service only 1 hour a week while your neighbor uses that same service 40 hours a week.

You can set up Oracle Pricing to price according to a usage charge, and that charge can vary depending on how much you use the subscription or service. To do this, you set up a *rate plan*, which is a collection of one time, recurring, and usage charges for a subscription.

The rate plan includes:

- Rate plan details, such as the name, description, and so on.
- A calculated price that includes the charge and charge components for each one time and each recurring charge.

You can:

- Associate your rate plan with a subscription item on the price list.
- Price a subscription according to a default rate plan.
- Apply tier adjustments or attribute adjustments.
- Create, update, or delete charges that are recurring or that happen one time in your rate plan.
- Assign as many rate plans as you need to the subscription item in the price list.

Pricing returns the usage charges that you use with the rate plan when it prices the subscription but it doesn't return any charge components. This behavior is different from calculating a one time charge or recurring charge because Pricing also returns the charge components for one time and recurring charges.

Try It

Assume you need to set up a rate plan for a telephone subscription. You change the rate according to how many minutes your customer uses.

1. Make sure you have these privileges:
    - Manage Price Lists (QP_MANAGE_PRICE_LISTS)
    - Manage In-Progress Price Lists (QP_MANAGE_IN_PROGRESS_PRICE_LISTS)
    - View Price Lists(QP_VIEW_PRICE_LISTS)
    - Approve Price Lists (QP_APPROVE_PRICE_LISTS)
2. Enable the features:
    - Go to the Setup and Maintenance work area, select the Sales offering, then click Change Feature Opt In.
    - Click the pencil in the row that has Subscriptions in the Name column.
    - Enable these features:
        - Rate Usage with Events
        - Integrate Order Management with Subscription Management to Process Subscriptions
3. Go to the Manage Pricing Charge Definitions task.
4. Create the usage charge:

Attribute	Value
Code	CALL_CHARGE

Attribute	Value
Name	Call Charge
Applies To	Price
Price Type	Usage You must use this value.
Charge Type	Call
Charge Subtype	Price
Usage UOM Class	Time Specify how you will measure usage. In this example, usage is according to minutes, so you use the Time class. A class contains a set of values. You use this attribute to specify the set of values that you want to allow for the usage.

For details, see *Manage Pricing Charge Definitions*.

5. Create the rate plan and specify the usage charges for your subscription. For details, see the Manage Rate Plans for Subscriptions section later in this topic.
6. View your charges:
    - Go to the Subscription Management work area, then search for your subscription number, such as CDRM_19012.
    - On the Subscription Number page, in the Products area, use the Pricing tab in the subscription to view details that describe the rate plan.
    - Examine details about the usage charge, pricing matrix, tired adjustments, and attribute adjustments.
    - In the List Price column, click **View Usage Charge**, then examine the charges.

For a detailed walk through with screen prints, go to *Technical Reference for Oracle Pricing (KB171412)*, then download the Set Up Pricing for Rate Plans attachment.

Manage Rate Plans for Subscriptions

Create and update the rate plan that you use for each subscription item. You can create and update the rate plan's one time charges, recurring charges, and usage charges.

Assume you need to manage a rate plan for the My Subscription item on the Corporate Segment Price List.

1. Go to **Home > Order Management > Pricing Administration > Price Lists**.
2. Search for and open Corporate Segment Price List for editing.

3. Search for this item:

Attribute	Value
Item	My Subscription

4. In the search results, click **Manage Rate Plans**.
5. On the Rate Plans page, click **Create Rate Plan**.
6. Enter this value, then click **Create**:

Attribute	Value
Name	Rate Plan for My Subscription

Note that you can use this rate plan only with a subscription that you manage in the Order Management work area. You can't use it with a subscription that you import.

7. Specify one-time charges, recurring charges, usage charges, and so on.

Click *here* for a handy demonstration that walks you through the set up. It starts at 01:50 in the presentation.

Specify Usage Charges According to Attribute Values

You can use the Pricing Administration work area to set up a rule on a rate plan that uses your customer's attributes, the subscription's attributes, or the sales order's attributes to determine the base price for the subscription's usage charge:

- Specify a wide range of attributes from more than one source to determine the base price on your rate plan.
- Specify these attributes when you can't use only the usage quantity to determine the base price.

Click *here* for a demonstration. It starts at 2:30 in the presentation.

You can also use the Manage Price Lists REST API to import these details instead of using the Pricing Administration work area. For example payloads, go to *Technical Reference for Oracle Pricing (KB171412)*, then download the Integrate Subscriptions attachment.

Tier You Adjustments on Rate Plan Charges

You can tier an adjustment on a rate plan charge. For example, assume you want to vary the discount according to quantity:

1. Enable these features:
    - Rate Usage with Events
    - Integrate Order Management with Subscriptions Management to Process Subscriptions
2. Create a recurring charge on a discount list:

Attribute	Value
Charge Definition	Monthly Fee

Attribute	Value
Calculation Method	Price
Base Price	20
Frequency	Month

3. Enable the Enable Tiered Adjustment option.
4. Set these values:

Attribute	Value
Tier Basis Type	Item Quantity
Aggregate	On Document
Apply Adjustment To	Highest Tier
Calculate	Per Unit

5. Add your tiers:

More Than	Less Than or Equal To	Adjustment Type	Value
10	-	Discount Amount	10
0	10	Discount Amount	5

Leave the Adjustment Basis attribute empty.

Note

- Pricing will keep your set ups even if you disable the Tiered Adjustment section.
- If the charge on your rate plan has a tiered adjustment but the Tiered Adjustment section is disabled, then Pricing will display a warning that it will delete the adjustment when you attempt to save it. You can delete it or cancel the save.
- Pricing constrains the list of values in the Tier Basis Type attribute depending on whether you specify the adjustment for a one time, recurring, or usage charge.
- If you set up a tier according to amounts, then make sure you use the Manage Pricing Bases task to set up the tier basis. For details, see *Manage Pricing Bases*.
- You can create and modify an SCM Journey to support tasks on the Rate Plans, Rate Plan Details, and Rate Plan Charge details pages. Use the journey to help you take the appropriate actions when you manage the rate plan. You can also browse or search popular journey resources to help you negotiate a new rate plan in Oracle Order Management.
- You can also use business rules in Oracle Visual Builder Studio to modify the Rate Plan Details and Rate Plan Charge Details pages. See *Extending Redwood Applications for HCM and SCM Using Visual Builder Studio*.

Guidelines

- You must already have a rate plan to add a usage charge to your subscription.
- The charge period determines how often Pricing and Subscription Management add up the usages for each usage charge. You must specify a time UOM, such as Day, when you set the UOM attribute for the charge period. Pricing uses the UOM class that you set for the RCS_DEFAULT_UOM_CLASS_CODE_FOR_SVC_DURATION profile option to validate the charge period that you set. For details about this profile option, see *Integrate Order Management with Subscription Management*.
- For details about the attributes that you can use for a rate plan's charge, including the charge period's UOM, go to *REST API for Oracle Supply Chain Management Cloud*, then expand **Order Management > Price Lists > Price List Items > Rate Plans > Rate Plan Charges**.

Different Ways That You Can Apply Usage Charges

- Apply a simple usage charge.
- Create a condition in a rate table that specifies the usage charge.
- Apply an attribute adjustment to the usage charge.
- Apply a tiered adjustment to the usage charge. You must use the Rate Plans REST API to apply a tiered adjustment.

Pricing Administration Work Area

Note these guidelines when you use the Pricing Administration work area to specify your usage charge:

- To use the Manage Rate Plans button, you must first add your subscription to the price list, then click **Save**.
- If you change the value in the Calculation Method attribute from Price to Pricing Matrix for an existing usage charge, and if you don't click **Add Columns** and add at least one column, then Pricing Administration will display only the required columns. You won't see any optional columns.
- The dates that you see for the rate plan's charges come from the rate plan. However, if you use the Pricing Administration work area to create the charges, then the end date for each charge will be empty. Note that the start date and end dates aren't visible on the rate plan's charge.
- You can use the Rate Plan page in Pricing Administration to add up to 500 rows to your pricing matrix. Each row represents one rule. If you need more than 500, then you must import them through REST API.
- The Rate Plan page doesn't display a column's default value even if the matrix class specifies a default value. Instead, you must manually enter the value. For background, see *Pricing Matrix Class*.

Import

You can import:

- One time and recurring charges
- Charge rules that apply usage charges according to how much you consume
- Manual price adjustments, including periodic adjustments

If you price a subscription before you import it, then your import usually includes the calculated price and the usage charges. The calculated price usually includes one time charges, recurring charges, adjustments, and discounts. The usage charge includes the charges that you will bill according to the actual amount that your customer consumes. Oracle Pricing uses a rate plan to capture these pricing details when you import.

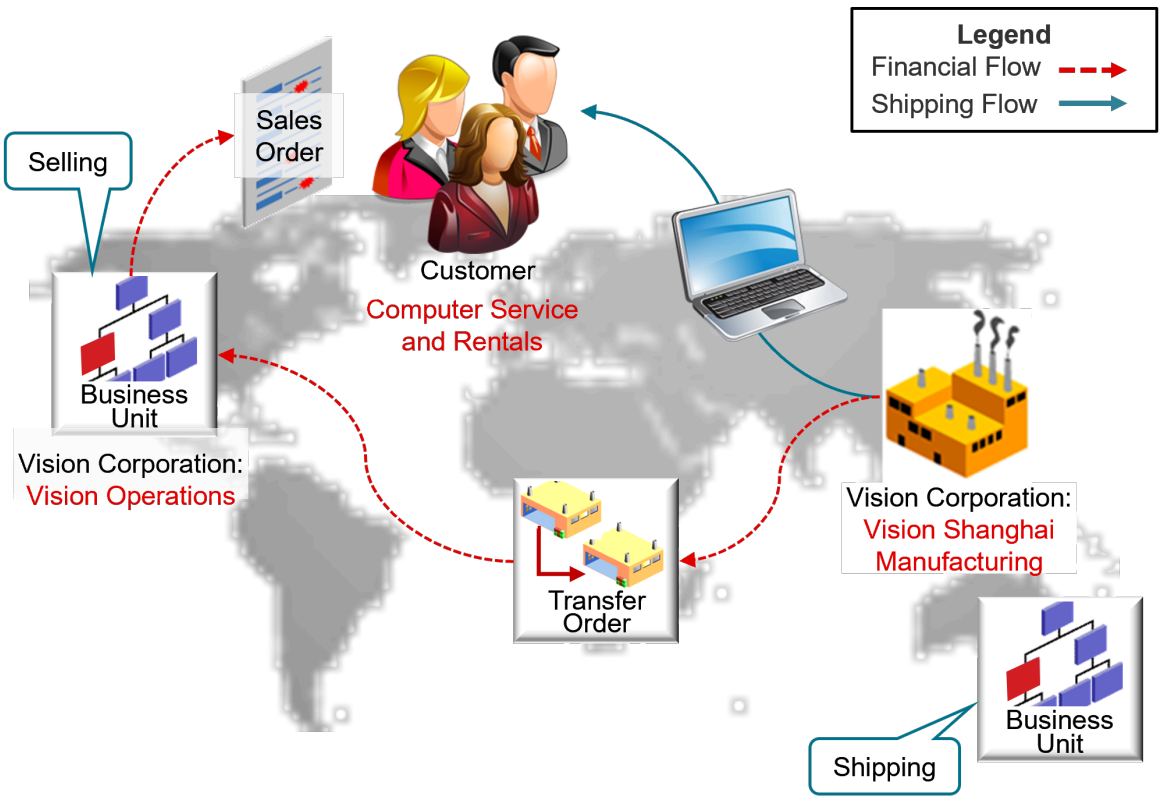
For more about import, see *Use Rate Plans with Your Subscriptions*.

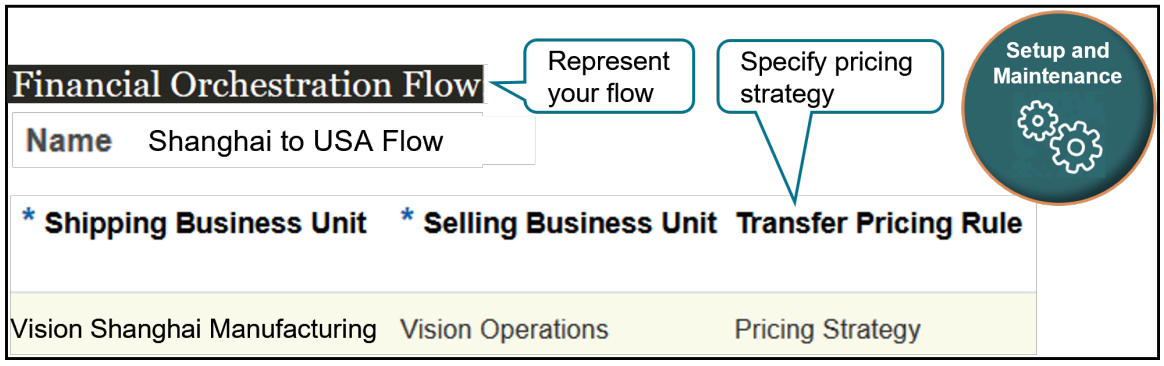
More

- *Use Rate Plans with Your Subscriptions*
- *Negotiate Rate Plans in Sales Order Currency*

Use Financial Orchestration and Oracle Pricing to Price Internal Transfers

Learn how to use Financial Orchestration and Oracle Pricing to price an item that you transfer between organizations that are part of the same company.



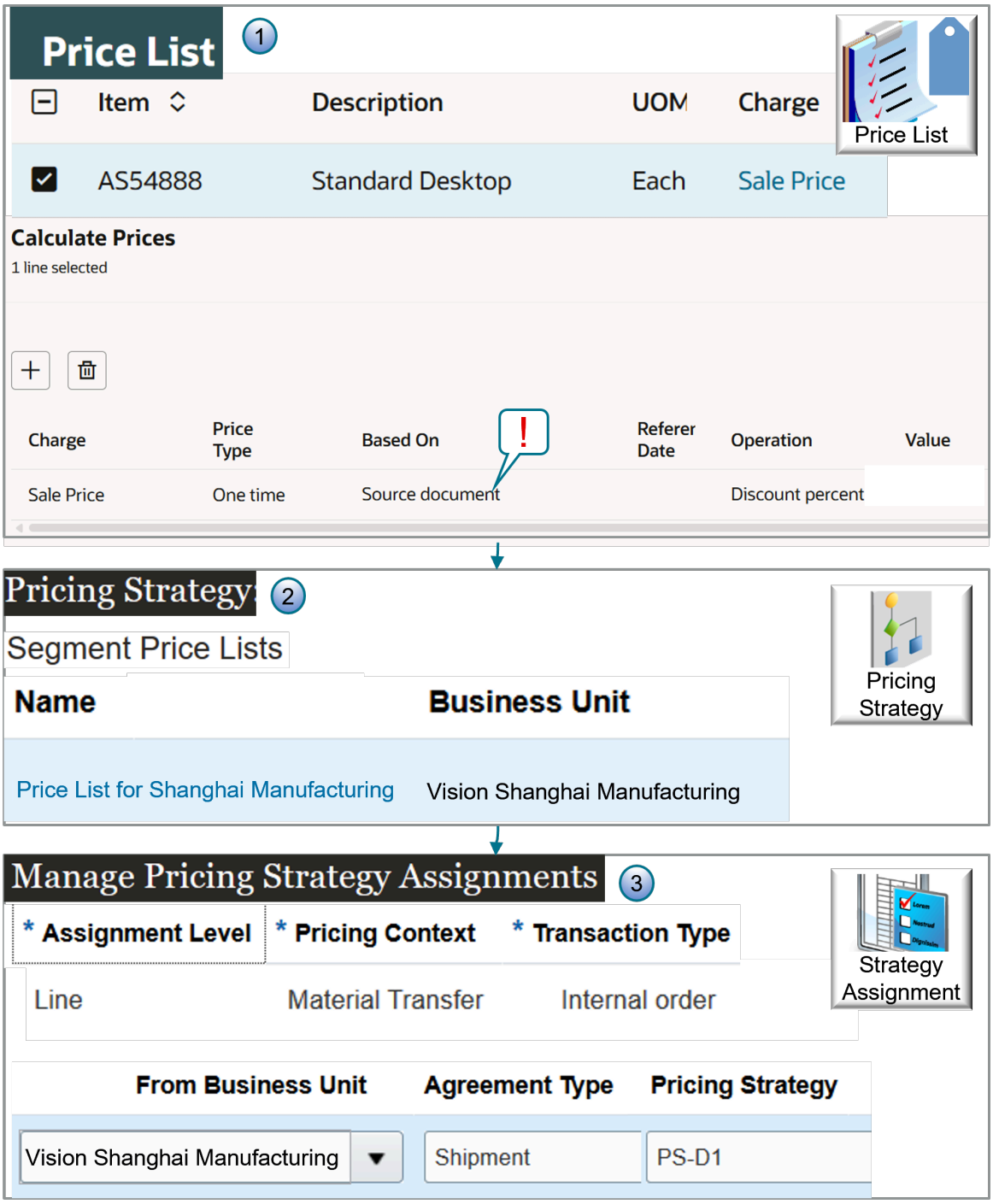


Note

- Use Financial Orchestration to represent your financial flow between the shipping business unit and the selling business unit.
- Use the Transfer Pricing Rule attribute on your flow and specify to use a pricing strategy to set the price for your financial flow. For example, if the shipping business unit is Vision Shanghai Manufacturing, and if the selling business unit is Vision Operations, then use a pricing strategy to automatically price an item that flows from Shanghai Manufacturing in China to Vision Operations in the United States. The flow will call Oracle Pricing to price the item at run time.
- Pricing will include any price lists, discount lists, cost lists, and shipping charge lists that you add to the strategy.
- For example, if you want to price according to cost, add a cost list, not a price list. If you need to calculate the margin, then use Cost Plus Pricing instead. For details, see *Manage Cost Lists* and *Cost Plus Pricing*.

- Mark up or mark down the price when you add the price list.
- Price standard items or configured items.

You do the rest of your setup in Pricing Administration:



Note

1. Create a price list specifically for your Shanghai factory. Use the price list to set charges for each item that Shanghai builds. For example, add the AS54888 to the list, then set the values, such as set the Calculation Method to Source Document Cost, and add a 10% discount.

2. Create a pricing strategy specifically for your Shanghai factory. Add the price list to the strategy.
3. Assign the pricing strategy for each order line that includes a transfer order.

   Create a condition.

   ```
   If you receive a shipment from Vision Shanghai Manufacturing, then assign the PS-D1 pricing strategy.
   ```

Modify Predefined Objects to Meet Your Requirements

Use Pricing Administration to modify predefined objects and meet your specific requirements:

Predefined Object	Description
Price Material Transfers pricing algorithm	Processes a financial flow that includes a material transfer. It uses the source document that you specify when you set up the financial flow.
Line Pricing Strategy Assignment matrix class	Specify conditions and results, such as how to identify the To Business Unit, From Business Unit, Agreement Type, and so on.
MaterialTransfer service mapping	Specify entities, attributes and their values during a financial orchestration flow that involves pricing.

Summary of the Setup

1. Assign the pricing strategy.
2. Create the price list.
3. Set up the financial flow.
4. Test your set up.

Here's your scenario for the example in this topic:

- Assume you're the supply chain controller for an organization that builds items in Shanghai, but most of your customers are in the United States. You ship the AS54888 Desktop Computer directly from the factory in Shanghai to your customer in the United States.
- You need to set up a financial flow that orchestrates the intercompany transaction between the shipping business unit and the selling business unit. You capture the financial transaction that happens between the Vision Shanghai Manufacturing business unit in Shanghai and the Vision Operations business unit in the United States.
- You need a rule that adds a 10% discount off the selling price that Shanghai charges.
- Assume you already have a pricing strategy for Shanghai. You will assign it, create a price list and financial flow, then test your set up.

For more about:

- Learn about financial orchestration. For details, see *Overview of Supply Chain Financial Orchestration*.
- Examine a similar example for drop ship. For details, see *Set Up Financial Flows for Drop Ship*.

1. Assign the Pricing Strategy

Set up a rule that assigns all shipments that come from the Shanghai business unit to your pricing strategy:

1. Go to the Pricing Administration work area.
2. Click **Tasks > Manage Pricing Strategy Assignments**.
3. On the Manage Pricing Strategy Assignments page, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Assignment Level	Line
Pricing Context	Material Transfer
Transaction Type	Internal Order In this example, Vision Corporation is the organization. This transaction happens between two different business units that are part of Vision Corporation. A sales order that stays in the same organization is an internal order, so we will use this rule to assign the strategy only for internal orders.

4. Click **Create Assignment Matrix**.
5. In the Create Assignment Matrix dialog, make sure these options have a check mark, then click **OK**:
    - From Business Unit
    - Agreement Type
6. In the Pricing Strategy Assignment Rules area, click **Actions > Add Row**, then set the values:

Attribute	Value
From Business Unit	Shanghai Manufacturing
Agreement Type	Shipment
Pricing Strategy	PS-D1 Assume you already created a pricing strategy named PS-D1.

7. Click **Save and Close**.

2. Create the Price List

Create the price list that you will use as part of the financial flow:

1. Click **Price Lists**.
2. On the Price Lists page, click **Actions > Create**.

3. In the Create Price List dialog, set the values, then click **Save and Edit**:

Attribute	Value
Name	Price List for Shanghai Manufacturing
Type	Segment Price List
Currency	USD
Business Unit	Vision Operations

4. On the Price List page, click **Price List Lines**.
5. In the search results area, click **Actions > Add Row**, then set the values:

Attribute	Value
Item	AS54888
Pricing UOM	Each
Line Type	Buy

6. Click **Create Charge**.
7. In the Charge area, set the values, then click **Save**:

Attribute	Value
Pricing Charge Definition	Sale Price
Calculation Method	Source Document Cost In this example, the sales order is the source document, so this value says that we're going to get values from the sales order when we calculate the charge.
Calculation Type	Discount Percent
Cost Calculation Amount	10% Here we are saying to discount the charge by 10% when we calculate the cost for the AS54888 from the sales order.

Attribute	Value

8. Click **Save and Close**.

3. Set Up the Financial Flow

1. Make sure you have the privileges that you need to administer Financial Orchestration.
2. Go to the Financial Orchestration work area.
3. Click **Tasks > Manage Financial Orchestration Flows**.
4. On the Manage Financial Orchestration Flows page, click **Actions > Create**.
5. In the Create Financial Orchestration Flow dialog, set the values, then click **OK**:

Attribute	Value
Name	Shanghai to USA Flow
Business Process Type	Shipment
Priority	1
Separate Primary and Financial Route	Doesn't contain a check mark.

6. In the Primary and Financial Routes area, click **Actions > Add Row**, then set the values:

Attribute	Value
Shipping Business Unit	Shanghai Manufacturing
Selling Business Unit	Vision Operations
Selling Trade Organization	Vision Operations
Transfer Pricing Rule	Pricing Strategy This value instructs Financial Orchestration to call Oracle Pricing, and to use the pricing strategy that you set up in the Pricing Administration work area.
Documentation and Accounting Rule	Standard
Receivables Invoice Type	Intercompany

Attribute	Value
Receivables Credit Memo Type	Credit Memo
Payment Terms	50/50 Select whatever value that meets your needs.

Leave the other attributes empty.

7. Set the value, then click **Save and Close**:

Attribute	Value
Status	Active

4. 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 the AS54888 to an order line.
3. Set this value: then save:

Attribute	Value
Warehouse	Shanghai Manufacturing

4. Click **Additional Order Details > Pricing and Currency**.
5. Verify the value:

Attribute	Value
Strategy Name	PS-D1

6. On the order line, in the Amount column, click the value.
7. Verify that the Line dialog contains the values:

Attribute	Value
Base List Price Applied from Price List for Shanghai Manufacturing	500
Discount Applied from Price List for Shanghai Manufacturing	50
List Price	450
Your Price	450

Related Topics

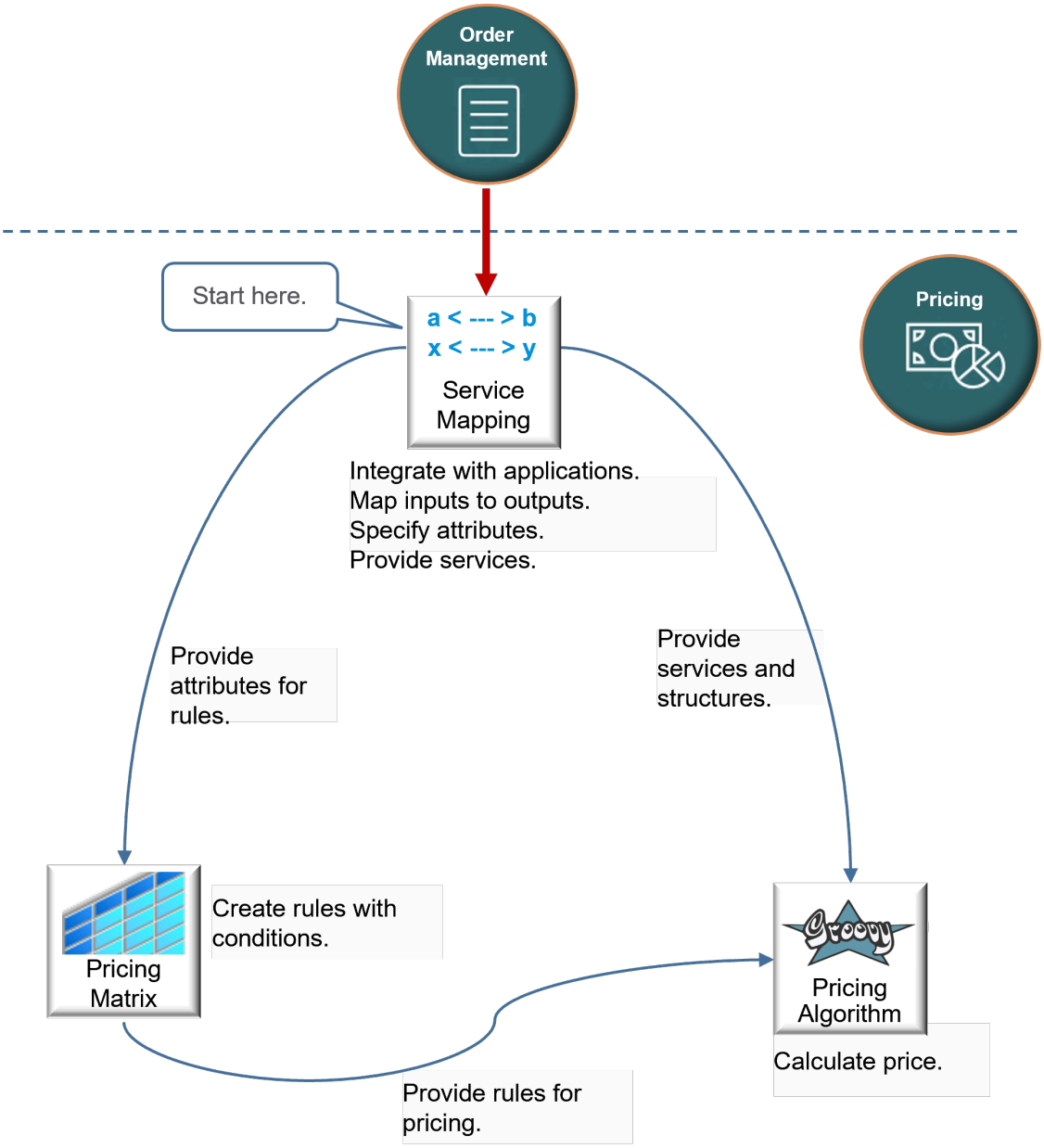
- Manage Cost Lists
- Cost Plus Pricing
- Overview of Supply Chain Financial Orchestration
- Set Up Financial Flows for Drop Ship

9 Service Mappings and Pricing Algorithms

Mappings

How Service Mappings, Pricing Algorithms, and Matrixes Work Together

Set up a service mapping, pricing algorithm, and matrix class so they work together to price an item.



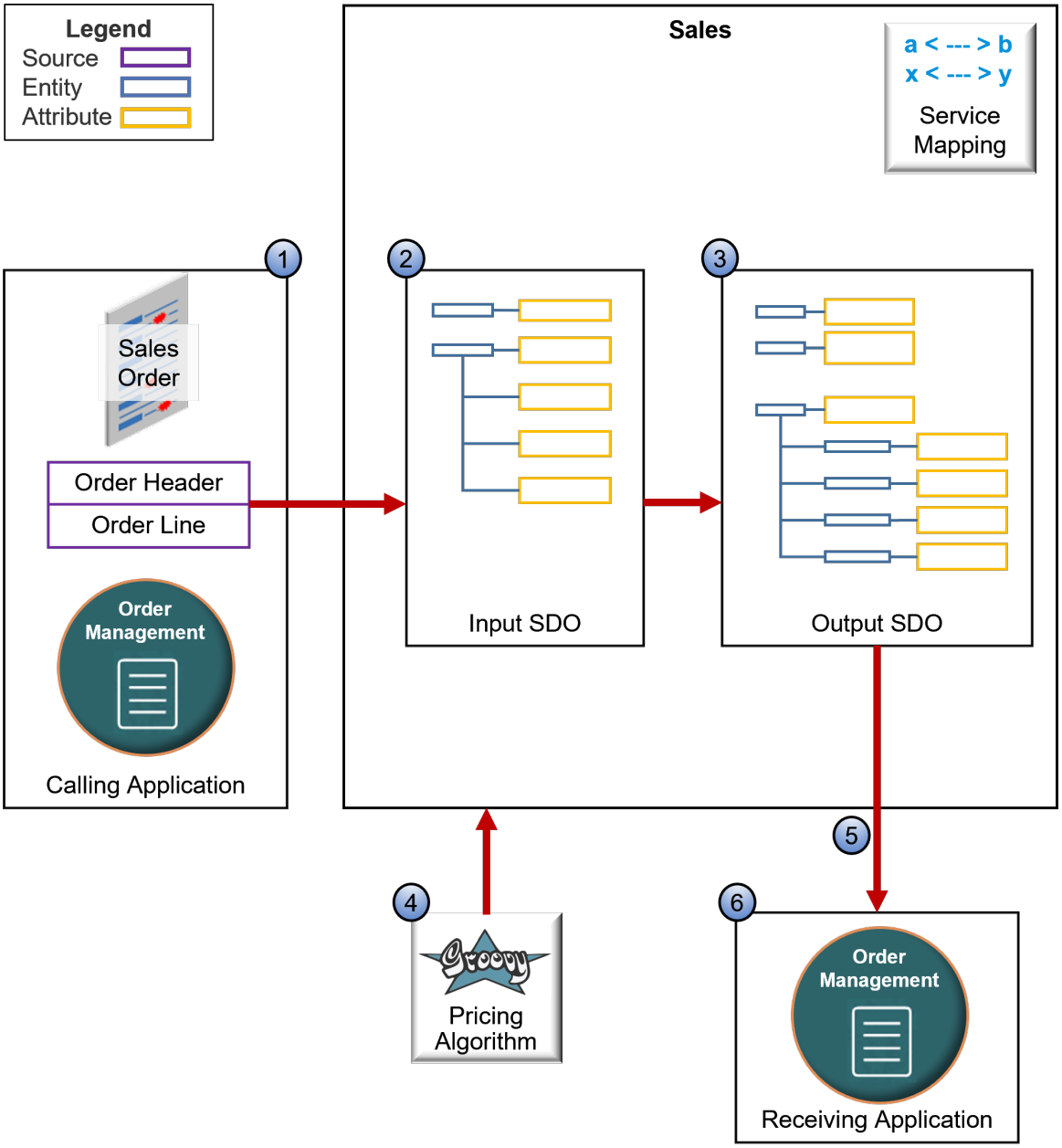
The pricing algorithm references a service mapping to price an item that it receives from an application, such as Oracle Order Management:

Object	Description
Service mapping	Integrate with the calling application that requires pricing, such as Order Management. Map inputs from the calling application to outputs that the matrix and algorithm can use. Set up attributes so rules in the pricing matrix can use them. Set up services and pricing entities to provide structure for variables that the pricing algorithm uses.

Object	Description
Pricing matrix	Set up conditions and dimensions in the pricing matrix. Choose attributes in the condition and result columns of the matrix class from the entity attributes you set up in the service mapping. The algorithm uses condition dimensions and their values to compare the runtime value against the value of the design time rule. The result values are the result of evaluating the conditions.
Pricing algorithm	Set up the algorithm so it uses the pricing matrix to evaluate and apply pricing rules at runtime. Set up the algorithm to transform the attribute values it receives from the service mapping to output values.

Example Flow

Assume Order Management is the calling application and the receiving application, and it requires Oracle Pricing to price a sales order:



How it works:

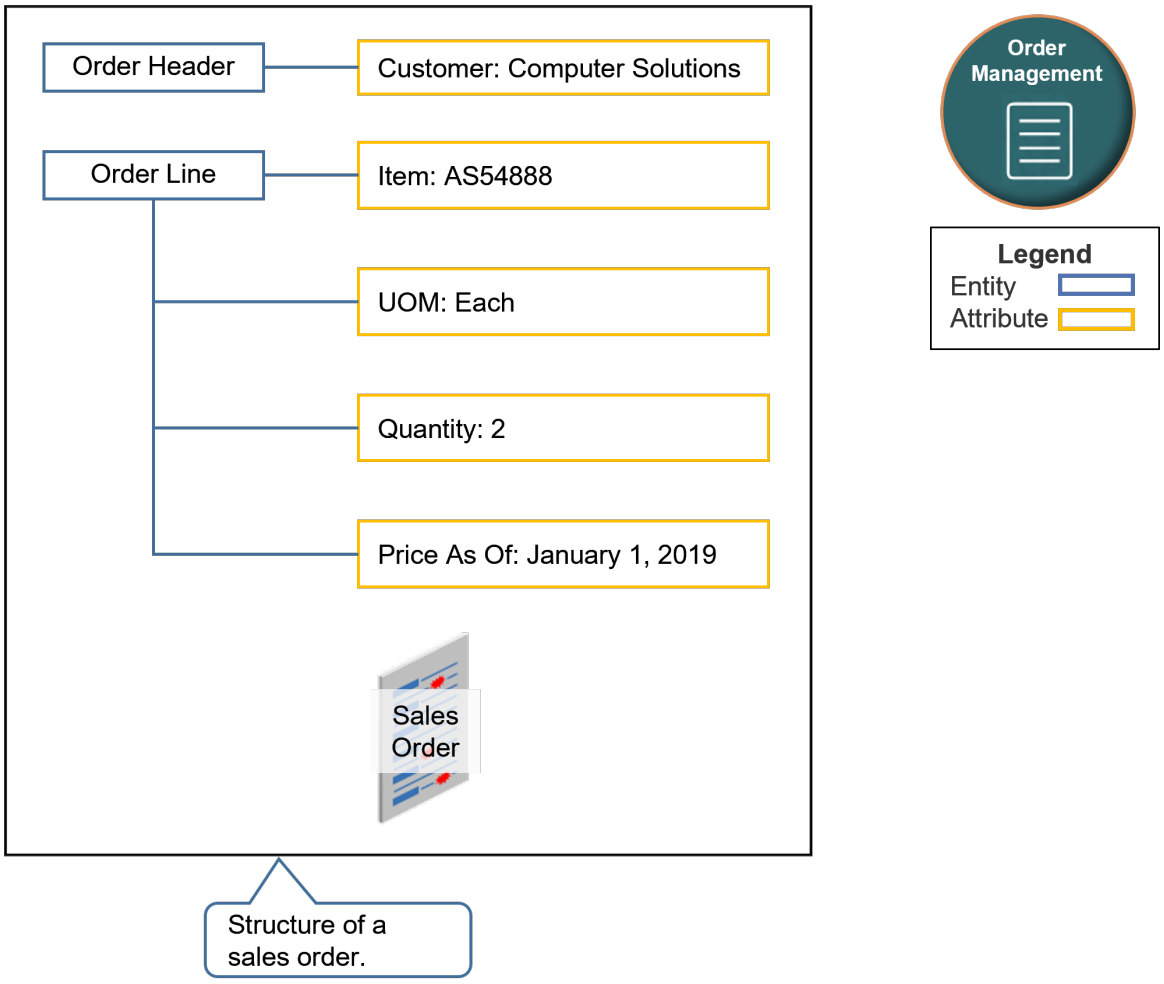
1. Order Management uses the predefined CalculateSalesOrderTotals service to send data for one sales order to Pricing.
2. CalculateSalesOrderTotals references the predefined Sales service mapping. This service mapping includes the OrderTotal source, and it uses OrderTotal to convert the data that Order Management sends into an input service data object (SDO).

   An SDO is a data structure that doesn't conform to a single programming language. It helps Pricing communicate with Order Management and other applications, such as Contract Management. It allows one service to call another service without having to use the same computer language.

   A service data object typically includes elements arranged in a tree structure that includes a root and branches, and it allows the calling service to access the data that the elements reference.

The OrderTotal source specifies the entities and attributes that the SDO contains and that the service will use as input to the pricing application.

For example, here's the `input sdo` that represents the structure of a sales order:



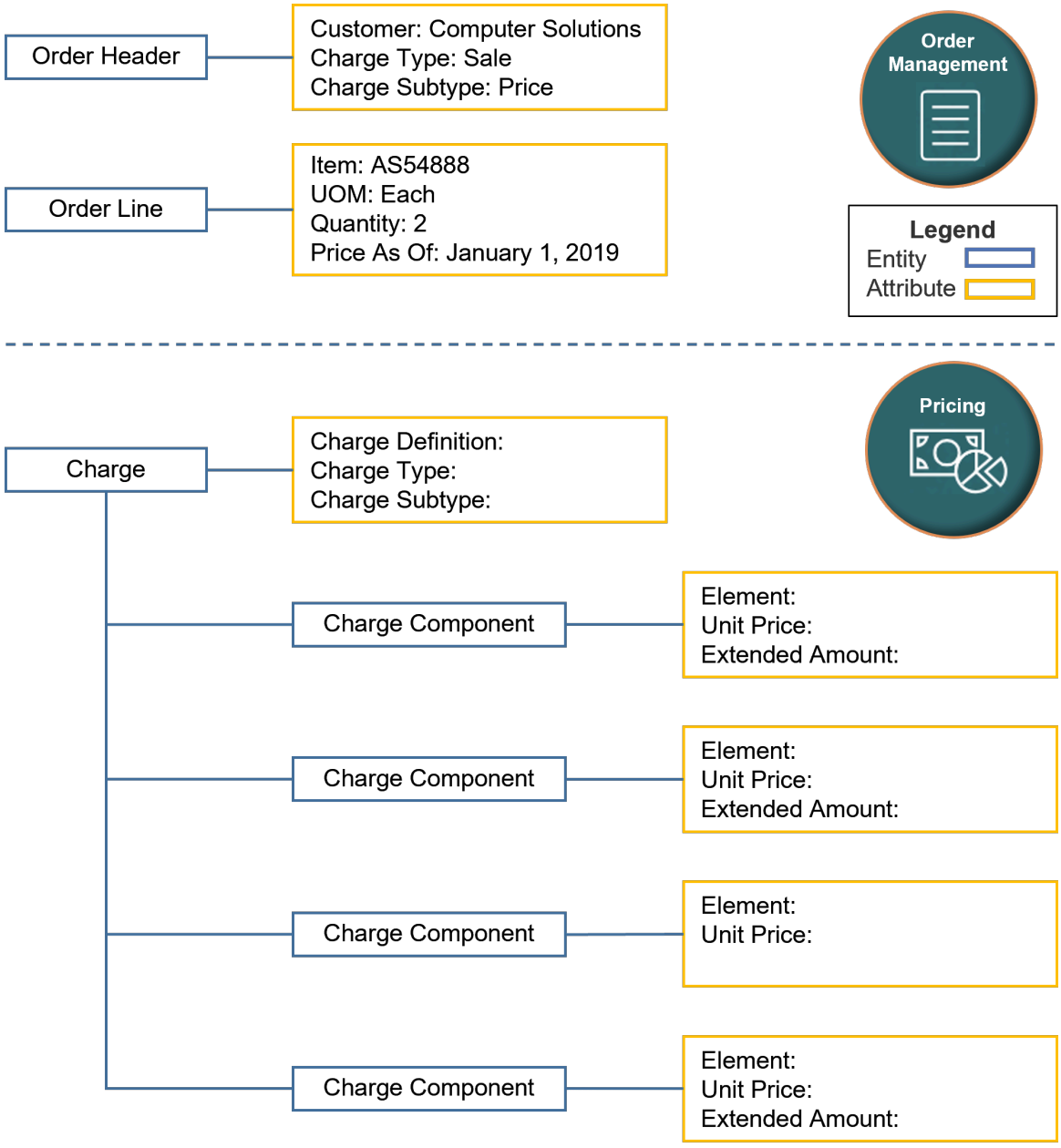
The OrderTotal source specifies to use:

- The Header entity and the Customer attribute on the Header.
- The Order Line entity and various attribute on the line, such as Item, UOM, and so on.

3. The Sales service mapping specifies sources, entities, and mappings between entities and attributes so Pricing can create the structure for the output SDO that it will send to Order Management.

   It adds the Charge entity and the ChargeComponent entity to the output SDO, but doesn't add values for the attributes that these entities contain.

   Here's the output SDO without pricing data:

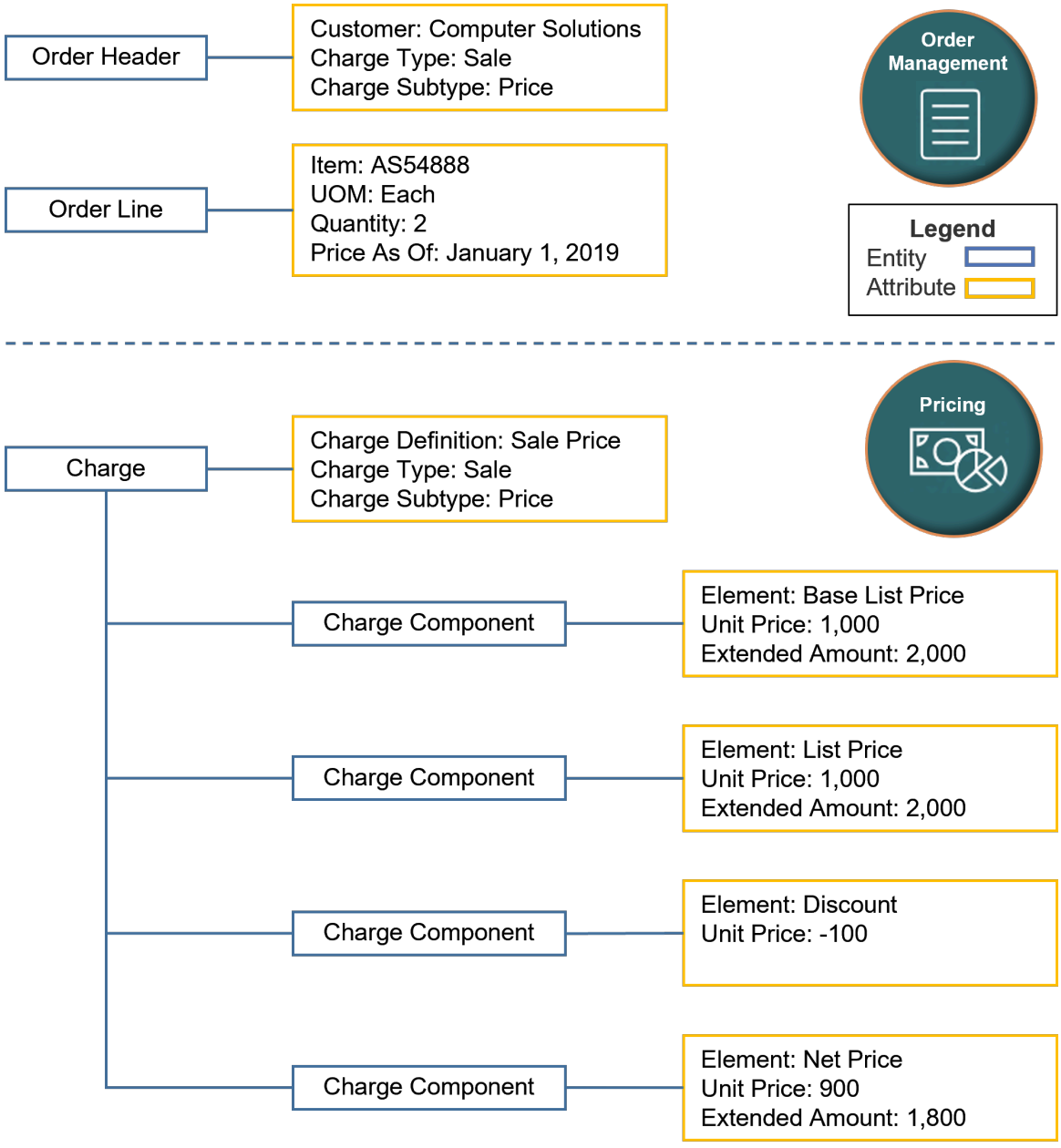


4. Pricing uses a pricing process assignment to identify the pricing algorithm to run, then applies the pricing algorithm to the service mapping. In this example, Pricing applies each step of the predefined Price Sales Transaction pricing algorithm to the Sales service mapping.

   Each algorithm step uses data from the output SDO and conditionally updates it in memory at runtime.

   In this example, the pricing algorithm adds values to attributes in the Charge entity and the ChargeComponent entity of the output SDO. It adds a value of $1,000 to the Unit Price attribute.

   Here's the output SDO with pricing data:



   Pricing does the calculation:

- **Base List Price**. Unit price of 1,000 multiplied by Quantity of 2 equals Extended Amount of 2,000.

  Net Price. Unit Price from Base List Price of 1,000 minus Discount of 100 equals Unit Price of 900. Unit Price of 900 multiplied by Quantity of 2 equals Extended Amount of 1800.

5. Pricing sends the output SDO to Order Management.
6. Order Management receives the output SDO and uses it to update entities in the sales order, such as Extended Amount.

Note that Pricing might use different services, sources, and algorithms for other examples.

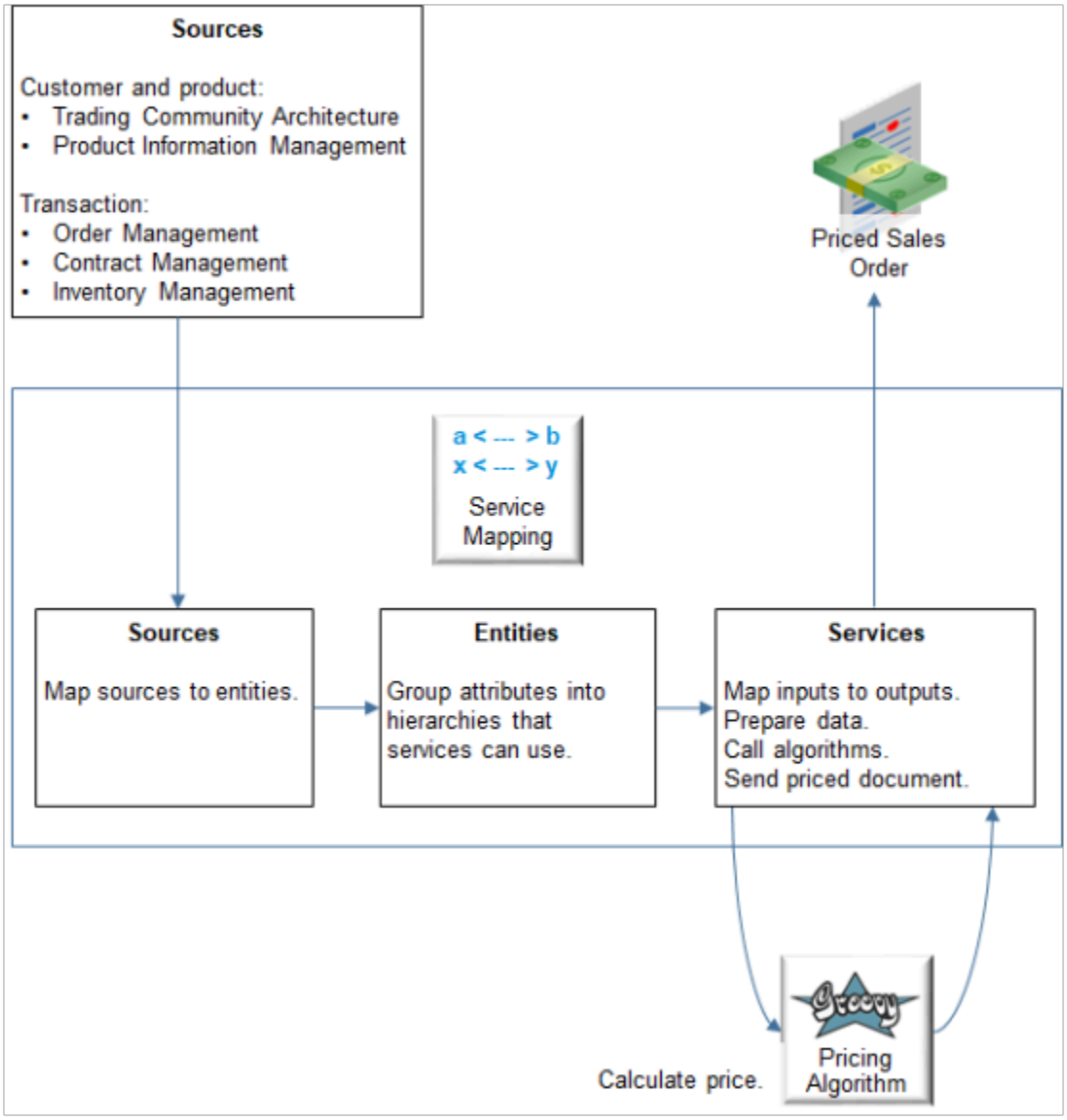
Related Topics

- Assign Pricing Operations to Pricing Algorithms
- How Profiles, Segments, and Strategies Work Together
- Pricing Algorithms
- Service Mapping

Service Mapping

Use a service mapping to integrate Oracle Pricing with inputs to pricing, objects in Pricing, and outputs from Pricing.

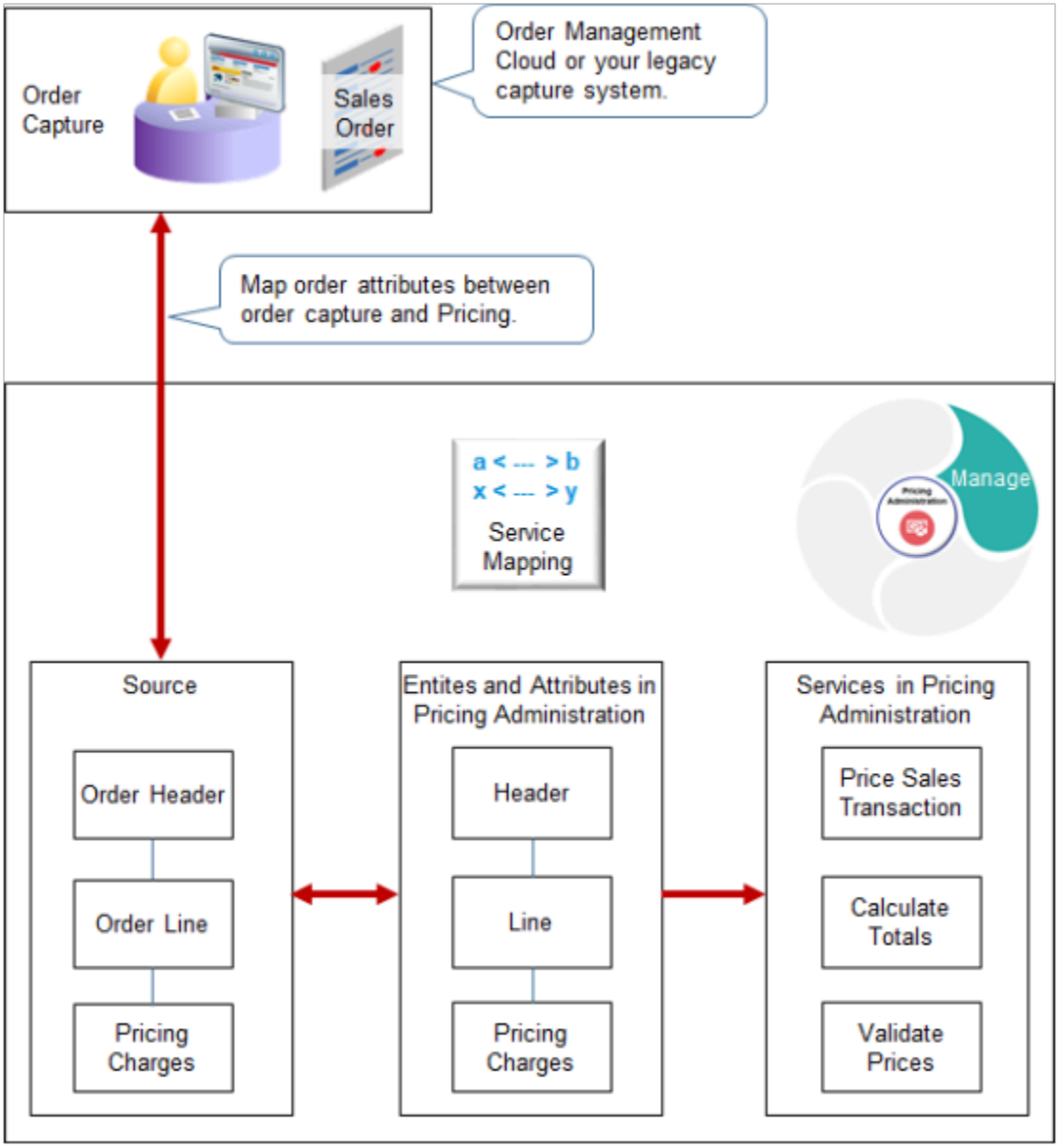
A service mapping is a map that creates relationships between services, sources, entities, and attributes. It specifies a group of pricing entities and attributes that a service can retrieve and update:



Note

- Map sources to entities that Pricing can use to price the document. Sources include your customer data, such as from Trading Community Architecture, item data, such as from Product Management, and data about the transaction that Pricing must price, such as a sales order from Oracle Order Management.
- Use entities to group attributes into hierarchies that a service can retrieve and update.
- Use services to map inputs to outputs, prepare data for algorithms, call algorithms, then communicate the priced document to the application that requires pricing, such as Order Management for a sales order.

Here's an example:



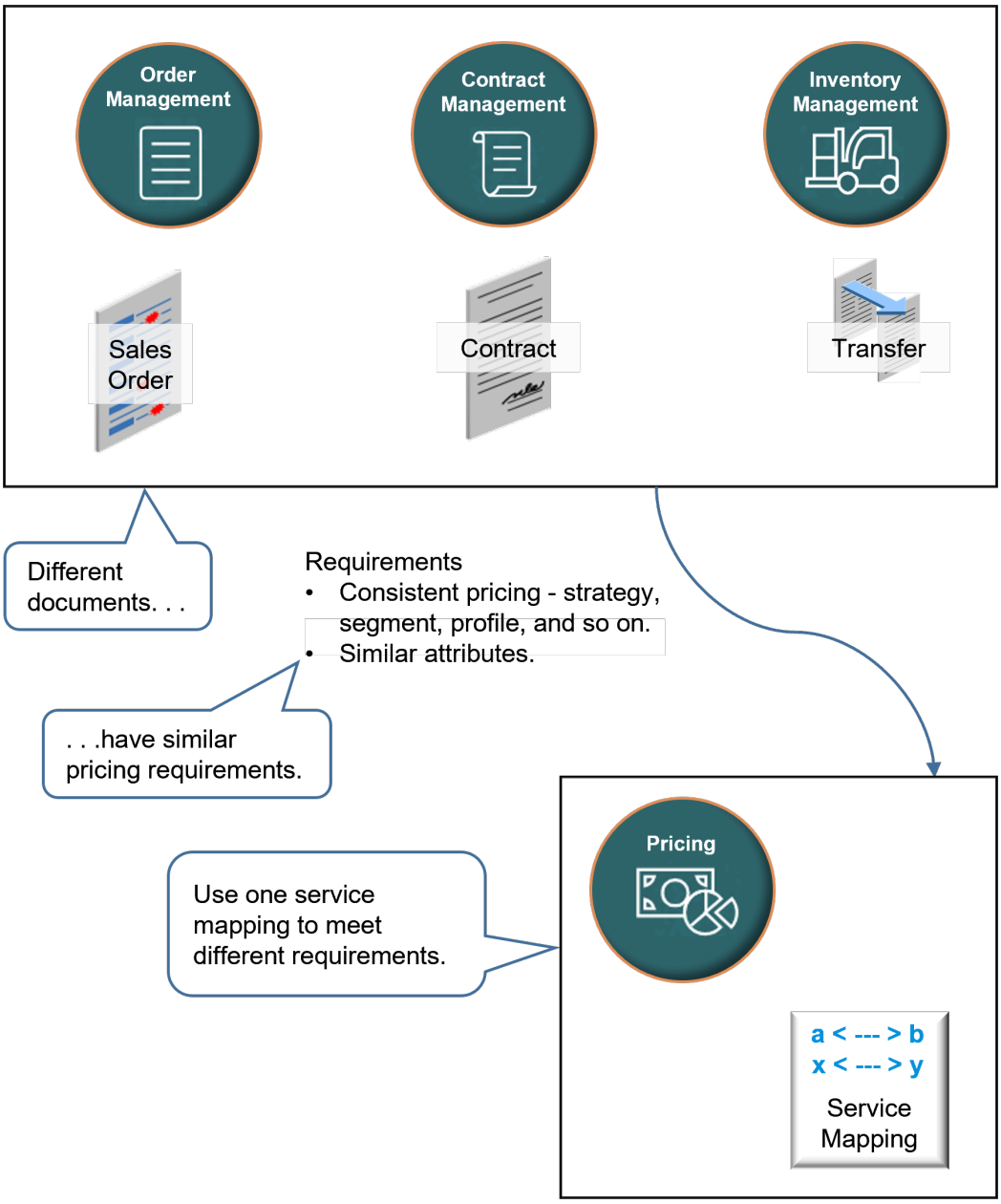
Note

- Map entities and attributes in a declarative environment without having to write software code.
- Use an order capture system as the source, such as Order Management or your legacy capture system.
- Map your attributes. For example, assume you sell professional services and your legacy system refers to your customers as clients. Map the Client attribute in your legacy system to the Customer attribute in Pricing.
- Extend entities to add your own behavior.

  For example, Pricing comes predefined to map all attributes in a sales order. You can modify the service mapping to map only some attributes. Add your own attribute to use in price calculations. For example, add a descriptive flexfield in Order Management that stores details about your own attribute, then use a service mapping to map it into Pricing.

Represent Different Types of Documents

Pricing receives requests to price documents from different applications:



For example:

- Price a sales order from Order Management.
- Price a contract from Contract Management.
- Price a transfer order from Inventory Management.

Documents have similar characteristics:

- They have similar pricing requirements. For example, they all need to calculate price according to pricing strategy, pricing segment, customer profile, and so on.

- They need Pricing to calculate price consistently.

  For example, assume you create a sales order for the AS54888 desktop computer for customer Computer Service and Rentals in Order Management. Sometime later, you create a service contract for the AS54888 in Contract Management. Pricing needs to apply the strategy, segment, and discounts in the same for Contract Management that it does for Order Management.

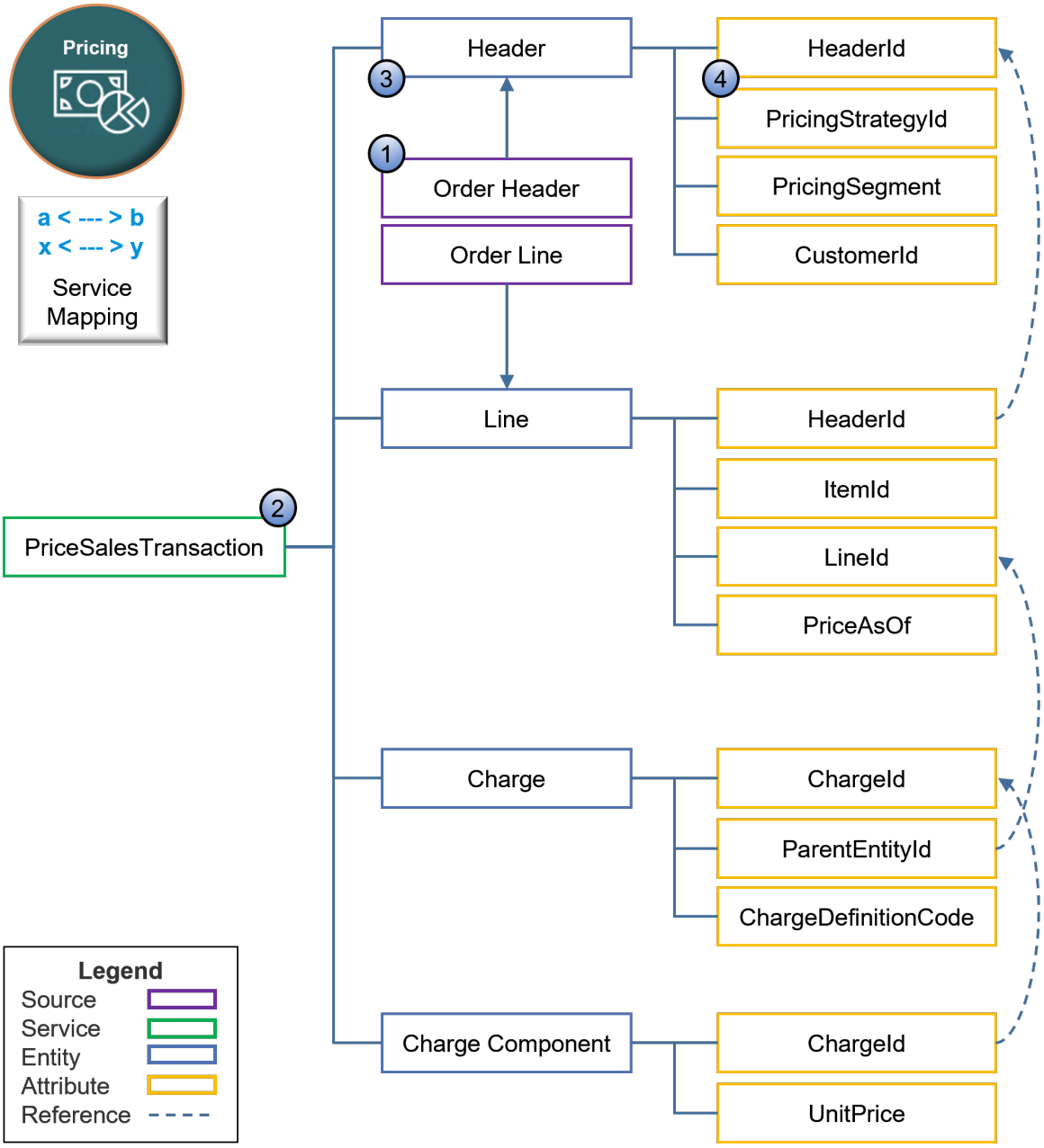
- They have attributes that exhibit similar characteristics. Use a service mapping to represent them with a single attribute in Pricing. This helps to simplify your pricing set up and maintenance.

  For example, assume Order Management, Contract Management, and Inventory Management each include 10 attributes there are similar to one another, resulting in 30 attributes that Pricing must process or represent. You can map all 30 into 10 attributes in a service mapping instead of representing and processing 30 attributes.

Map Inputs to Outputs

A service mapping contains entities, sources, and services. Pricing uses it to map the input SDO to the output SDO.

Here are some of the main parts of the predefined Sales service mapping. Pricing uses it to price a sales order:



You can specify objects in a service mapping:

Object	Description
1. Source	The source is a mapping that specifies the relationship between the source and entities. It provides a structure so Pricing can model data in the input SDO. For example, the predefined Sales service mapping includes the OrderTotal source, and this source includes details about the Order Header and Order Line: • Maps between the OrderLine source and the Header entity • Maps between the OrderLine source and the Line entity

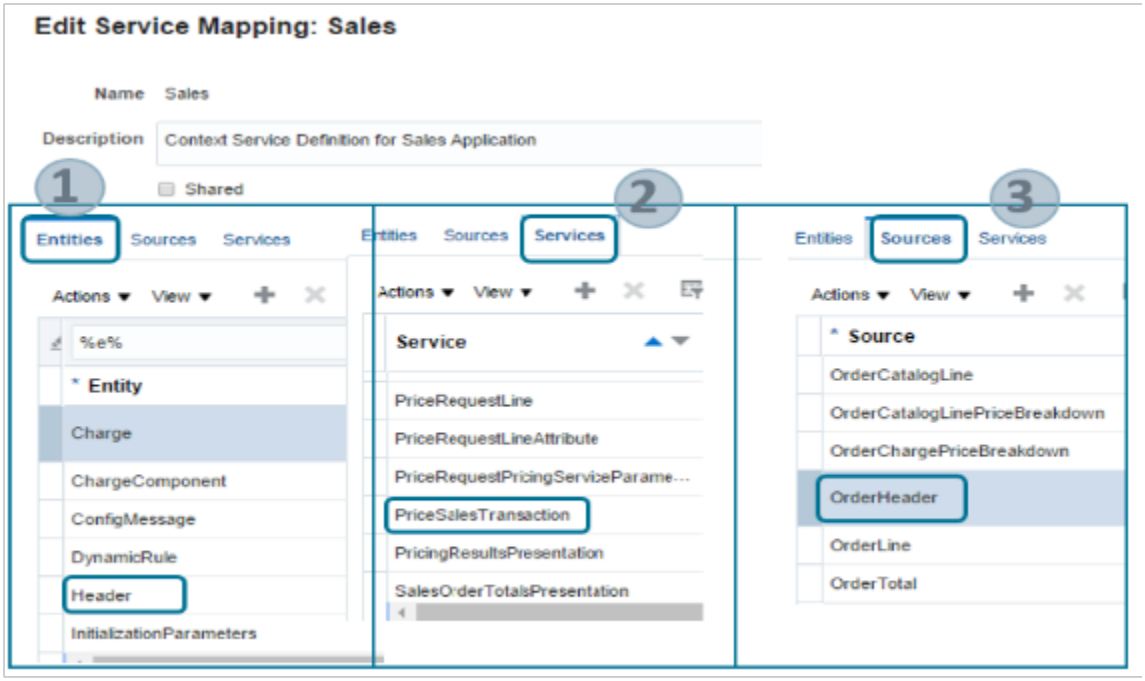
Object	Description
2. Service	Pricing uses the PriceSalesTransaction service to price a sales order. Use the service to: • Reference the Java method or pricing algorithm that calculates price. • Map inputs to outputs. For example, the PriceSalesTransaction service references the Line entity, and it allows read and write actions on the Line entity. A service mapping service isn't a web service. You can use the predefined services for most implementations.
3. Entity	Reference the entities that the service mapping requires to structure the output SDO. In this example, the output SDO includes four entities. Each entity references one or more attributes. For example, here are the attributes that the Header entity references: • **PricingStrategyId.** Allows the service mapping to map the input SDO to the pricing strategy on the output SDO. • **PricingSegmentCode and CustomerId.** Allows the service mapping to map the input SDO to the pricing segment on the output SDO. Here are the properties you can set for an entity on a service: • Entity. Select the entity you require from the list. If you need a new one, then click **Actions > Add Row**. Enter a unique name, and use camel capitalization, such as MyEntityName. • Alias. Don't use an alias. Leave it empty. • Read. Enable this option when you use this entity as an input to pricing. Pricing will populate it on the input SDO. For example, Header.CustomerId is a read-only entity, so enable Read but don't enable Write. • Write. Enable this option when you use this entity as an output from pricing. Pricing will populate it on the output SDO. For example, you use the Charge entity to read and write, so enable Read and enable Write.
4. Attribute	Reference the attributes that the service mapping requires to structure the output SDO. Each service mapping fulfills a finite purpose. For example, the Sales service mapping prices a sales order, so it references attributes that a typical sales order includes, such as ItemId so it can identify the item in the sales order, and UnitPrice so it can multiply unit price by quantity, and do other calculations. Here are the properties you can set for an attribute on an entity on a service: • Attribute. Select the attribute you require from the list. The list displays all attributes that the entity contains. • Alias. Don't use an alias. Leave it empty. • Read. Enable this option when you use this attribute as an input to pricing. Pricing will populate it on the input SDO. • Write. Enable this option when you use this attribute as an output from pricing. Pricing will populate it on the output SDO.

Object	Description
	Read and Write on the attribute are independent of Read and Write on the entity. For example, if you enable Read and disable Write on entity y, but enable Read and Write on attribute x, then Pricing will populate attribute x on the input and the output SDO.

Other service mappings might reference an entirely different set of attributes. For example, Order Management uses the SoldToPartyId attribute to identify the customer, but Contract Management uses PartyId. You can use a single attribute in the service mapping, such as HeaderId, to represent SoldToPartyId and PartyId, then use the same service mapping to price sales orders and contracts.

Example

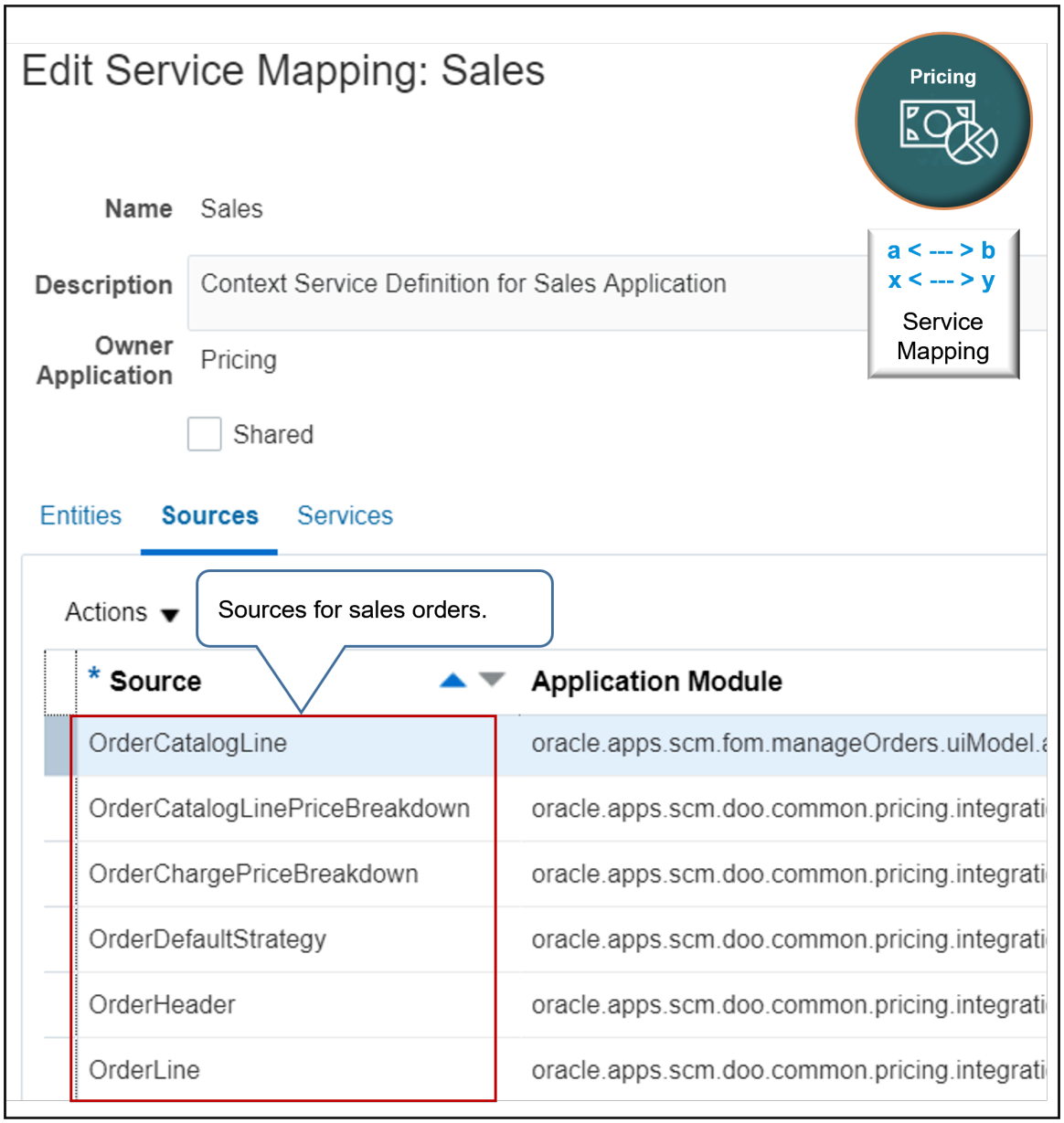
You can map between the Header entity and the OrderHeader source:



If the Order Entry Specialist enters a value in the Customer attribute, then Pricing uses the PriceSalesTransaction service because this service references the Price Sales Transaction algorithm that identifies the strategy, and it uses the OrderHeader source because the Customer attribute is part of the order header where the Order Entry Specialist enters the customer name.

Sources

A source is the data source that Pricing uses to create the input service data object when it calls the service on a service mapping:



Pricing comes predefined with sources for Order Management and Contract Management, and these sources already include entities and attributes. For example, the OrderHeader source includes entities and attributes that represent attributes you find on the order header, such as CustomerId to represent the Customer attribute on the header, and SellingBusinessUnitId to represent the Business Unit attribute.

Here are some of the more important sources you use to price a sales order:

Source	Description
CalculateSalesTotals	Calculate price totals on the sales order.
CreateRollupCharges	Calculate roll up charges on the sales order.

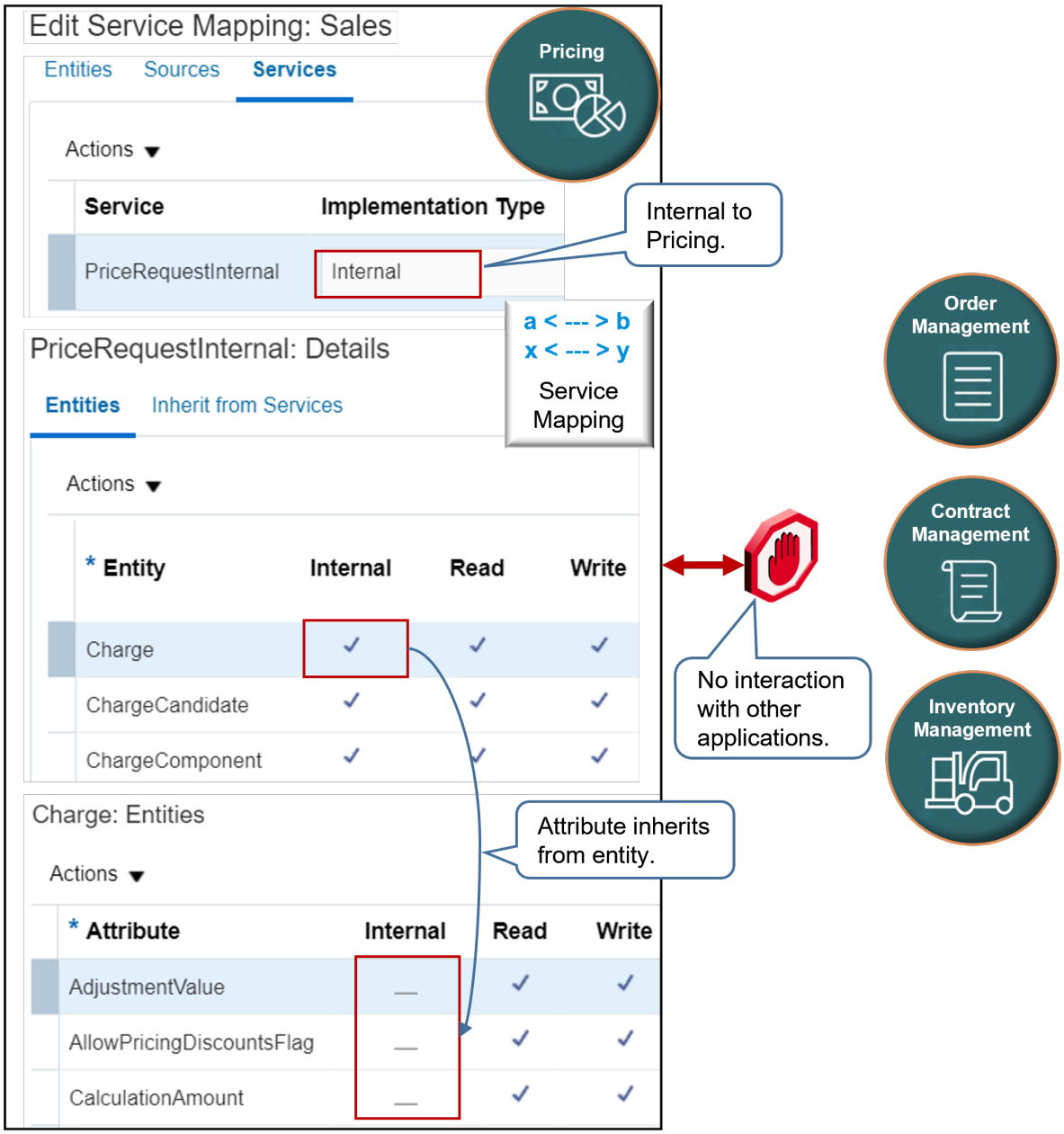
Source	Description
GetSalesPricingStrategy	Determine the default pricing strategy.
PriceSalesTransaction	Price a transaction in the sales order.
PricingResultsPresentation	Display the charge breakdown.
SalesOrderTotalsPresentation	Display the totals breakdown.
ValidateSalesPrices	Validate prices on the sales order against pricing guidelines and other pricing rules.

Services

Pricing uses its own services to price a document. Some services are internal. The reside inside Oracle Pricing. Others are external. They reside outside of Oracle Pricing.

Internal Services

Pricing uses its own internal services to populate values in the pricing algorithm:

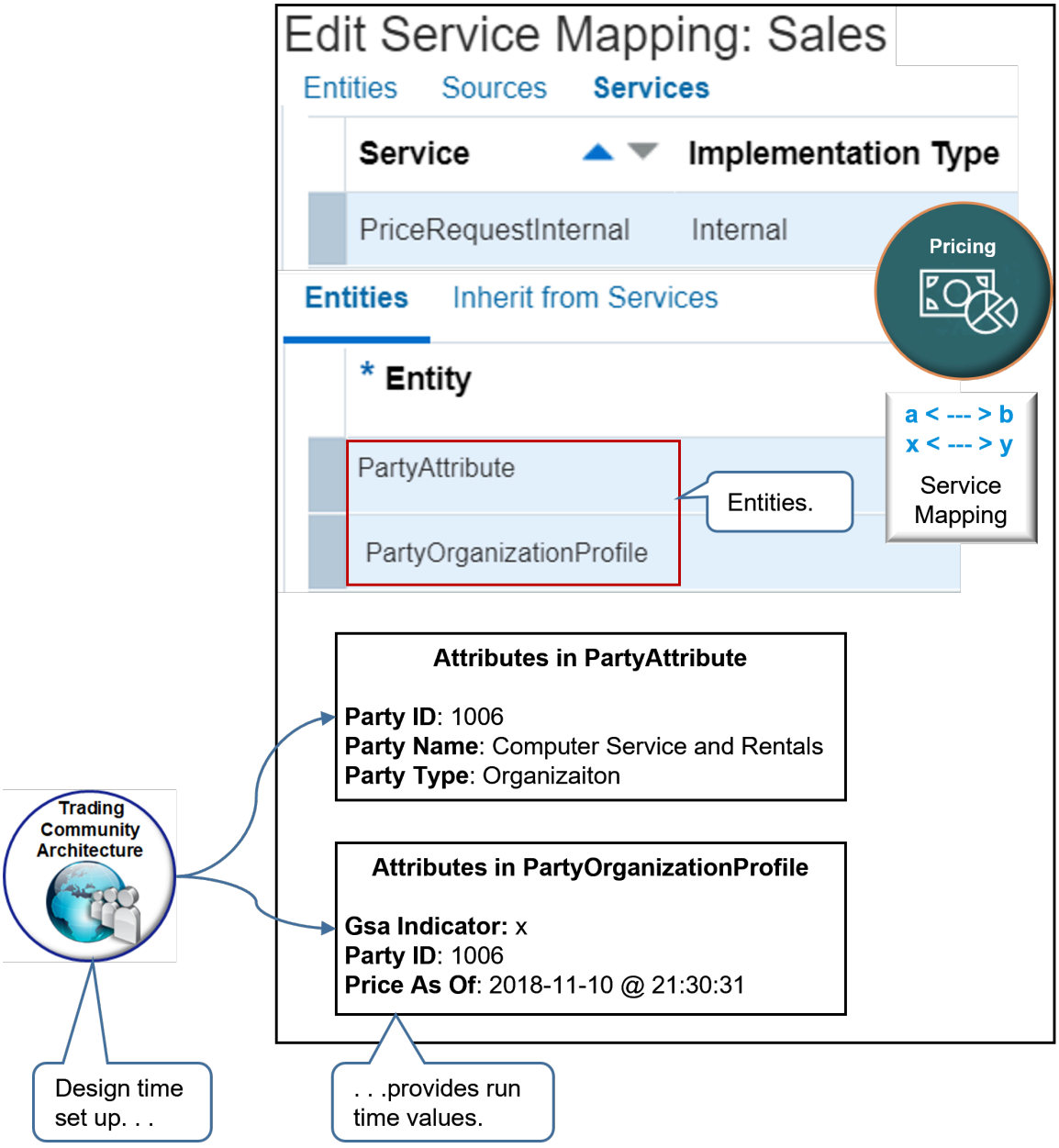


Note

- The term **internal** means Pricing calculates these values entirely within Oracle Pricing and doesn't share them with the calling or any other application, such as Order Management.
- The term **external** basically means any processing that happens outside of Oracle Pricing, such as rendering price in the charge breakdown of a sales order in Order Management.
- PriceRequestInternal is an example of an internal service. It comes predefined with entities and attributes that Pricing uses to populate values in the algorithm.
- The child attributes of an entity inherit the Internal value from the entity. For example, if the Charge entity is Internal, then all child attributes of Charge are internal.

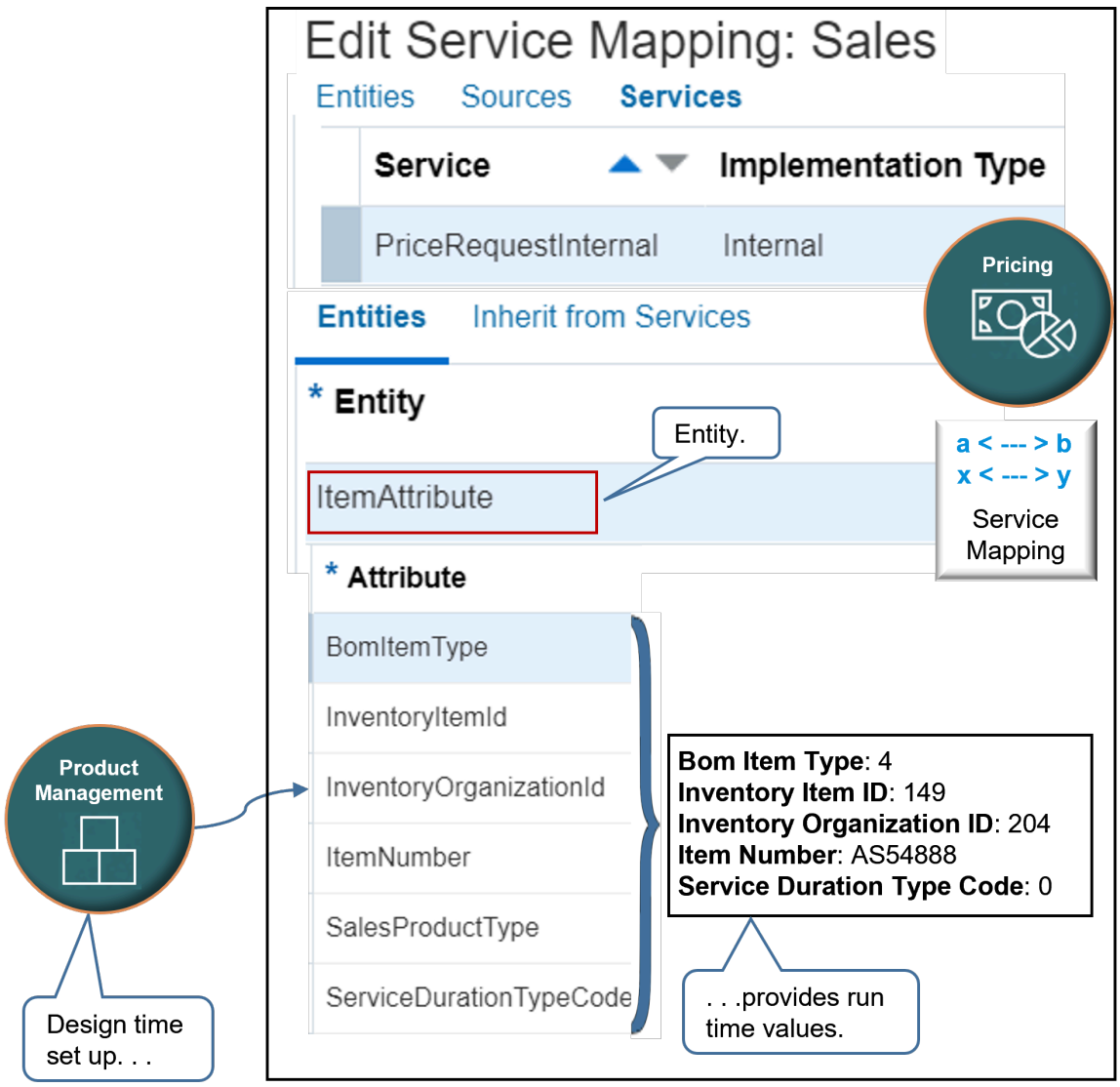
Customer and Item Data

Pricing uses the PartyAttribute entity and the PartyOrganizationProfile entity of the Sales service mapping to store data about your customer who purchases the item:



Pricing gets this data from the set up you do in the application you use to store customer data, such as Trading Community Architecture.

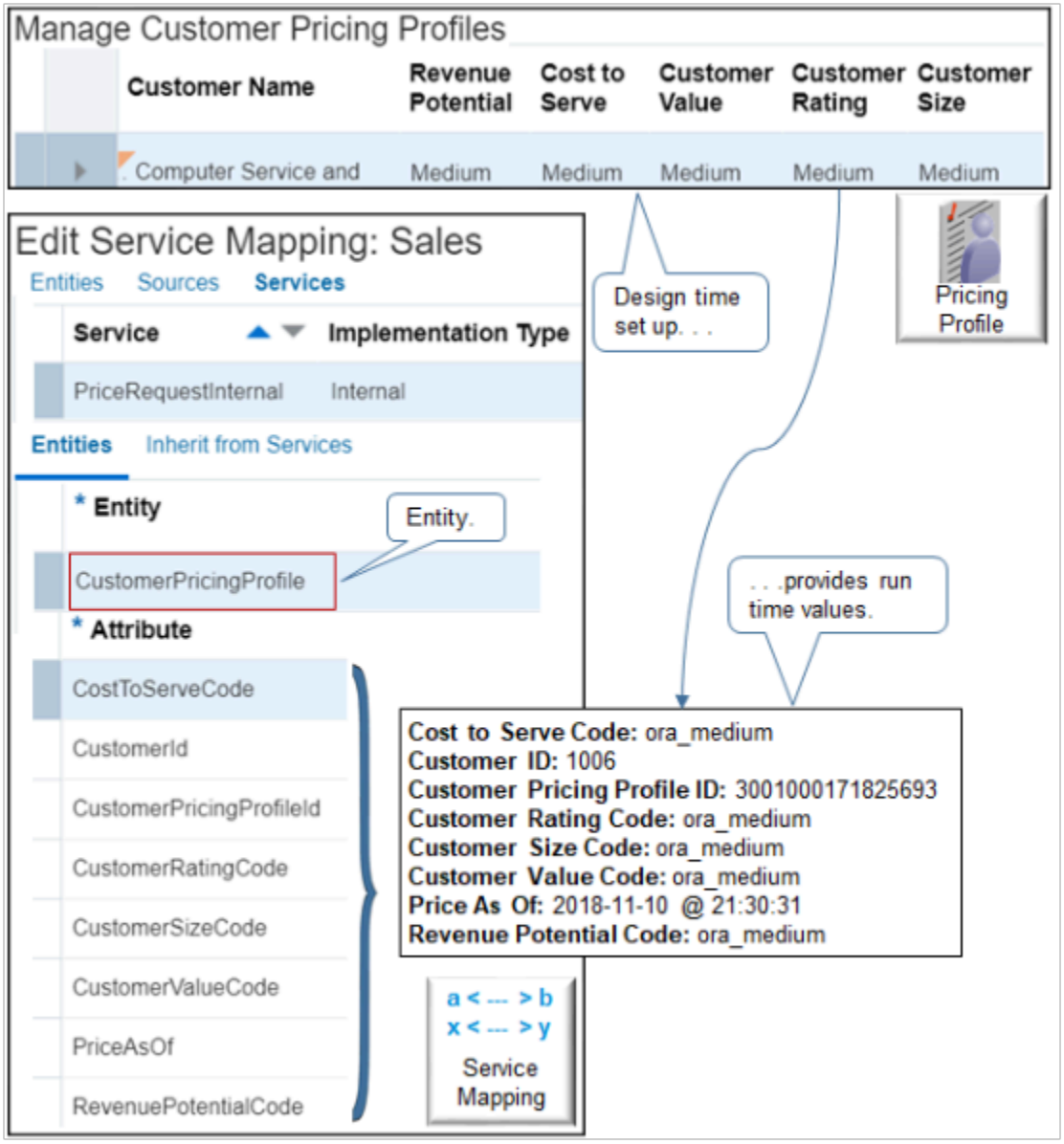
Pricing uses the ItemAttribute entity to store data about the item you need to price:



Pricing gets this data from the set up you do during design time in Product Management. For example, if you set the value for the item as AS54888 in Product Management, then the Item Number attribute of the ItemAttribute entity in Pricing contains AS54888 at run time.

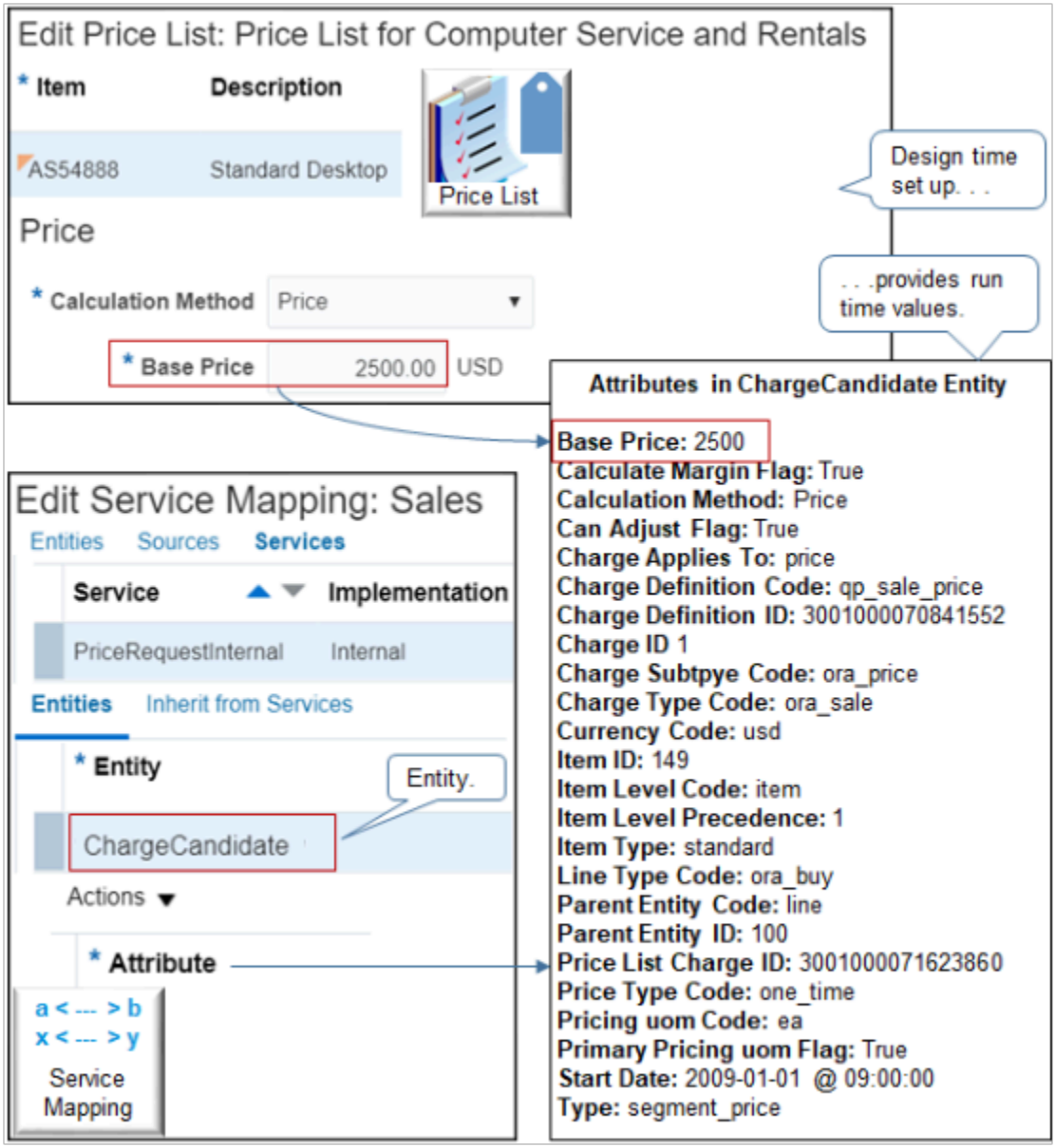
Profiles, Segments, and Strategies

Pricing uses profiles, segments, and strategies in all pricing scenarios. It stores profile data in the CustomerPricingProfile entity. It uses your design time set up on the Manage Customer Pricing Profiles page to determine the attributes to use, such as Cost to Serve, and the value to use for each attribute, such as Medium:



Price Lists

The ChargeCandidate entity contains runtime values according to your design time set up On the Price Lists page:

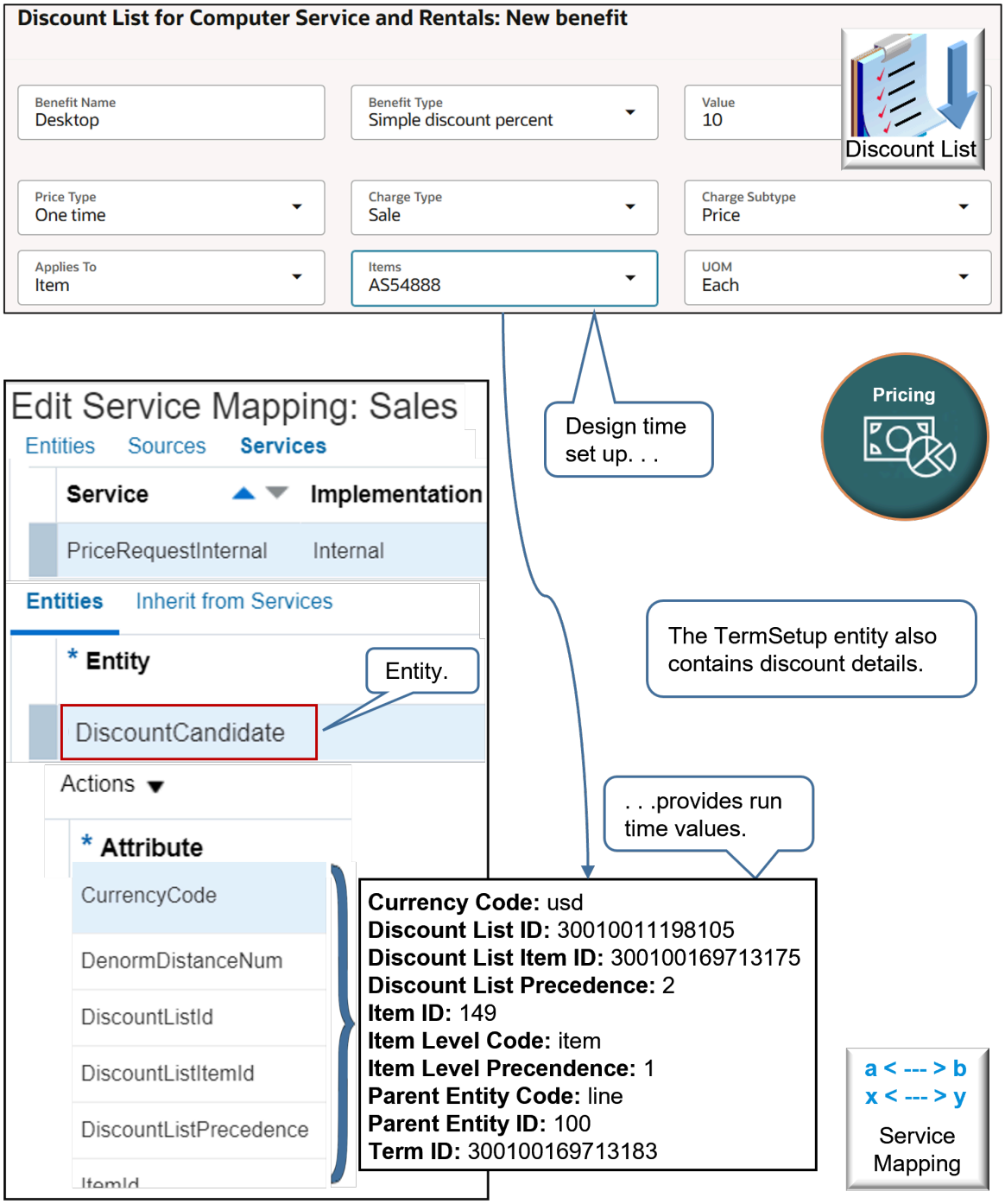


For example, at design time, assume you set up a price list for customer Computer Service and Rentals, and add item AS54888 with a base price of 2500 and calculation method of Price. Here's what pricing does at run time:

- Sets the Base Price attribute of the ChargeCandidate entity to 2500.
- Sets the Calculation Method attribute to Price.
- Uses values from the price list to set all the other attributes.
- Calculates ChargeCandidate each time that you add an order line, and uses these values to create charges and charge components.

Other Lists

Other lists, such as discount lists, work the same way:

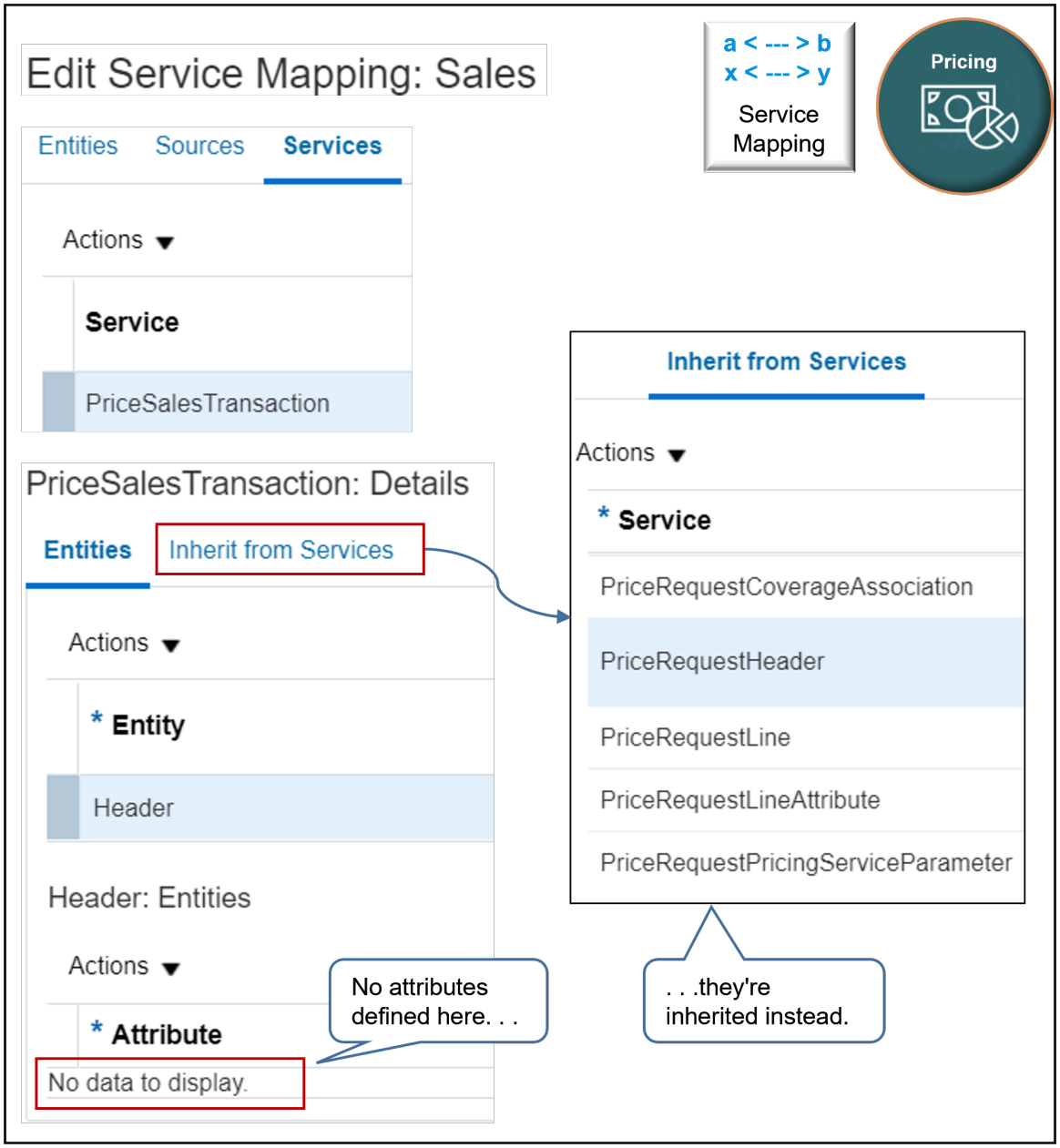


Note

- If you set up a discount list on the Manage Discount Lists page, then Pricing uses design time values from the discount list to populate attributes on the DiscountCandidate entity at run time.
- If you set Item Level on the discount list to Item, then Pricing sets the ItemLevelCode attribute of the DiscountCandidate entity to Item.
- Pricing also uses the TermSetup entity to store details about the discount, such as AdjustmentAmount.

Inherit Attributes from Other Services

Some service mappings inherit their entities and attributes from another service mapping. For example, the Header entity of the PriceSalesTransaction service mapping doesn't include any attributes. But don't worry. It inherits header attributes from the PriceRequestHeader service instead:

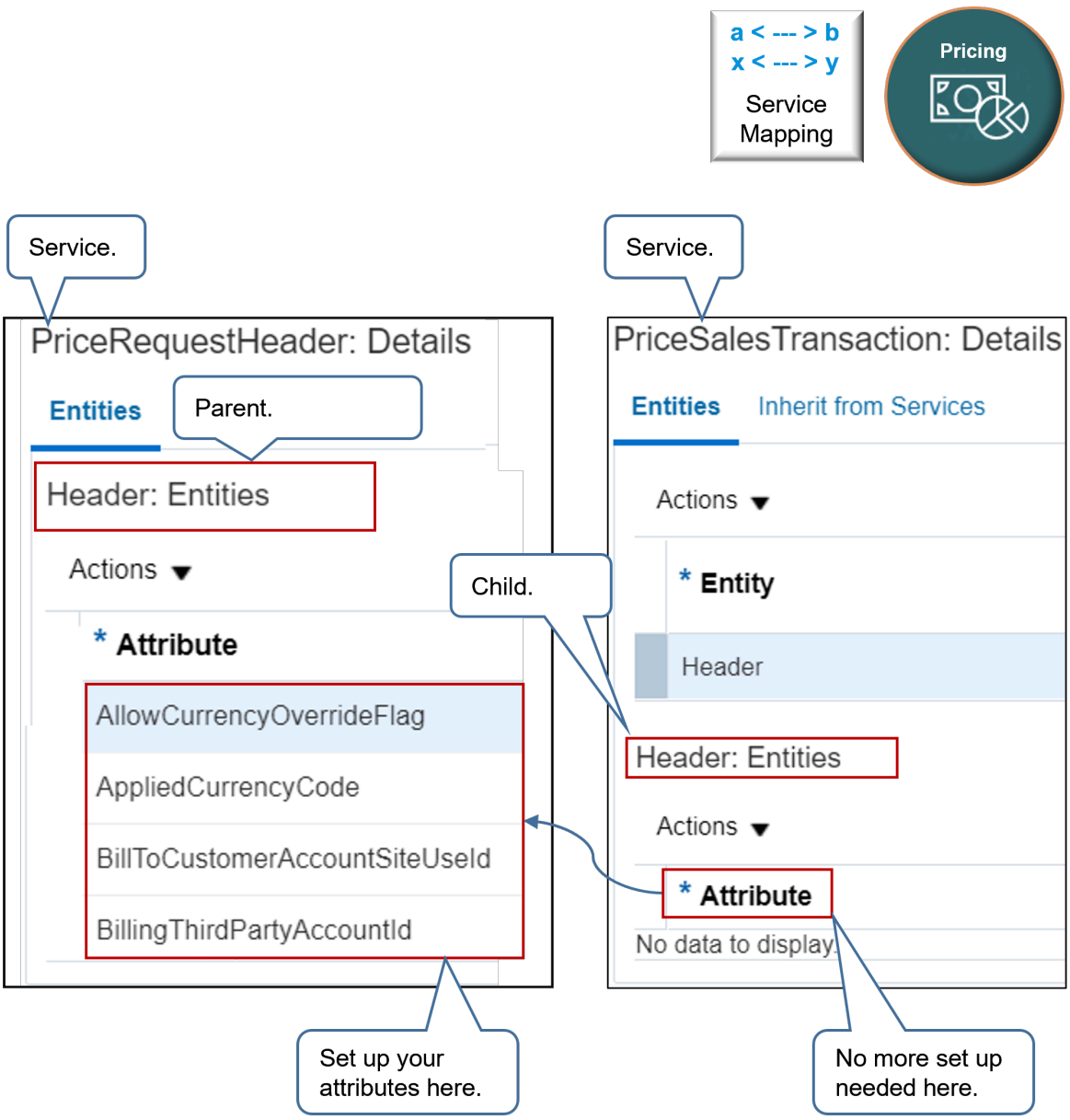


Note

- Click **Inherit from Services** to view the list of services.
- PriceSalesTransaction comes predefined to inherit the entities and attributes that these services reference. For example, PriceRequestHeader references the AllowCurrencyOverrideFlag attribute, so PriceSalesTransaction inherits AllowCurrencyOverrideFlag.

- This set up enables you to make modifications in one service, PriceRequestHeader, then use them in PriceSalesTransaction, or any other service mapping that inherits from PriceRequestHeader.
- Inheritance includes the Read and Write set ups from the parent. For example, if Read is enabled and Write is disabled on AllowCurrencyOverrideFlag in PriceRequestHeader, then Read usage is enabled and Write usage is disabled on AllowCurrencyOverrideFlag when you use it in PriceSalesTransaction.
- PriceSalesTransaction also comes predefined to inherit from other services, such as PriceRequestLine for all the entities and attributes that an order line contains.
- Inheritance saves you time and maintenance because you do most of your entity and attribute set up in a single parent service instead setting up entities and attributes in a bunch of services, then having to manage all of them.

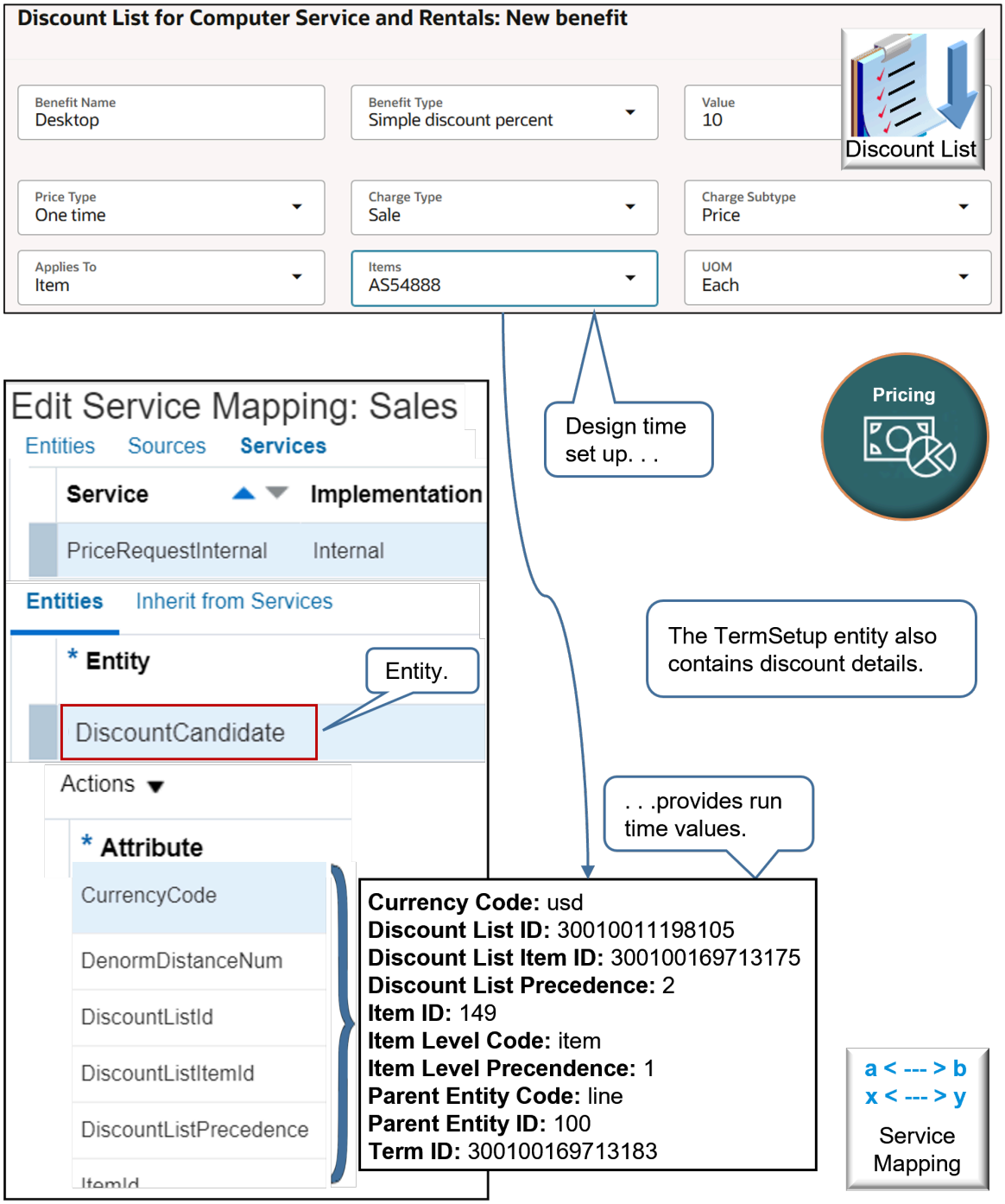
Here's the relationship between the parent and child:



- Modify the parent and you're done. Your modifications are automatically available in the child. You don't need to do any other set up.
- The child inherits all the entities and attributes from the parent. Use them as if you set them up on the child.

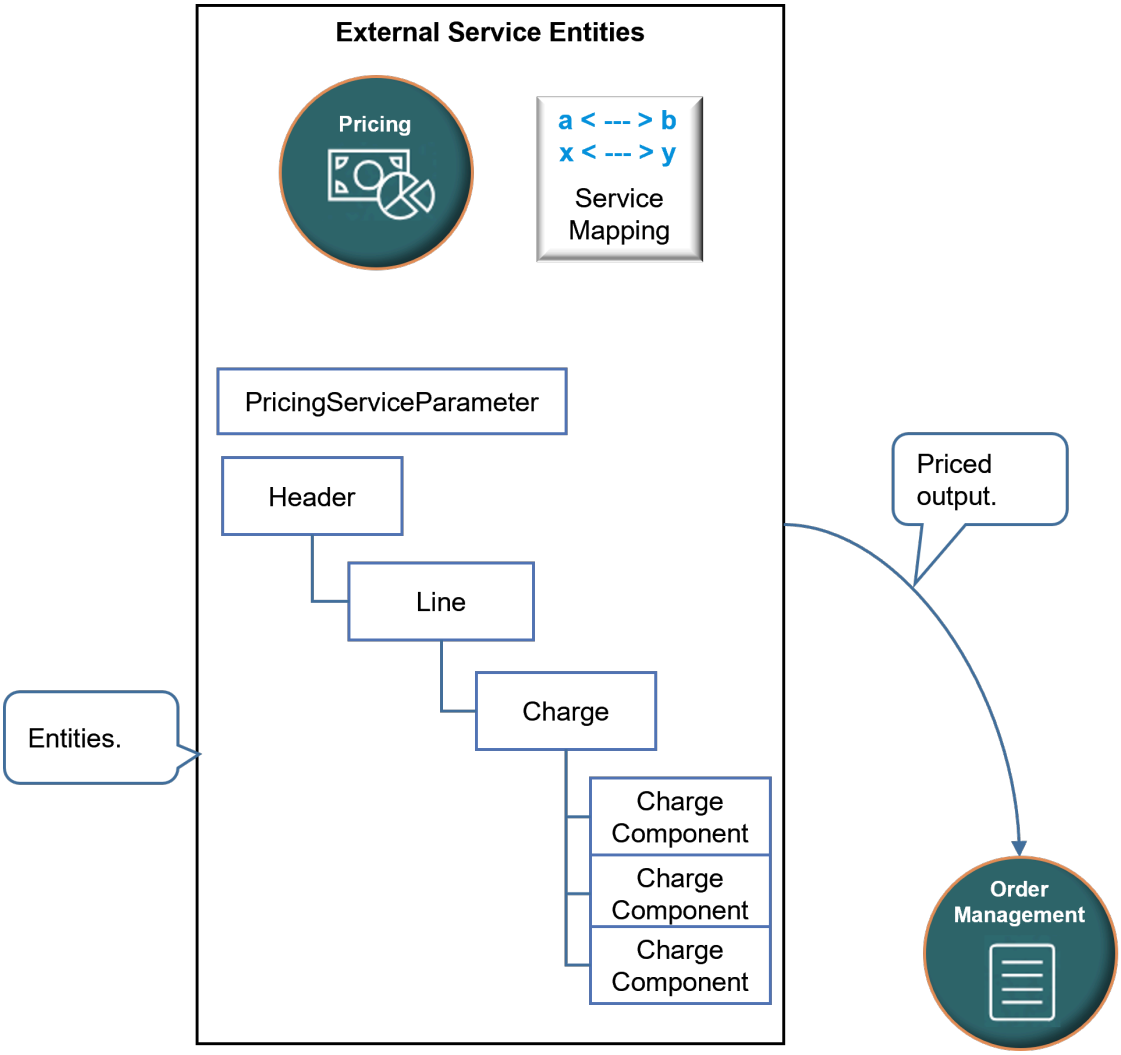
PriceRequestInternal

PriceRequestInternal inherits entities and attributes from other services. If you must modify behavior that one of these other services control, then modify it instead of modifying PriceRequestInternal. For example, modify CalculateSalesTotals to change behavior that Pricing uses to calculate the sales order total:



Output

Here are the entities where Pricing writes data into the view object it uses to communicate output details back to Order Management after it finishes pricing:



Here's the structure it uses. The structure reflects the structure of a sales order.

```
Header
 Line
 Charge
 Charge Component
 Charge Component
 Charge Component
 Charge Component
```

Dot Notation

The service mapping uses dot notation.

```
Service.Entity
 Entity.Attribute
```

where

- Service is the name of the service.
- Entity is the name of the entity that the service contains.

- Attribute is the name of the attribute that the entity contains.

For example:

```
PriceSalesTransaction.Header
 Header.CustomerId
```

where

- `PriceSalesTransaction.Header` identifies the Header entity in the PriceSalesTransaction service.
- `Header.CustomerId` identifies the CustomerId attribute of the Header entity that resides in the PriceSalesTransaction service.

For another example.

```
PriceSalesTransaction.Line
 Line.InventoryItemId
```

where

- `PriceSalesTransaction.Line` identifies the Line entity in the PriceSalesTransaction service.
- `Line.InventoryItemId` identifies the InventoryItemId attribute of the Line entity that resides in the PriceSalesTransaction service.

Examples of Using Service Mappings

Get detailed instructions about how to use service mappings to meet your business requirements:

- *Manage Service Mappings*
- *Assign Pricing Strategy According to Order Type*
- *Create Discounts That Accumulate or Cascade*
- *Create Price Lists for Each Customer*
- *Use Extensible Flexfields with Pricing*
- *Adjust Price According to Predefined Attributes*
- *Override Base List Price*

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Service Mappings
- Pricing Algorithms

Create a Sandbox So You Can Edit Service Mappings

You must use a sand box when you edit a service mapping. If you access the Manage Service Mappings page outside of a sand box, then the page disables the actions you need to edit the mapping, such as Add Row.

Try it:

1. Go to the Sandboxes work area.
2. On the Sandbox page, click **Create Sandbox**.

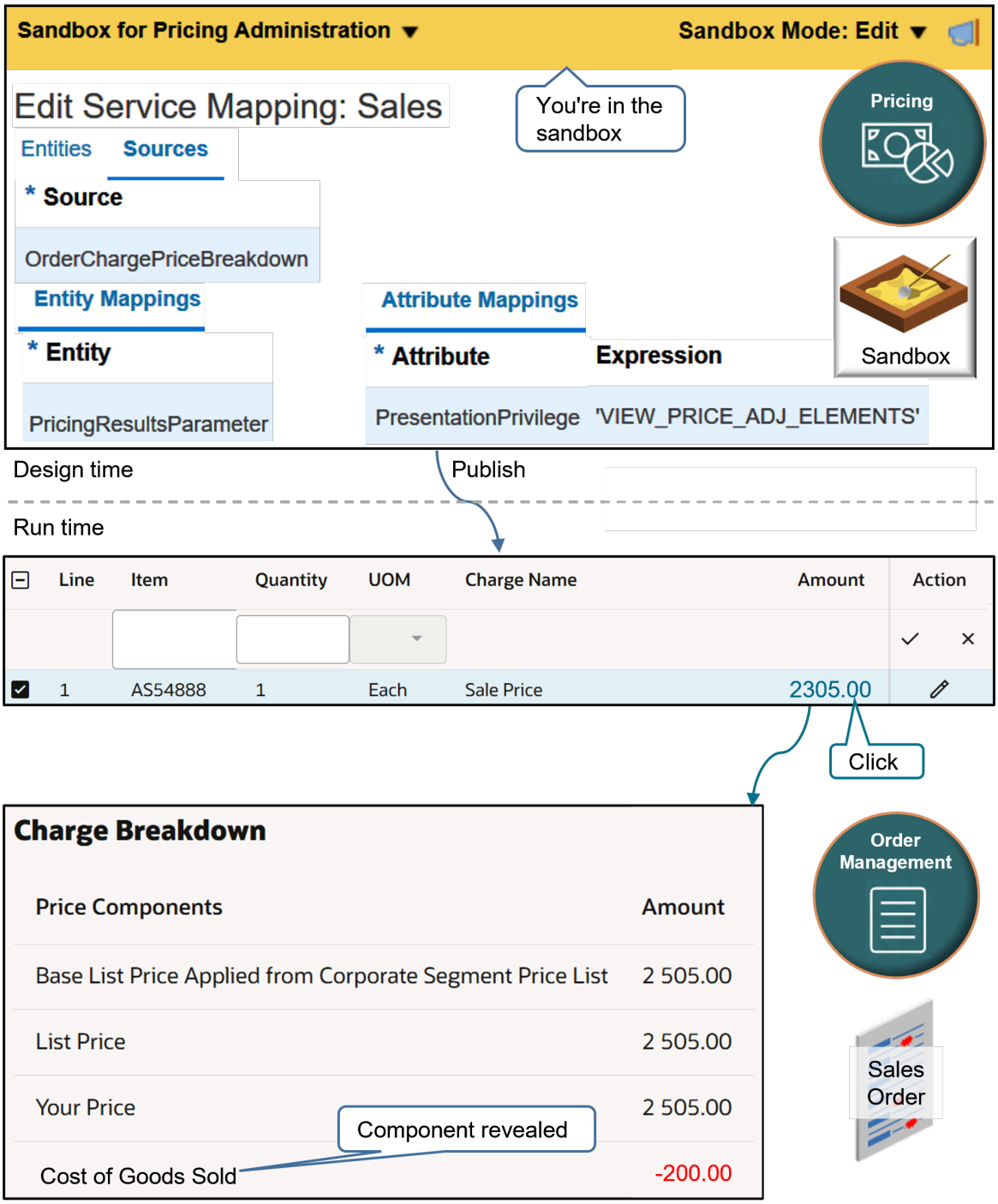
3. On the Create Sandbox page, set the value:

Attribute	Value
Name	Enter any text. For this example, enter Sandbox for Pricing Administration.

4. Add a check mark to the Manage Service Mappings tool, then click **Create and Enter**.

5. Go to the Pricing Administration work area, then edit your service mapping.

   A yellow banner displays at the top of the page to indicate that you're in the sandbox:



6. At the top of the page, click **Sandbox for Pricing Administration > Publish**.

Related Topics

- Create and Activate Sandboxes

Manage Service Mappings

Add attributes that contain your data, add attributes to the service, and add attributes to the source.

Assume you must add an attribute to a predefined service mapping so Pricing can map freight details for an order line.

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

Summary of the Set Up

Manage a service mapping:

1. Add the attribute that will contain your data.
2. Add your attribute to the service.
3. Add your attribute to the source.

Add the Attribute That Will Contain Your Data

1. Create a sand box. For details, see *Create a Sandbox So You Can Edit Service Mappings*.
2. On the Pricing Administration page, click **Tasks > Manage Service Mappings** under Pricing Configuration.
3. On the Manage Service Mappings page, click **Sales**.

   Pricing comes predefined with the Sales service mapping that prices sales orders and order lines.
4. On the Edit Service Mappings page, click **Entities**.

   You use the Entities tab to specify the entity that receives the output of the service mapping.
5. Click the **row** that includes Line in the Entity column.

   The order line contains the freight on board attribute, so you add it to the Line entity.
6. In the Line Details area, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Attribute	FreightOnBoardCode_Custom You must include the _Custom suffix for any attribute or entity you create. Don't include spaces.
Type	String Make sure you use a type that's compatible with the attribute you plan to use. For example: o Use String to represent the order type. o Use Integer to represent the order number.
Allow Null	Contains a check mark

Attribute	Value

Add Your Attribute to the Service

Add the attribute to the service that will process the request so the service can reference the attribute when it maps the input service data object (SDO) to the output SDO:

1. Click **Services**, then click the **row** that includes PriceRequestLine in the Service column.

   Each of the services that come predefined with Pricing performs a finite service. For example, PriceRequestLine processes a request to price an order line, CalculateSalesOrderTotals processes a request to calculate the total price for a sales order, and so on.
2. In the PriceRequestLine Details area, in the Entities tab, click the **row** that includes Line in the Entity column.
3. In the Line Entities area, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Attribute	FreightOnBoardCode_Custom You might need to scroll to the bottom of the list to locate this attribute.
Read	Contains a check mark
Write	Doesn't contain a check mark

Add Your Attribute to the Source

The service mapping uses the source to structure the input SDO. In this example, the calling application, Oracle Order Management, will send a code for freight on board, so you set up the source so it can accommodate the code:

1. Click **Sources**, then click the **row** that includes OrderLine in the Source column.

   Each source that comes predefined with Pricing provides a unique model that the service mapping can use to structure the input SDO. Consider these example sources:

   - **OrderLine.** References entities on the order line, such as the Line entity. These entities reference attributes on the order line, such as quantity, the date that the customer requested the order line, and so on.
   - **OrderHeader.** References entities on the order header, such as the Header entity. These entities reference attributes on the order header, such as the customer identification, the selling business unit, pricing strategy, and so on.
2. In the OrderLine Details area, in the Entities tab, click the **row** that includes Line in the Entity column.

3. In the Line Details area, in the Attribute Mappings tab, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Attribute	FreightOnBoardCode_Custom
View Object Attribute	FreightTermCode

If you encounter a problem, see the Service Mappings subtopic. For details, see *Troubleshoot Your Pricing Setups*.

Related Topics

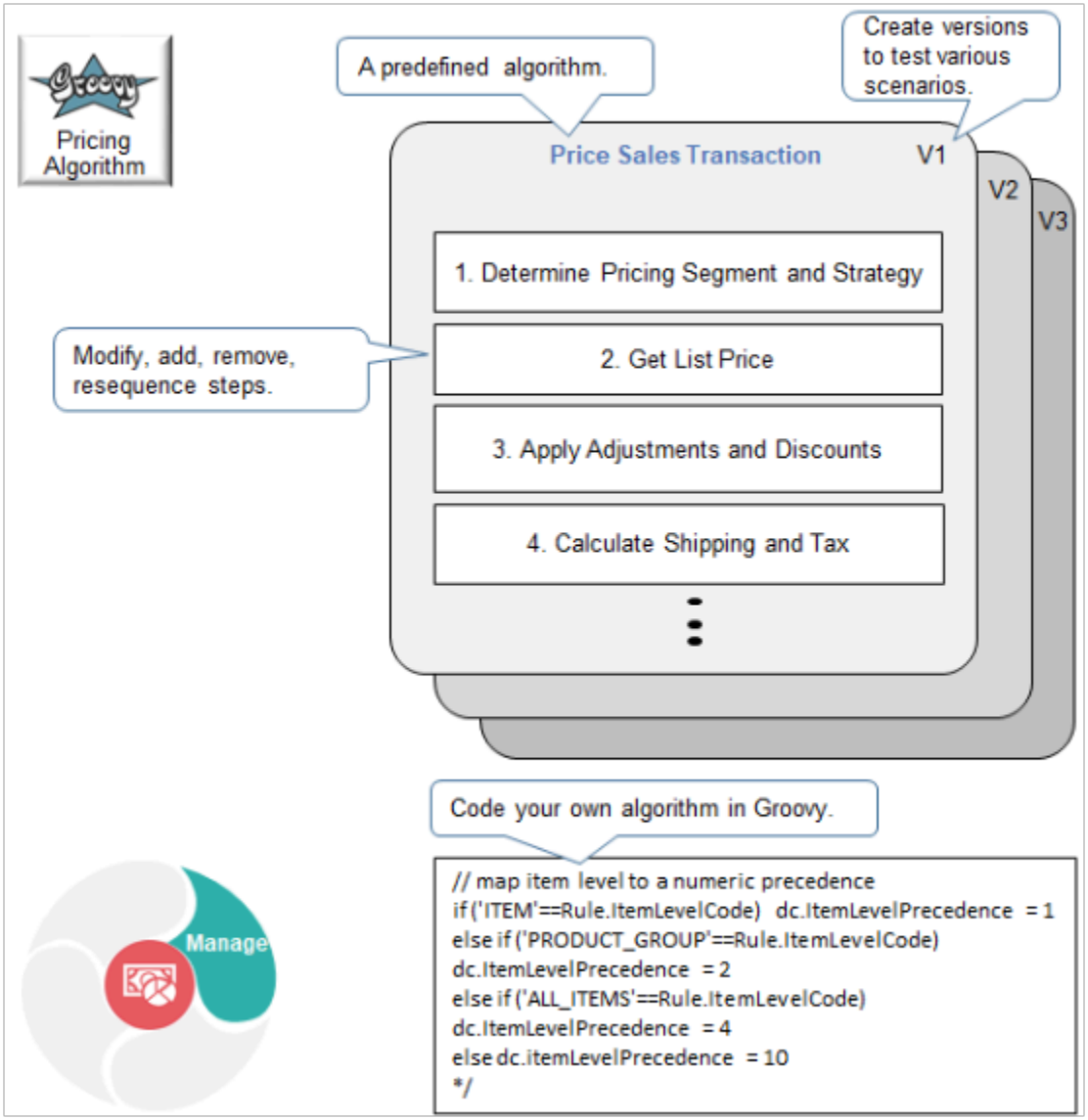
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Troubleshoot Your Pricing Setups
- Create a Sandbox So You Can Edit Service Mappings
- Overview of Oracle Pricing

Algorithms

Overview

Pricing Algorithms

Use a pricing algorithm to manage pricing logic and price calculations.



A pricing algorithm is a set of rules that uses conditional logic, variables, functions, and Groovy script to manipulate data that affects pricing. Use it to modify the logic that Oracle Pricing uses when it prices an item:

- Create procedural logic. Pricing uses different pricing algorithms to calculate different prices. For example, the predefined Price Sales Transaction algorithm uses logical steps to price each sales transaction:
    - Determine Pricing Segment and Strategy
    - Get List Price
    - Apply Adjustments and Discounts
    - Calculate Shipping and Tax
- Modify, reuse, resequence, add, or remove steps.
- Add functions and variables.
- Create your own algorithm.
- Use Groovy script and Expression Builder to code and extend your algorithm.

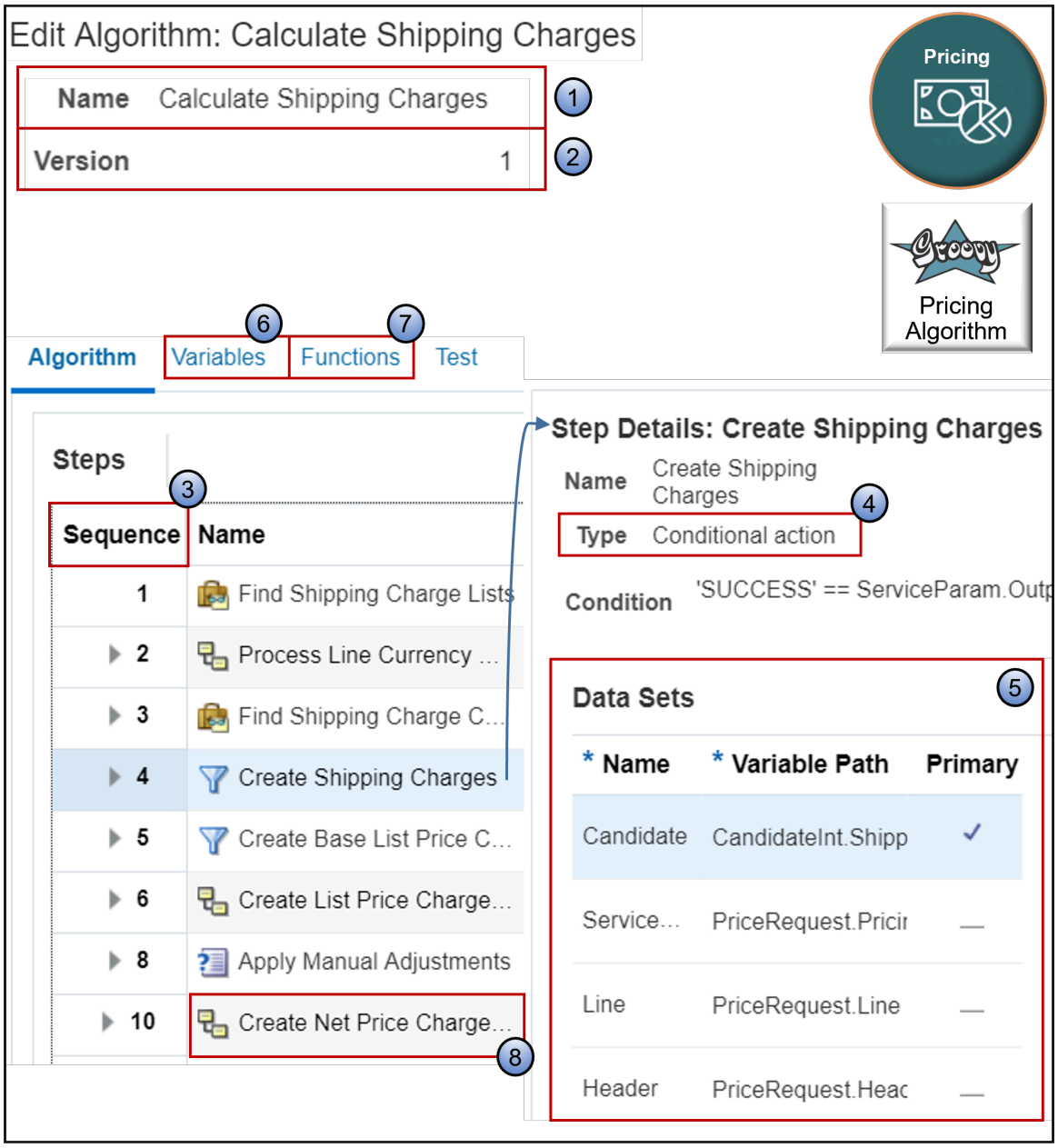
- Add your own dynamic pricing calculation. For example, calculate a commodity price that changes daily. For another example, set up pricing for your sales orders that's different from pricing for your service contracts, and set up pricing for your service contracts that's different from pricing for your subscriptions.
- Create and publish different versions of your algorithm and test them to try out different scenarios.
- Reference another pricing algorithm from your own pricing algorithm.

For use cases that use pricing algorithms, see the Use Cases chapter, starting at *Overview of Pricing Use Cases*. For even more use cases, see *Technical Reference for Oracle Pricing (KB171412)* on My Oracle Support, then examine the various attachments.

Parts of a Pricing Algorithm

A pricing algorithm manipulates data in the output service data object (SDO) that Pricing uses to calculate price, such as adding a value to each attribute in the SDO.

Here are the parts of a typical pricing algorithm:



Note

1. **Name.** Each pricing algorithm achieves a finite goal, such as determining the shipping charge list to use when calculating the shipping charge, and creating the shipping charges for each order line that requires a shipping charge calculation.
2. **Version.** Pricing increments the version of the pricing algorithm each time you publish it.
3. **Sequence.** A series of steps that the pricing algorithm processes in a sequence. You can move each step up or down to modify the sequence.

   Assume a tax authority requires that you calculate tax only on the item that you are selling, and not on the shipping charge. You can move Step 10, Compute Tax, to immediately above Step 4, Create Shipping Charges.

4. **Step Type.** Click the step, then use the Step Details area to modify the step.

   The Type specifies the type of logic that the step uses, such as Conditional Action, Nested Action, Group, or Subalgorithm.

   Step 4, Create Shipping Charges, is a conditional step that writes shipping charges to the output SDO only if the order line is a candidate for a shipping charge. Here's the condition it uses:

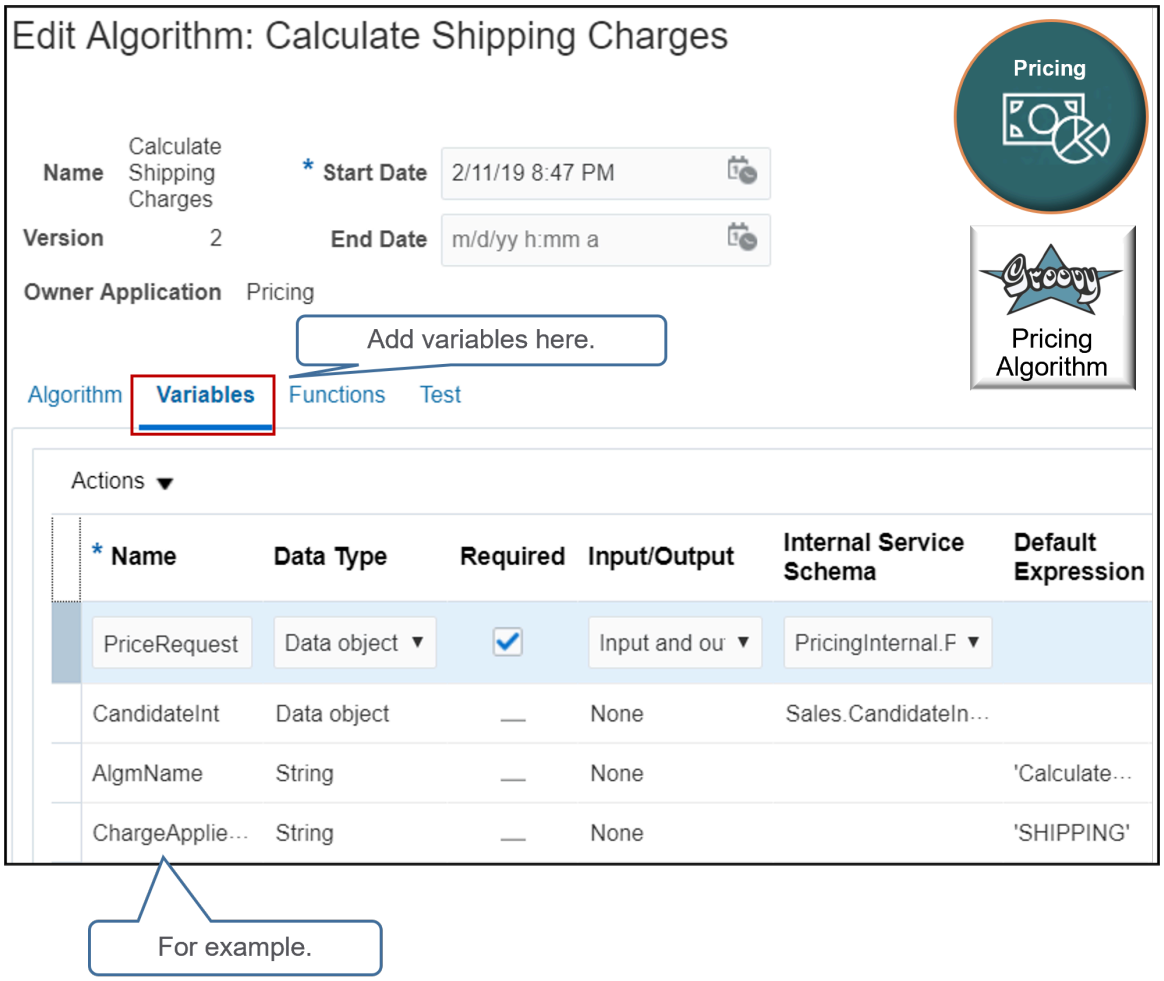
Condition	Description
`'SUCCESS' == ServiceParam.OutputStatus && 'LINE' == Candidate.ParentEntityCode`	Run this step only if the order line is a candidate for a shipping charge. Here's the format that the condition uses: o `SUCCESS` o `ServiceParam`. Identifies a data set in this pricing algorithm step. o `OutputStatus`. o `&&`. o `LINE`. o `Candidate`. Identifies a data set in the pricing algorithm step. o `ParentEntityCode`.

   For details, see *Manage Pricing Algorithm Steps*.
5. **Data Sets.** Each row in the Data Sets area defines a data set, and each data set filters the set of records that the step processes.
6. **Variables.** Set up variables that the pricing algorithm uses. For example, if you set the NeedsCurrencyConversion variable to Yes in the Calculate Shipping Charges algorithm, then each algorithm step that involves currency will do a conversion.
7. **Functions.** Set up functions that receive data from the algorithm, process it, then return one or more values to the algorithm.
8. **Subalgorithm.** Another pricing algorithm that this pricing algorithm references.

   Step 10 of the Calculate Shipping Charges algorithm references the Create Net Price Charge Component subalgorithm. At runtime, Pricing runs the subalgorithm, then returns to the subsequent step. It returns to step 11.

Variables

A variable in a pricing algorithm stores a value that can change depending on conditions or details that pass through each algorithm step. Use it to receive data from the object that calls the pricing algorithm, or to send data to the object:

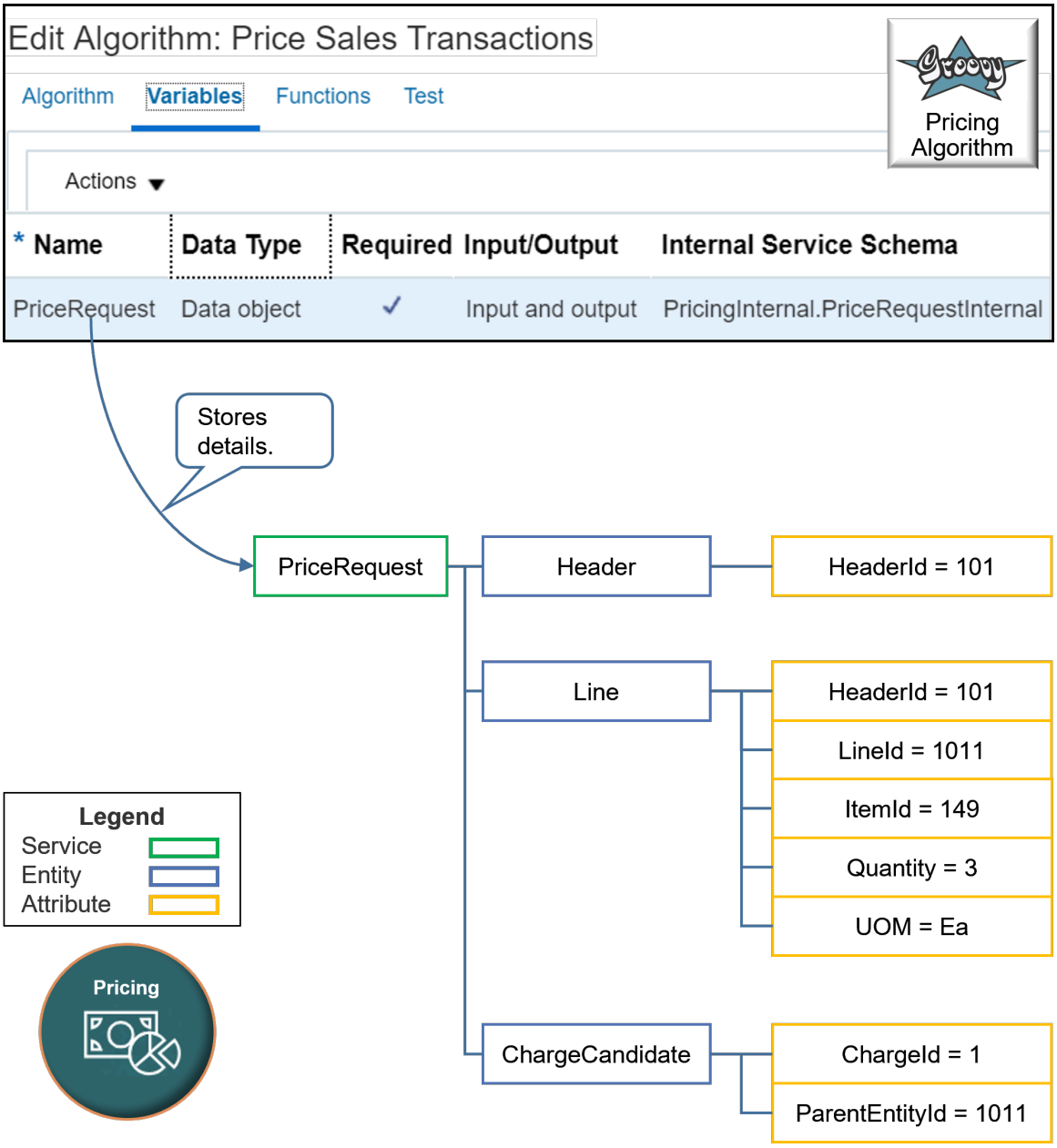


Here are the attributes you can use with a variable:

Attributes	Description
Name	Alphanumeric text that describes the variable. Use a name that's unique in the algorithm. Don't include spaces.
Data Type	Data type, such as String. If you set Data Type to Data Object, then you must set a value in the Internal Service Schema attribute.
Required	Add a check mark to make this variable required in an algorithm step.
Input/Output	Specify usage: • **Input**. The object that calls this pricing algorithm can send the value of this variable to the pricing algorithm. • **Output**. The algorithm can send the value of this variable to the object that calls this pricing algorithm.

Attributes	Description
Internal Service Schema	If you set the Data Type variable to Data Object, then set Internal Service Schema to the service that provides details to this variable, or that can use the output that this variable provides. For example, the CalculateSalesOrderTotals service calculates the total price of a sales order. If you must use this variable to calculate total price, then set Internal Service Schema to CalculateSalesOrderTotals. Most algorithms include at least one Data Object variable.
Default Expression	Groovy expression that specifies the default value for the variable. If the input payload results in setting the value of the variable to a value that isn't empty, then Pricing ignores the expression. You can use an expression for each data type except Data Object.

Most algorithms include the PriceRequest variable. It stores details for the PriceRequest service data object according to the transaction you must price, such as a sales order:



Note

- PriceRequest uses the PricingInternal.PriceRequestInternal internal service schema.
- PriceRequest stores details for the Header, Line, and ChargeCandidate entities.
- Pricing creates the ChargeCandidate from the segment price list on the pricing strategy.

Types of Pricing Algorithms

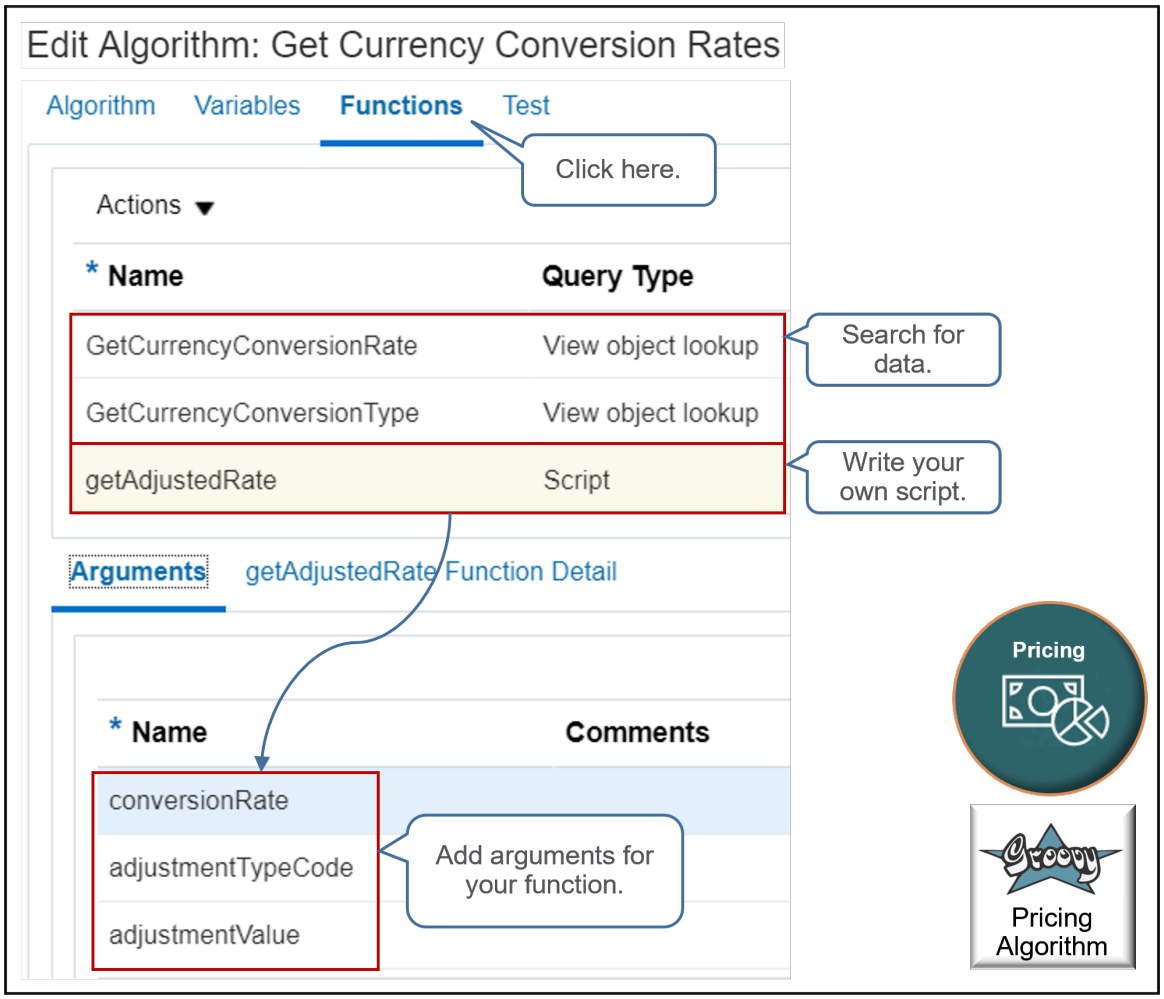
Type	Description
New	A pricing algorithm that Pricing provides for the first time:

Type	Description
	• A new pricing algorithm includes functionality that doesn't exist in prior releases. It might also include corrections for features from prior releases. • New pricing algorithms come predefined in the current release. • The Manage Algorithms page uses version 0 to indicate a new pricing algorithm. • You must manually promote new algorithms when you update. For background, see *Performing Your Quarterly Update (Doc ID 2337485.1)*.
Current	A pricing algorithm that you promoted from a prior release to the current release: • A current pricing algorithm is up-to-date with the current release. • The Manage Algorithms page uses version 1 to indicate a current pricing algorithm. • You can modify a current pricing algorithm.
Extended	A pricing algorithm that you create so it meets your specific business needs. You click **Actions > Create** on the Manage Algorithms page to create an extended pricing algorithm.
Modified	A predefined pricing algorithm that you modified.

Functions

A function in a pricing algorithm is a routine that receives data from the pricing algorithm, processes it, then returns one or more values to the pricing algorithm. You can use a view object lookup function or a script function.

Here's an example of how the predefined Get Currency Conversion Rates algorithm uses two view object lookups, one script, and three arguments:



Note

Attribute	Description
Name	Enter alphanumeric text. Use camel case and make the first letter lower case. Don't include spaces. For example, enter myPricingFunction.
Query Type	Choose a value: • **View Object Lookup**. Search for data in the Oracle database and return a result. • **Script**. Call a script that determines the value.
Argument	Click the Arguments link, then add arguments: • Use arguments to communicate data from the algorithm step to the function, and from the function to the step.

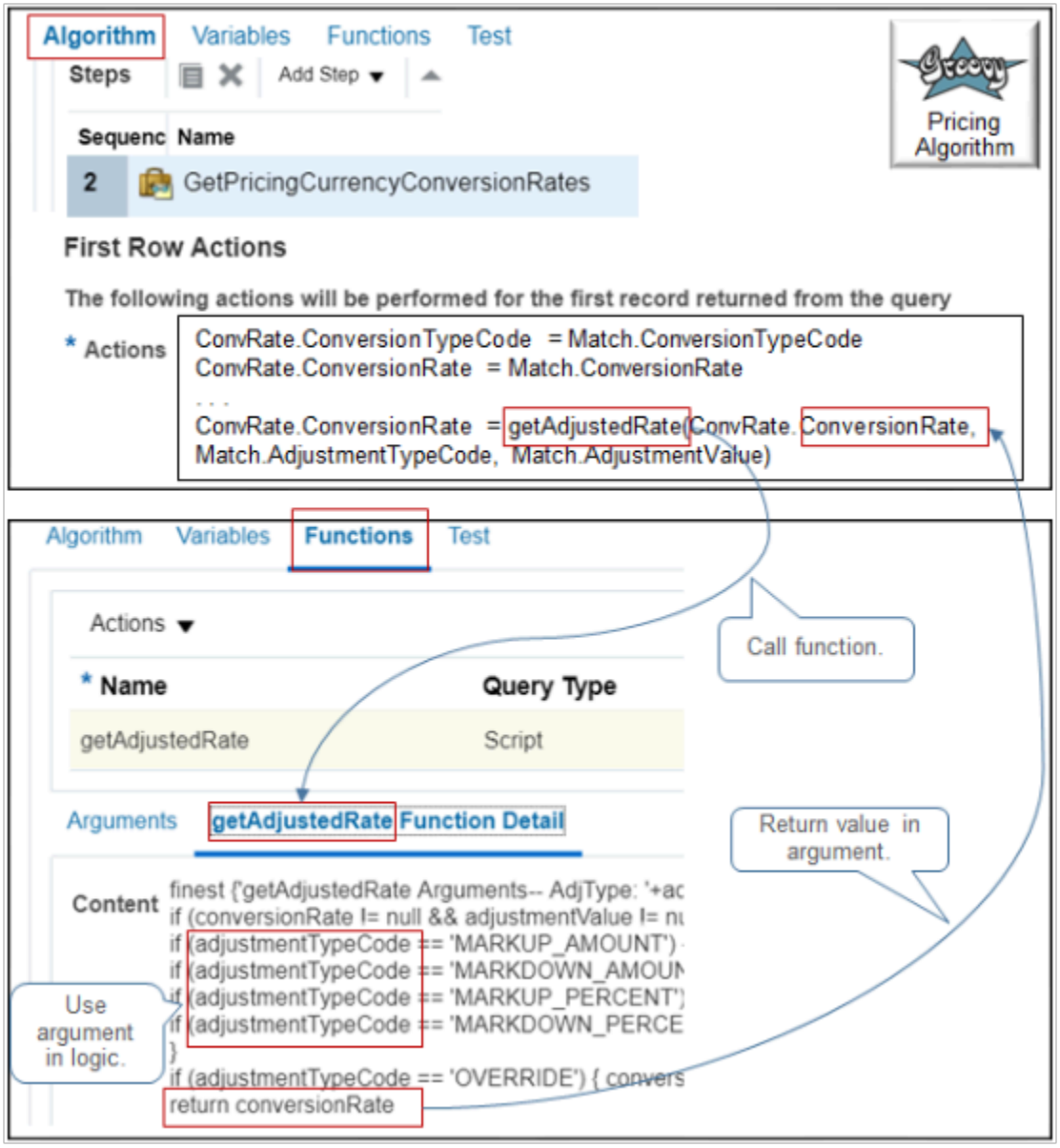
Attribute	Description

	• Enter the name:
	  ○ Make sure its unique in the pricing algorithm across all functions.
	  ○ Use camel case and make the first letter upper case.
	  ○ Don't includes spaces.
	  ○ For example, MyPricingFunctionArgument.
	• Sequence your arguments.
	  Pricing uses the arguments you add in the same sequence you add them. For example, if you add argument x, and then y, and then z, then Pricing uses argument x first, and then y, and then z.
	  Click Move Up or Move Down to modify the sequence.
	• You don't assign a data type to an argument. The argument takes the data type according to the value that the function sends through the argument. For example, if the function sends numeric data through argument x, then argument x is Numeric.
	• Use the Arguments tab to add arguments for a view object lookup or script.

Example of Script

A script runs procedural logic then returns values.

Here's an example of a step that uses script to call a function:



Note

- Step 2, GetPricingCurrencyConversionRates, needs to get the currency conversion rate for the transaction.
- Script in the First Row Actions section of GetPricingCurrencyConversionRates calls the getAdjustedRate function.
- The function uses arguments, such as AdjustmentTypeCode and AdjustmentValue, to calculate the conversion rate.
- The function returns the conversion rate in the ConversionRate argument.

Here's the complete script of the first row action. Notice how the script uses arguments frequently, such as ConversionRate, but makes only one call to the getAdjustedRate function. To improve performance, make only one call to the function in your script.

```
ConvRate.ConversionTypeCode = Match.ConversionTypeCode
ConvRate.ConversionRate = Match.ConversionRate
```

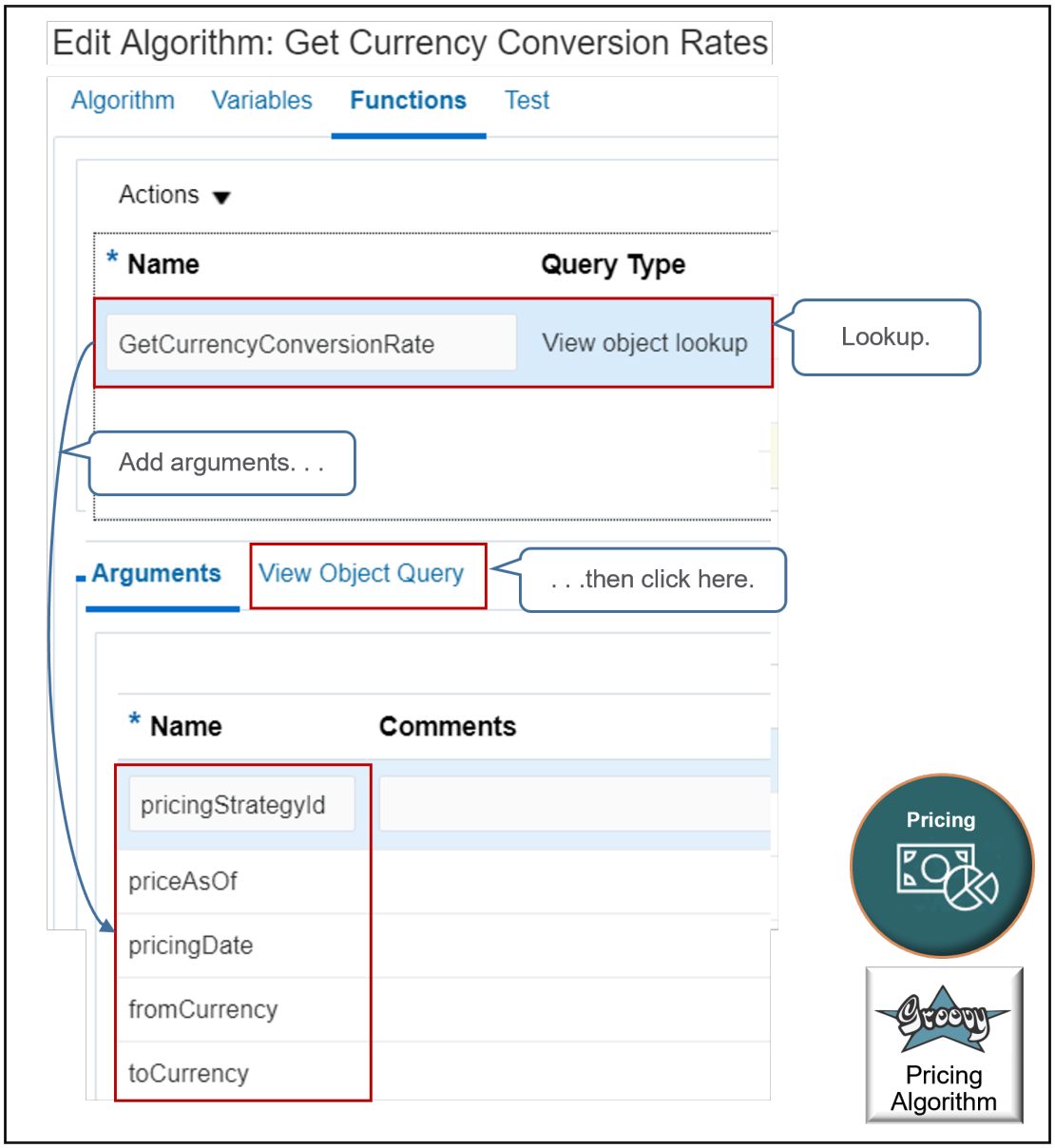
```
finest {'test GetCurrencyConversionRate - on First row'}
if (Match.ConversionTypeCode == 'GL_SOURCED') {
finest {'get GL Rate for -- FromCurr: '+FromCurrencyCode+' ToCurr: '+ToCurrencyCode+' PriceAsOf: '+PriceAsOf
+' GlConvType: '+GlConversionTypeCode}
 rate = pricingUtil.getGlRate(FromCurrencyCode, ToCurrencyCode, PriceAsOf, GlConversionTypeCode)
finest {'GL Rate returned -- ConversionRate: '+rate}
ConvRate.ConversionRate = rate
}
finest {'ConversionRate: '+ConvRate.ConversionRate}
if (Match.AdjustmentTypeCode != null && ConvRate.ConversionRate != null &&
 ((ConvRate.ConversionTypeCode.contains('GL') && ConvRate.ConversionRate != -1 && ConvRate.ConversionRate !=
 -2) || !ConvRate.ConversionTypeCode.contains('GL'))) {
finest {'Rate Adjustment -- AdjType: '+Match.AdjustmentTypeCode+' AdjValue: '+Match.AdjustmentValue}
 ConvRate.ConversionRate = getAdjustedRate(ConvRate.ConversionRate, Match.AdjustmentTypeCode,
 Match.AdjustmentValue)
}
finest {'Conversion Error Check - ConvType: '+ConvRate?.ConversionTypeCode}
if (ConvRate?.ConversionTypeCode?.contains('GL') && (ConvRate.ConversionRate == null ||
 ConvRate.ConversionRate == -1 || ConvRate.ConversionRate == -2)) {
finest {'GL Conversion Error'}
 ConvRate.ConversionRate = null;
 ConvRate.MessageTypeCode = 'ERROR';
 ConvRate.PrcMessageName = 'QP_PDP_PRICE_GL_CURR_ERROR';
 ConvRate.GlMessageName = (ConvPurpose == 'VALIDATE_PRICE')?'QP_PDP_POLICY_GL_CURR_ERROR':null;
 ConvRate.PrcErrorMessage = getFndMessage('QP','QP_PDP_PRICE_GL_CURR_ERROR', null);
 ConvRate.GlErrorMessage = ((ConvPurpose == 'VALIDATE_PRICE')?
getFndMessage('QP','QP_PDP_POLICY_GL_CURR_ERROR', null):null);
}
else if (!ConvRate?.ConversionTypeCode?.contains('GL') && (ConvRate.ConversionRate == null)) {
finest {'QP Conversion Error'}
 ConvRate.ConversionRate = null;
 ConvRate.MessageTypeCode = 'ERROR';
 ConvRate.PrcMessageName = 'QP_PDP_PRICE_CURR_LIST_ERROR';
 ConvRate.GlMessageName = (ConvPurpose == 'VALIDATE_PRICE')?'QP_PDP_POLICY_CURR_LIST_ERROR':null;
 ConvRate.PrcErrorMessage = getFndMessage('QP','QP_PDP_PRICE_CURR_LIST_ERROR', null);
 ConvRate.GlErrorMessage = (ConvPurpose == 'VALIDATE_PRICE')?
getFndMessage('QP','QP_PDP_POLICY_CURR_LIST_ERROR', null):null;
}
```

Here's the complete script of the getAdjustedRate function. Use `finest` to declare your arguments, then add logic, such as If statements.

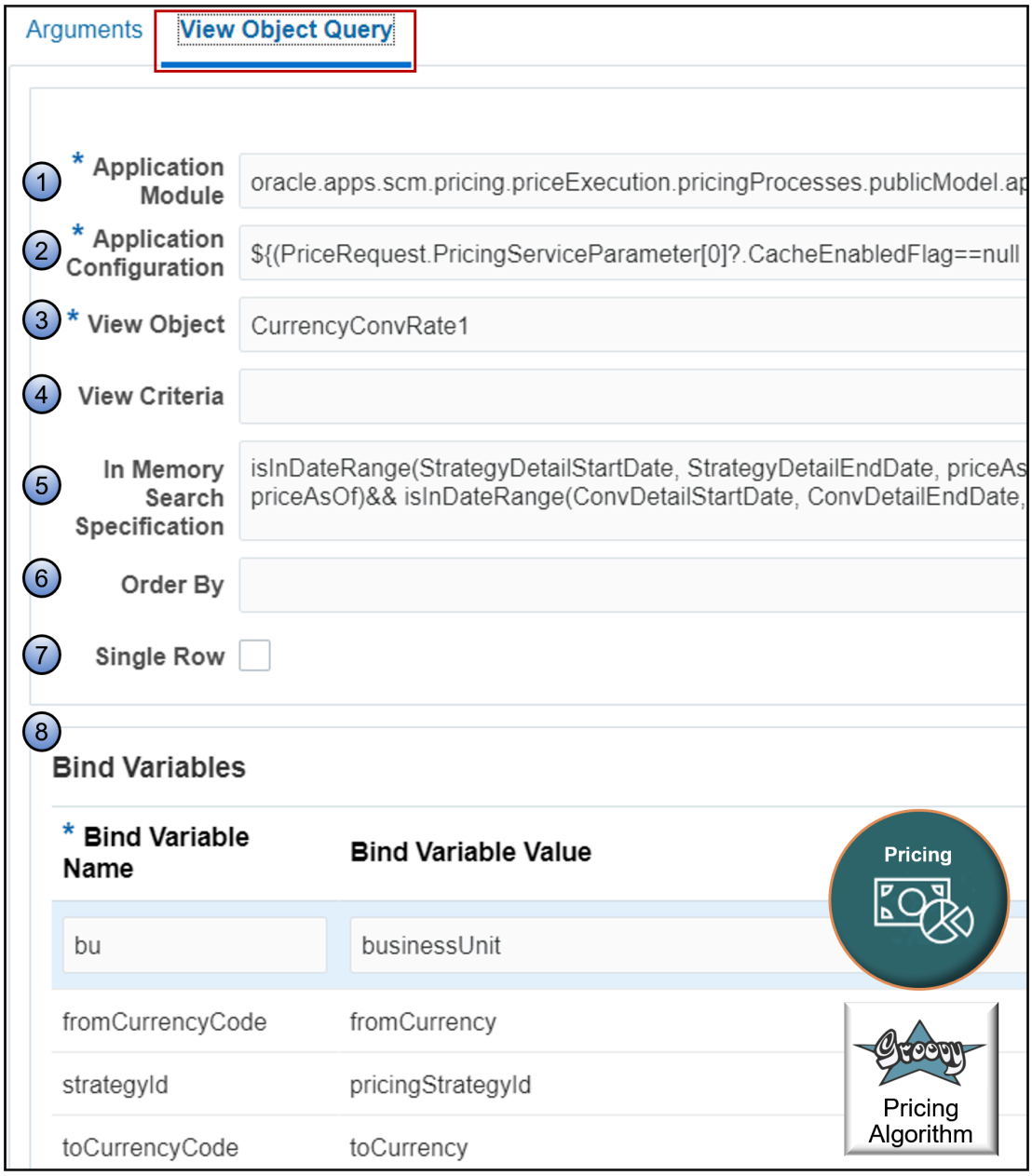
```
finest {'getAdjustedRate Arguments-- AdjType: '+adjustmentTypeCode+' AdjValue: '+adjustmentValue+' convRate:
 '+conversionRate}
if (conversionRate != null && adjustmentValue != null) {
if (adjustmentTypeCode == 'MARKUP_AMOUNT') { conversionRate += adjustmentValue}
if (adjustmentTypeCode == 'MARKDOWN_AMOUNT') { conversionRate -= adjustmentValue}
if (adjustmentTypeCode == 'MARKUP_PERCENT') { conversionRate += conversionRate * adjustmentValue / 100}
if (adjustmentTypeCode == 'MARKDOWN_PERCENT') { conversionRate -= conversionRate * adjustmentValue / 100}
}
if (adjustmentTypeCode == 'OVERRIDE') { conversionRate = adjustmentValue}
return conversionRate
```

Example of View Object Lookup

Here's an example of a step that uses a view object lookup to call a function. To start, add your arguments:



Then click View Object Query, and set up the query. Here's a typical view object query. Note that view object query and view object lookup mean the same thing:



Here's the code that this example uses:

1. Application Module:

Example	Description
`oracle.apps.scm.pricing.priceE`	Identify the application module.

Example	Description

A pricing algorithm is part of the pricing process so use PricingProcess:

- `oracle.apps.scm.pricing.priceExecution`. The path to PricingProcess.
- `applicationModule.PricingProcessAM`. Specifies to use PricingProcessAM, where AM means application module.

Learn about Application Module and Application Configuration. For details, see *Select Choices in Pricing Matrixes*.

2. Application Configuration:

Example	Description
`${(PriceRequest.PricingService || PriceRequest.PricingServicePar 'PricingProcessAMShared' : 'PricingProcessAMLocal'}`	You must use this configuration for most algorithms.

3. View Object:

Example	Description
`CurrencyConvRate1`	Enter the name of the view object.

Pricing comes predefined to use CurrencyConvRate1 for the GetCurrencyConversionRate function.

You can also use the fully qualified name. Here's an example that identifies the name of the view object when you use a descriptive flexfield.

```
oracle.apps.flex.scm.pricing.priceExecution.pricingProcesses.priceListCharges.view.PriceListChargeDFFVO
```

where

- `oracle.apps.flex.scm.pricing.priceExecution.pricingProcesses.priceListCharges`. The path to the priceListCharges entity of the pricingProcesses application module for flexfields.
- `PriceListChargeDFFVO`. Name of the view object, where DFFVO means descriptive flexfield view object.

4. View Criteria:

Example	Description
No value for this example.	Criteria that determines whether you can query the view object according to the bind variables.

Example	Description

5. In Memory Search Specification:

Example	Description

`isInDateRange(StrategyDetailSt`
` StrategyDetailEndDate,`
` priceAsOf) &&`
`isInDateRange(ConvListStartDat`
` ConvListEndDate,`
` priceAsOf)&&`
`isInDateRange(ConvDetailStartD`
` ConvDetailEndDate,`
` pricingDate)`
	Add search specifications that filter results of the view object that's currently in memory.

This example includes three isInDateRange functions that specify how to search.

The first isInDateRange filters according to pricing strategies:

- `isInDateRange`. Name of the search specification.
- `StrategyDetailStartDate`. Filter results so they only include pricing strategies that start on or after this date.
- `StrategyDetailEndDate`. Filter results so they only include pricing strategies that start on or before this date.
- `priceAsOf`. Filter results so they only include records where Pricing used the strategy to price an item on a specific date.

The second isInDateRange filters according to currency conversion lists:

- `isInDateRange`. Name of the search specification.
- `ConvListStartDate`. Filter results so they only include currency conversion lists that start on or after this date.
- `ConvListEndDate`. Filter results so they only include currency conversion lists that end on or before this date.
- `priceAsOf`. Filter results so they only include records where Pricing used the conversion list to price an item on a specific date.

The third isInDateRange filters according to currency conversion details:

- `isInDateRange`. Name of the search specification.
- `ConvDetailStartDate`. Filter results so they only include currency conversion details that start on or after this date.
- `ConvDetailEndDate`. Filter results so they only include currency conversion details that end on or before this date.

- `pricingDate`. Filter results so they only include records where Pricing used the conversion details to price an item on a specific date.

The specifications are cumulative. For example, this specification filters according to pricing strategy, and currency conversion list, and conversion detail.

Assume the search specifications filter to return only pricing strategies a and b, currency conversion lists c and d, and conversion detail e. If records 1, 2, and 3 include pricing strategies a and b, and if records 7 and 8 include currency conversion lists c and d, and if record 9 includes conversion details e, then the search results will include records 1, 2, 3, 7, 8, and 9.

6. Order By:

Example	Description
No value for this example.	Sort the records of the data set in ascending order or descending order, according to one or more columns.

For example, to use precedence when sequencing the query results according to prices on the pricing strategy, set Order By to Precedence.

Order By affects performance. Don't use it unless you really need it.

For details, see the Data Sets section of this topic.

7. Single Row:

Example	Description
Disabled for this example.	Check whether the view object can return only one row. For example, a price list includes only one Price List Id. If you query according to Price List Id, then enable Single Row. Enabling this option decreases processing time because the algorithm proceeds immediately after it receives the one record instead of searching through all possible records.

8. Bind Variables:

Example	Description
Bind Variable Name. For example, fromCurrencyCode.	Name of the bind variable that you add to the view object.
Bind Variable Value. For example, fromCurrency.	Use a fixed value, Groovy expression, or argument name. In this example, fromCurrency is the name of argument for this function.

If you add a lookup for a hierarchical view object, then you can click the Hierarchical Query tab, then set properties:

Properties	Description
Connect By	Specify how to connect.
Query Result Alias	Specify the alias.

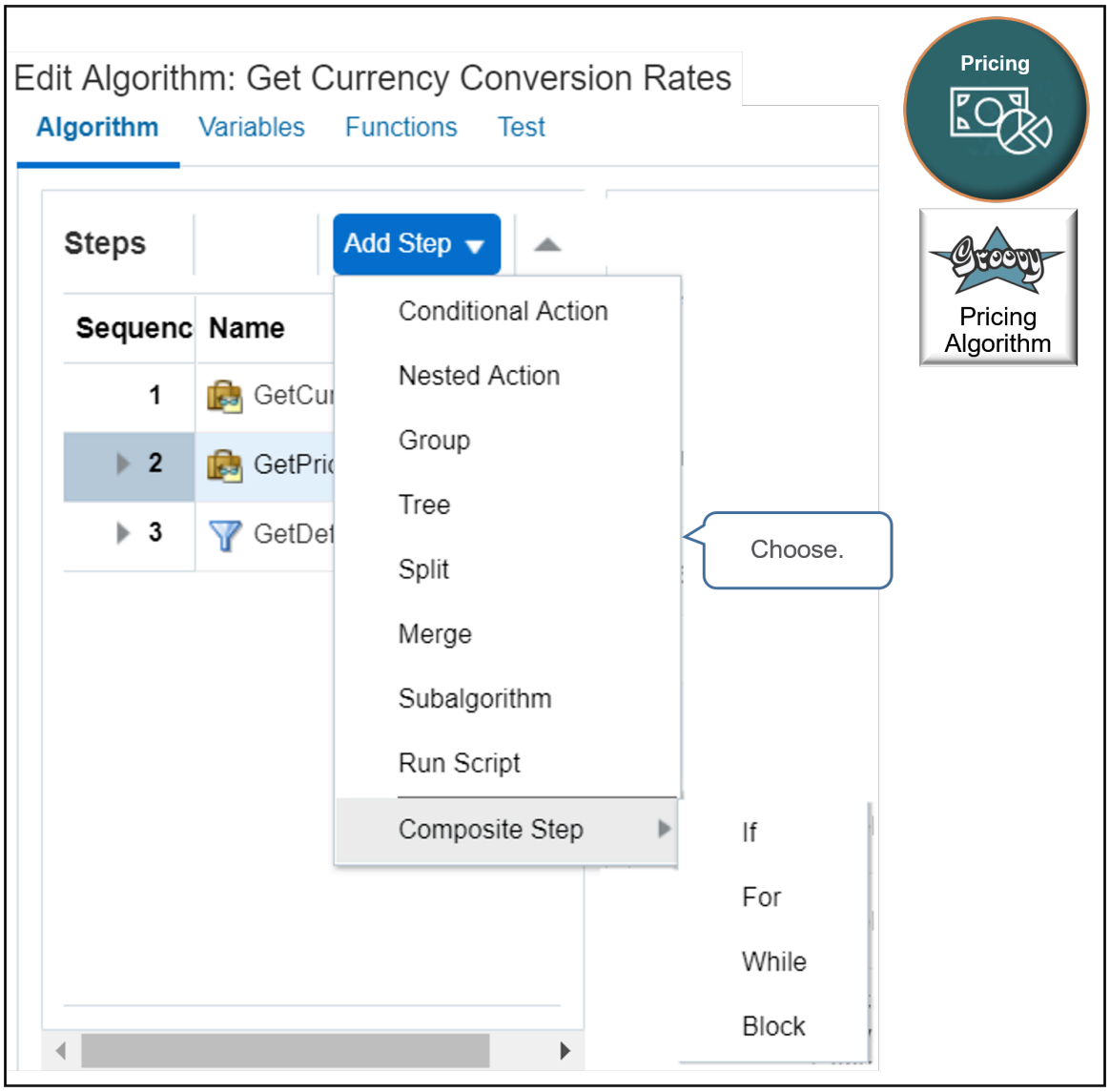
Related Topics

- Overview of Oracle Pricing
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Pricing Algorithm Steps

Manage Pricing Algorithm Steps

Set up logic for your pricing algorithm steps.

Here are the types of steps that you can add:



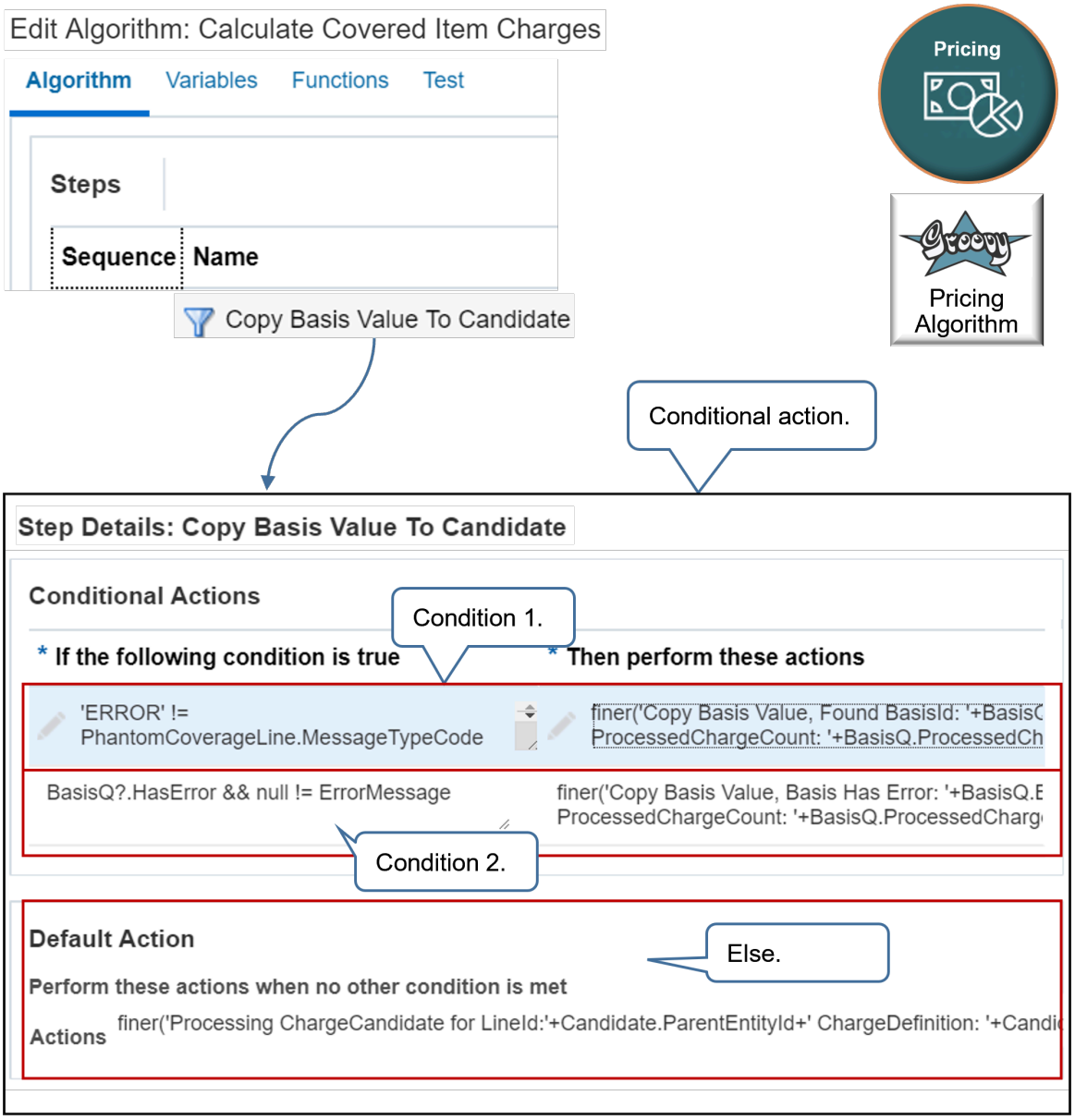
You typically use Conditional Action, Nested Action, Subalgorithm, or Group.

Conditional Action Step

If the primary data set meets the condition you specify, then a conditional action step does an action on all rows in the set. Here's the structure a conditional action step uses.

```
If (condition 1) { . . . }
else if (condition 2) { . . . }
else if (condition 3) { . . . }
. . .
else { default }
```

Here's some example conditions from the Copy Basis Value to Candidate step of the predefined Calculate Covered Item Charges algorithm:



In the next example, examine a step that calculates a rollup charge for a configured item. Here's what it does:

- Evaluates each row of the primary data set.
- Iterates through the charge for the root item of the configured item, and the charge for each child item.
- Creates a rollup charge for each charge definition, as necessary.

Here's a more detailed description of what it does:

- Gets charges from the PriceRequest payload.
- If `Charge.RollupFlag = true`, which indicates that the payload contains at least one rolled up charge, then create a RollupCharge variable in the payload.
- Populate the RollupCharge variable with rolled up charges for each child item of the configured item.

- Set the OutputStatus to SUCCESS on the ServiceParam data set to indicate that this step successfully created the rollup charges.

Examine a step that includes a conditional action:

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Calculate Rollup Charges for Prepriced Orders**, then click **Actions > Create Version**.
4. In the Name column, click the new version of Calculate Rollup Charges for Prepriced Orders that you just created.
5. On the Edit Algorithm page, click the **Create Rollup Charge** step.
6. In the Step Details area, in the Condition window, examine the condition:

Condition	Description
`!Charge.RollupFlag && 'LINE' == Charge.ParentEntityCode && ('ROOT' == Line.ItemType || 'COMPONENT' == Line.ItemType)`	Set the exit criteria for this step. In this example, the condition sets the OutputStatus to SUCCESS in the ServiceParam data set to indicate that the step successfully created the rollup charges.

7. In the Step Details area, notice the parts of the Execute Condition area:

Part	Description
Local Variable	Declare variables that apply only to the code you write in the Conditional Actions and Default Action areas.
Conditional Actions	Set up conditional logic that Pricing applies for the step.
Default Action	If the conditional action doesn't evaluate to true, then set up the logic to use for the default action step.

8. In the Local Variables area, notice the variables:

Variable Name	Description
Rollup	Identifies each charge when Pricing iterates through the code in the Action window in the Conditional Actions area.
RollupChargeCounter	Counts the iterations.

9. In the Conditional Actions area, notice the condition:

True Condition	Description
RollupCharge.size() = 0	Specifies the condition.

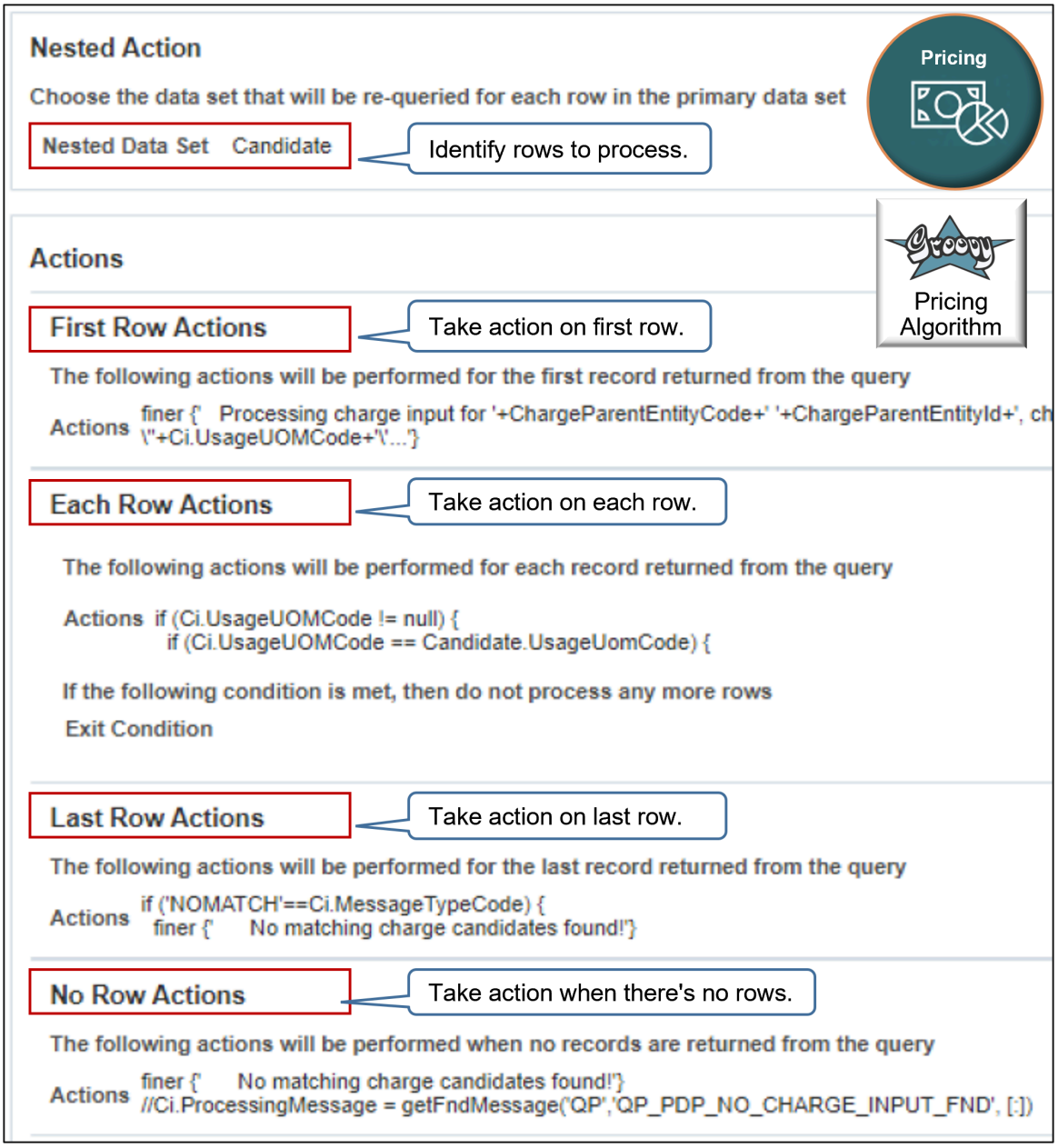
10. In the Conditional Actions area, examine the code in the Actions window.

    This code gets charges for each child item of the configured item, then populates the RollupCharge variable with the rolled up charge for the child. Pricing iterates through the code for each child item until it finishes pricing the child.
11. Click **Add Condition > True Condition**.

    Notice that Pricing added a new condition in the Conditional Actions area. If you add more than one condition, then Pricing evaluates each condition in the sequence that the Conditional Actions area displays them. Click Move Up or Move Down to modify the sequence.
12. In the Conditional Actions area, click **Delete**.
13. On the Manage Algorithms page, delete the In Progress version of pricing algorithm Calculate Rollup Charges for Prepriced Orders.

Nested Action Step

A nested action step does a different action on each row of the data set:



Note

- Use the Nested Data Set to identify the rows that the nested action processes.
- Here's what the actions do:

An Action on This Row	Applies the Action on This Record
First row action.	The first record that the query returns.
For each row action.	Each record that the query returns, including the first record.

An Action on This Row	Applies the Action on This Record
Last row action.	The last record that the query returns.
No row action.	No record. Applies when the query doesn't return any records.

Here's the structure a nested action step uses.

```
For each Nested Set Row {
 If First Row {. . .}
 For Each Row {. . .}
 If Last Row {. . .}
 If No Row {. . .}
}
```

Assume you must implement a condition:

- If more than one order line on a sales order references the same item, then apply a discount on the item.

You can write a pricing algorithm step that applies action x to the first row of the data set, and action y to the other rows of the data set.

Examine a step that uses a nested action to find a charge on the shipping charge list for each item on each order line, then writes the candidates to the output SDO:

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Calculate Shipping Charges**, then click **Actions > Create Version**.
4. In the Name column, click the new version of Calculate Shipping Charges that you just created.
5. On the Edit Algorithm page, click the **Find Shipping Charge Candidates** step.
6. In the Step Details area, in the Condition window, examine the condition:

Condition	Description
`'SUCCESS' == ServiceParam.OutputStatus && 'ERROR' != Line.MessageTypeCode && 'ORDER' == Line.LineCategoryCode && ('STANDARD' == Line.ItemType || 'ROOT' == Line.ItemType) && Line.AppliedShippingChargeList = null && finer(AlgmName + ': finding charge candidates for line ' + Line.LineId + ' (' + Line.InventoryItemId + ')') == null`	Set the exit criteria for this step. The condition sets the OutputStatus to SUCCESS in the ServiceParam data set to indicate that this step successfully found a charge on the shipping charge list for each item on each order line, and then wrote the candidates to the output SDO.

7. In the Nested Action area, notice that you can choose the data set that this step will query each time it processes a row in the primary data set.

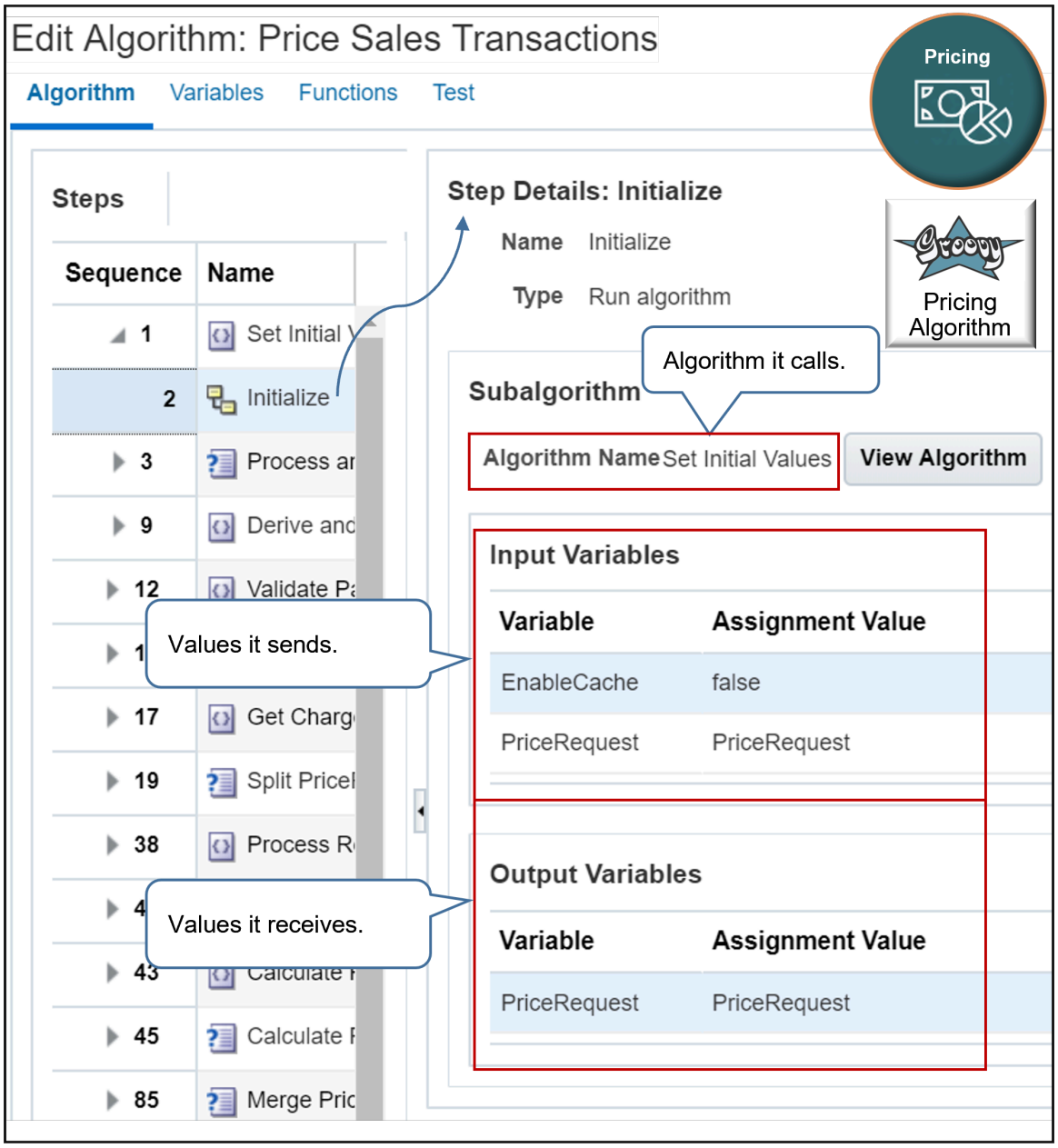
   In this example, notice that the step comes predefined to use the Charge data set because it includes the data that the step requires to find a charge on the shipping charge list for each item, such as the PricingStrategyId to use for the order line, the SellingBusinessUnitId for the order header, and so on.
8. In the Actions area, click **Add Action**, then notice that you can specify a different action for each row that the query returns.

   For example, choose First Row Actions to add an action that this step performs only on the first row that the query returns.
9. In the First Row Actions area, in the Actions window, examine the code, and notice that it finds a charge on the shipping charge list for each item that the first row references.

   To simplify viewing, copy the code to the clipboard, then paste it into a word processing application.
10. In the No Row Actions area, in the Actions window, examine the code, and notice that it handles a situation where the step can't locate a shipping charge list for the item.
11. On the Manage Algorithms page, delete the In Progress version of the Calculate Shipping Charges pricing algorithm.

Subalgorithm Step

A subalgorithm step calls another pricing algorithm. Here's an example:



Note

- The Initialize step of the predefined Price Sales Transactions algorithm calls the Set Initial Values algorithm.
- Use the Algorithm Name in the Subalgorithm area to choose the subalgorithm.
- Use input variables to specify values that the calling algorithm sends to the subalgorithm.
- Use output variables to specify values that the subalgorithm sends to the calling algorithm.
- The Step Details area adds variables that the subalgorithm contains when you set the Algorithm Name. If you require more, create them, then add them.

Group Step

A group step groups the rows in the primary data set according to the value of an attribute, then processes a different action for each group.

For example, the predefined Aggregate Roll Up Charge Components pricing algorithm rolls up charges so your customer can view one value for a configured item. This step creates an aggregate charge component that sums the individual charge components for a rollup charge.

Examine a step that finds a charge on the shipping charge list for each item on each order line, and then writes the candidates to the output SDO:

1. Go to the Pricing Administration work area.
2. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Aggregate Roll Up Charge Components**, then click **Actions > Create Version**.
4. In the Name column, click the new version of Aggregate Roll Up Charge Components that you just created.
5. On the Edit Algorithm page, click the **Create Aggregate Charge Component by Element Code** step.

   This step examines charge components that aren't aggregated and adds them together according to the price element code for each rollup charge. It then creates the corresponding aggregate charge component and sets the RollupFlag for the component to true.
6. In the Step Details area, in the Condition window, examine the condition:

Condition	Description
`'SUCCESS' == ServiceParam.OutputStatus && !(Comp.IsPassedIn ?: false) && Rollup != null && !(Rollup.IsPassedIn ?: false) && 'LINE' == Rollup.ParentEntityCode && (PriceElement == null || Comp.PriceElementCode == PriceElement) && ! Comp.RollupFlag && Comp.AggregateChargeComponentI == null && Comp.SourceTypeCode!='MANUAL_ ADJUSTMENT'`	Sets the exit criteria for this step. The condition sets the OutputStatus to SUCCESS in the ServiceParam data set to indicate that the step successfully added charge components according to the price element code for each rollup charge, then created the corresponding aggregate charge component.

7. In the Group by Attributes area, notice the predefined attributes. This step uses them to arrange rows in the primary data set into groups:

Attribute	Description
Rollup.ChargeId	Groups rows in the primary data set according to the ChargeId attribute in the Rollup data set.

Attribute	Description
Comp.PriceElementCode	Groups rows in the primary data set according to the PriceElementCode attribute of the `Comp` data set.

Each attribute you add in the Group by Attributes area adds the potential for a separate group of rows in the primary data set. Pricing arranges them sequentially according to the attributes you add.

If you add more than one attribute, then Pricing places the group according to the sequence that the Group by Attributes area displays them. It places the first group first, the second group immediately after the first group. It places rows that aren't in any group immediately after the last group.

For example:

- The ChargeId attribute in rows two, five, and seven of the Rollup data set each equal 123.
- The PriceElementCode attribute in rows three, eight, and nine of the `comp` data set each equal 456.

Here's the sequence the step uses to arrange rows in the `comp` primary data set for this example:

- Row two of the Rollup data set
- Row five of the Rollup data set
- Row seven of the Rollup data set
- Row three of the `comp` data set
- Row eight of the `comp` data set
- Row nine of the `comp` data set
- All other rows that aren't in the first group or the second group

8. In the Actions area, click **Add Action**, then notice you can specify different actions for rows in each group:

Actions	Description
Group First Row Actions	Group actions for the first row.
Group Each Row Actions	Group actions for each row.
Group Last Row Actions	Group actions for the last row.

9. On the Manage Algorithms page, delete the In Progress version of the Aggregate Roll Up Charge Components pricing algorithm.

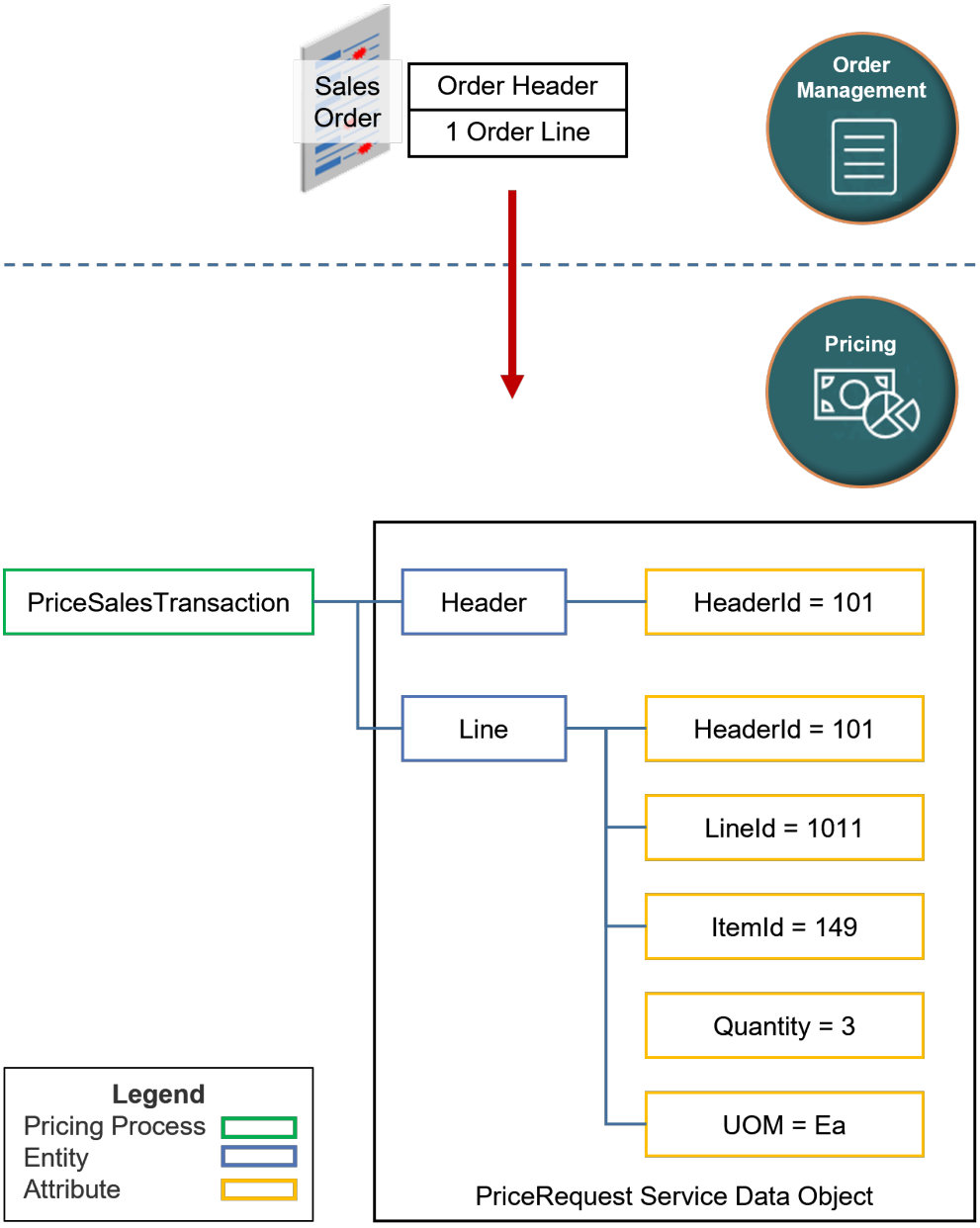
Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Modify Your Pricing Algorithm's Variables
- Overview of Oracle Pricing
- Manage Pricing Algorithms

Data Sets in Pricing Algorithm Steps

Use a data set to filter the set of records that the step processes. All steps use a data set, and all data sets have the same attributes. Some steps include more than one data set.

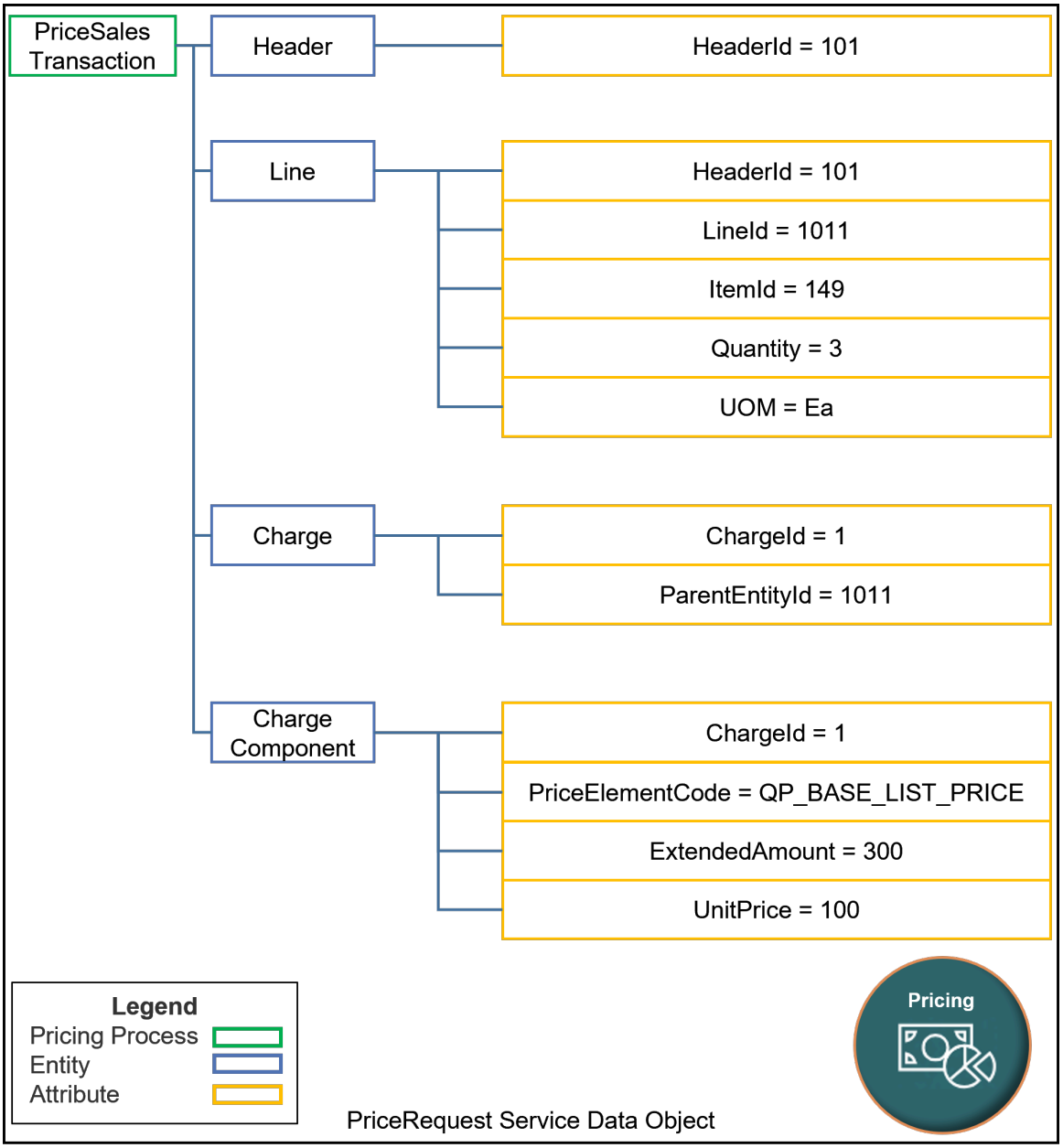
Let's take a look at an example where a user in Order Management manually adjusts price. Order Management sends a request to Oracle Pricing to reprice the item according to the adjustment, and validate it to make sure it falls within your pricing set up. For example, to make sure it doesn't exceed the adjustment that the price list allows:



Note

- The sales order includes one order line.
- Pricing represents the sales order in the PriceRequest service data object.
- The PriceSalesTransaction pricing process sends data in attributes, such as HeaderId.
- The data is flat. It isn't hierarchical. The header and line are at the same level. This means you can reference a line without worrying about including the hierarchy in your reference.

The PriceRequest service data object adds charges:

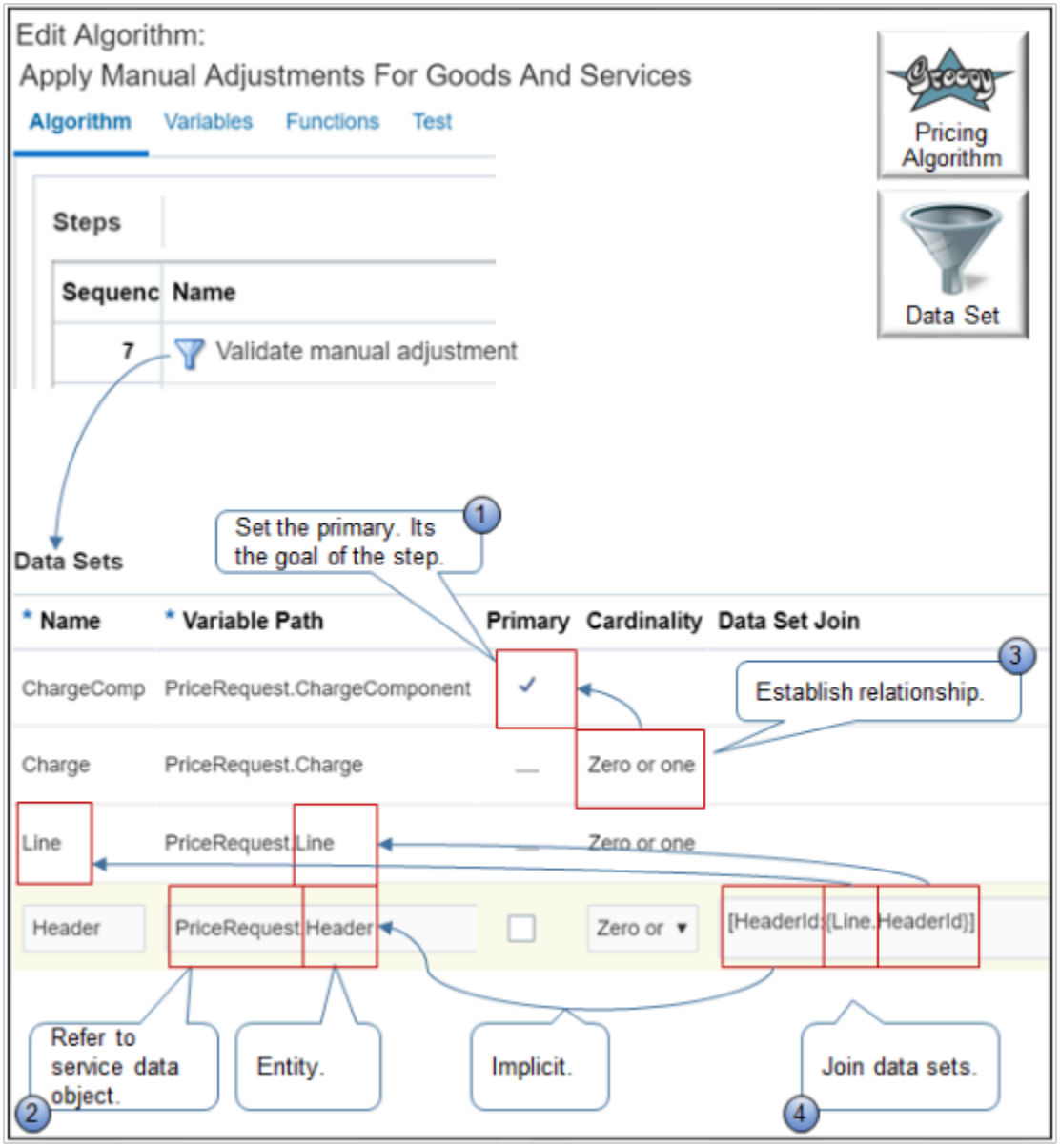


Note

- The sales order includes only one line, and the line includes only one charge.
- Each charge in the PriceRequest service data object includes one charge component.
- Assume some steps of the pricing algorithm have already run and produced values for charges and charge components, such as 300 for ExtendedAmount.

Structure of the Data Sets

Here are some of the data sets from the Validate Manual Adjustment step of the predefined Apply Manual Adjustments For Goods And Services pricing algorithm:



Note

1. The primary data set stores data that meets the goal of the step. For example, if the goal of the step is to calculate a charge, then set ChargeComponent as the primary.

   The primary data set is usually the first data set in the Data Sets area.
2. Use the Variable Path to point to the service data object. For example, `PriceRequest.Header` references the Header entity in the PriceRequest service data object.
3. Use Cardinality to specify the number of records in the nonprimary data set you're adding to the primary data set. For example, `Zero or one` in the Header data set means there is no more than one header for each ChargeComponent.

4. Use the Data Set Join to join the data set you're adding to another data set in the Data Sets area. Use this format.

   ```
   [implicit reference: {target.targetAttribute}]
   ```

   where

   - **implicit reference** is the name of an attribute in the entity of the data set you're currently editing.
   - **target** is the name of the data set you're joining to.
   - **target attribute** is the name of an attribute in the entity that the target data set references.

   For example, join the Header data set to the Line data set.

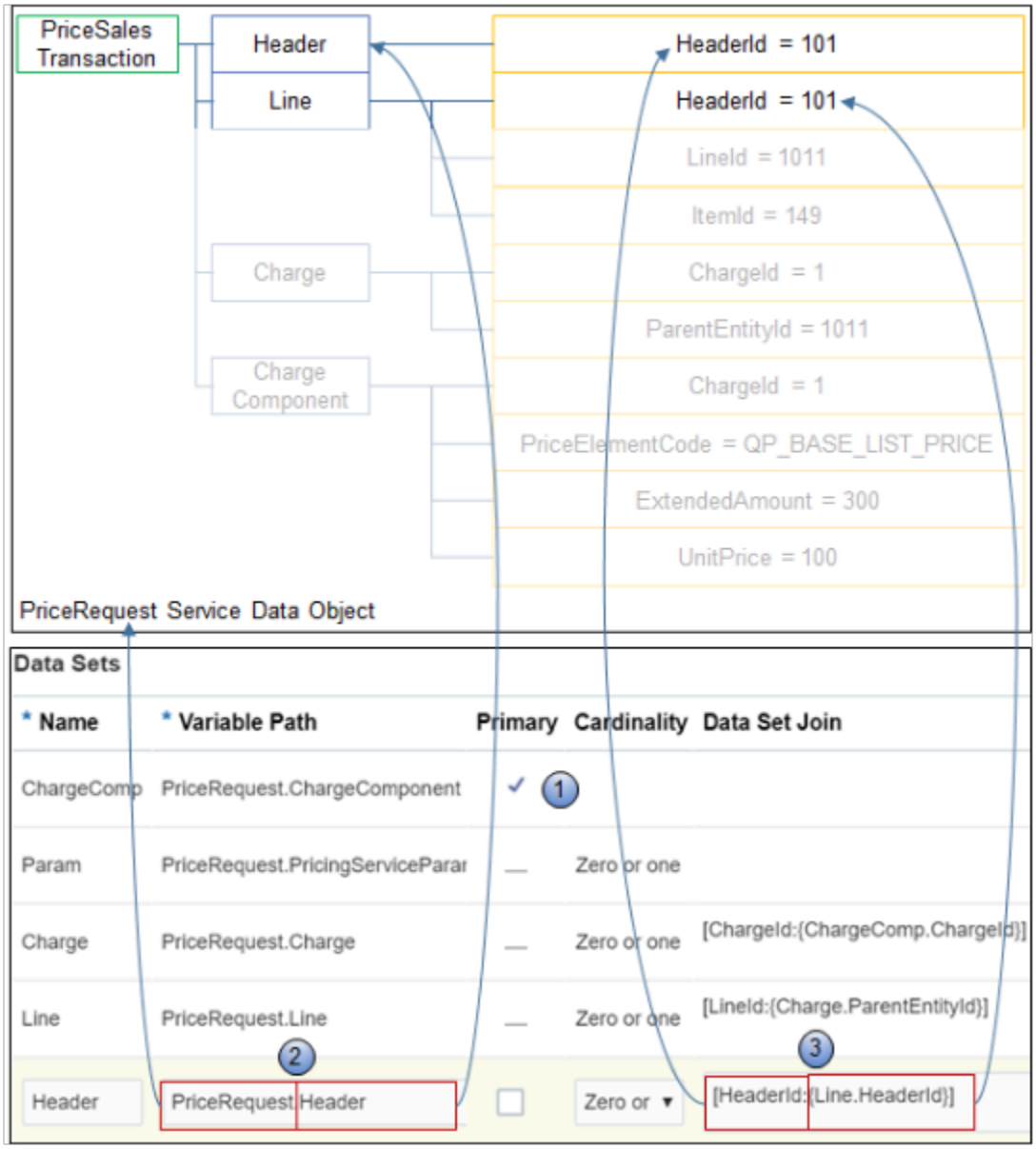
   ```
   [HeaderId: {Line.HeaderId}]
   ```

   where

   - **HeaderId:** is the name of an attribute in the Header entity. It identifies the order header. An attribute you include before the colon is an implicit reference to the entity you specify on the Variable Path of the data set you're currently editing.
   - **Line** is the name of the data set you're joining to.
   - **.HeaderId** is an attribute of the Line entity. The Line data set references the Line entity. An attribute that you include after the dot (.) is an explicit reference to an attribute in the target data set's entity.

     Each order line is part of a sales order. The Line entity includes the HeaderId attribute so the data model knows where the line belongs.

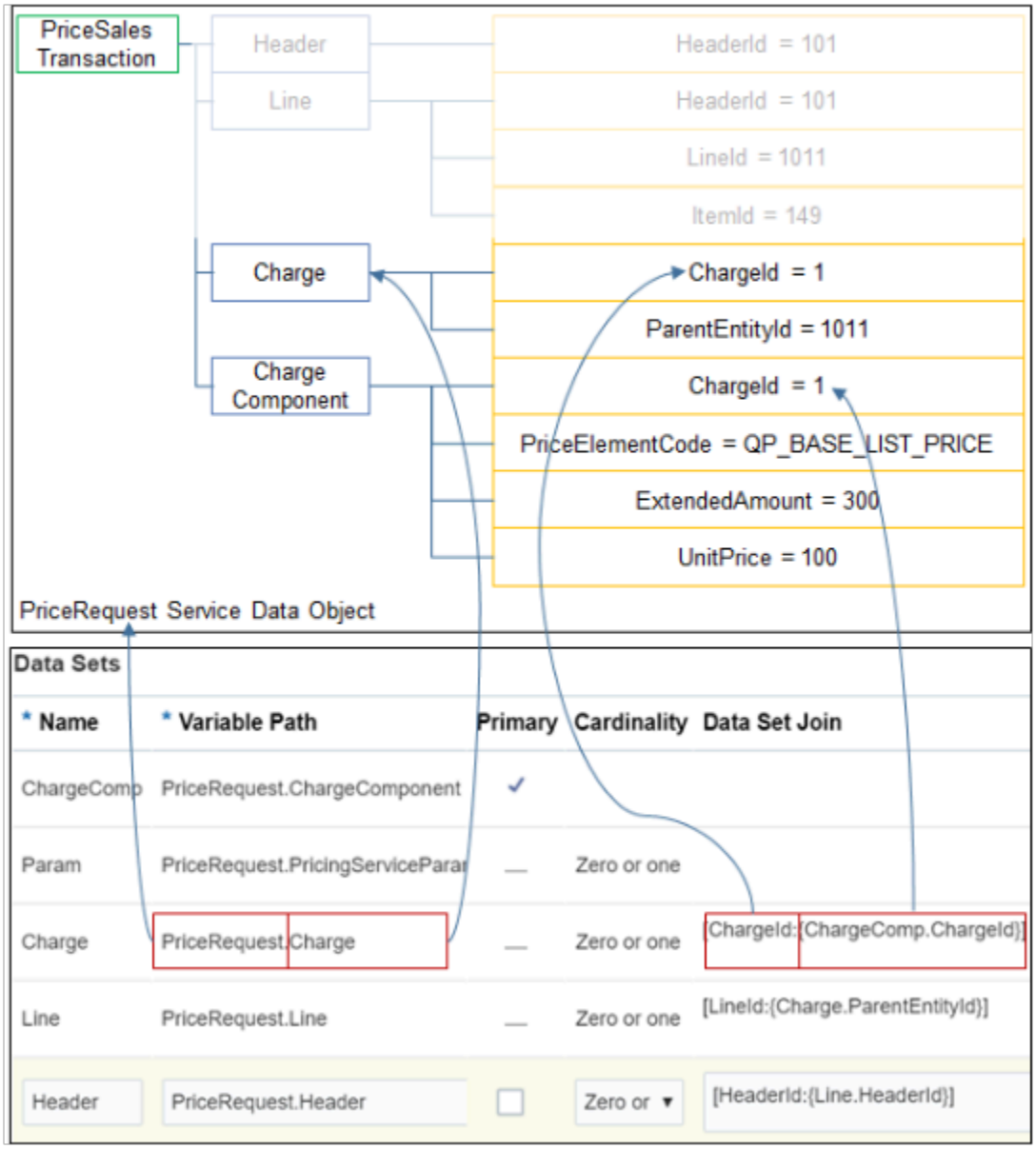
You can map the data set to the service data object:



Note

1. Add the primary data set first. For example, `PriceRequest.ChargeComponent` references the ChargeComponent entity in the PriceRequest service data object.
2. Add a separate data set for each entity. For example, to reference the Header entity, add a data set, name it Header, and set Variable Path to `PriceRequest.Header`.
3. Add attributes to join the data sets.

Add another data set. For example, add the Charge data set:



Note

- Keep adding data sets until you finish adding the ones you need to populate all entities and attributes in the service data object that the algorithm needs to finish the step.

  For example, make sure you populate attributes in conditions and actions that the step uses.
- Use Move Up and Move Down in the Data Sets area to sequence data sets according to dependency. The algorithm processes data sets in the sequence that the Data Sets area displays them.

  For example, the algorithm must calculate charges before it can apply them to the order line. So, arrange data sets so the Charge data set is above the Line data set.

Tip: Use an application to create a diagram of your service data object, such as Microsoft Visio or Powerpoint. Refer to the diagram when you add data sets.

More Join Examples

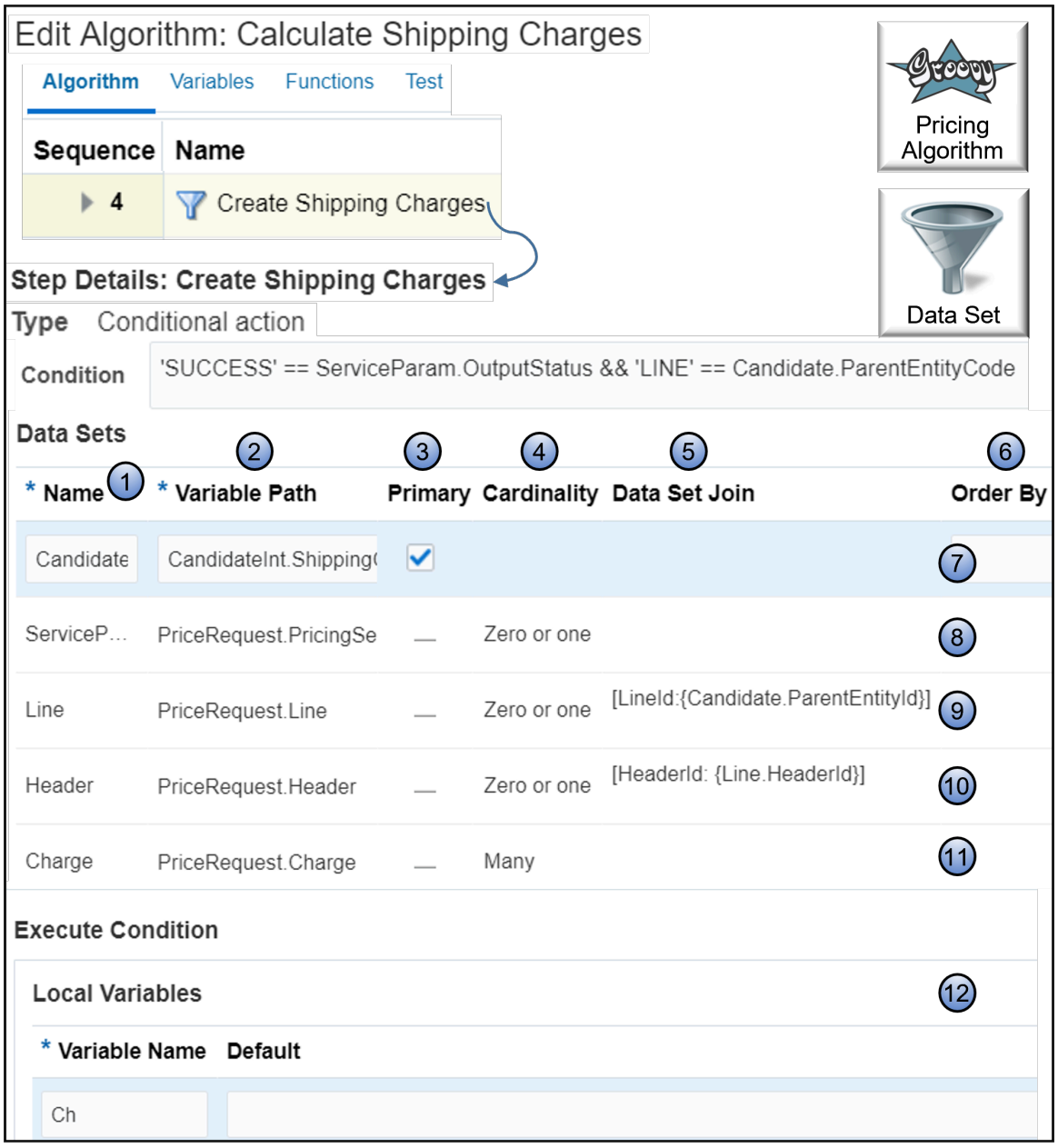
Requirement	Data Set Join	Cardinality
Join one order line to one order header. You match one attribute.	`[HeaderId: {Line. HeaderId}]`	An order line can exist only in one sales order, and a sales order includes only one header, so use: • One • Zero or one
Join a manual price adjustment to a charge. You match more than one attribute. An order can include more than one charge. You must specify all the attributes of the charge so your join can identify the correct charge to use.	`[ParentEntityId: {Charge.ParentEntityId}, ParentEntityCode: {Charge.ParentEntityCode}, RollupFlag: {Charge.RollupFlag}, ChargeAppliesTo: 'PRICE']`	More than one manual adjustment can exist for a charge, so set Cardinality to Many.

Another way to match a manual price adjustment to a charge.
 	
Use a Groovy expression to match the ChargeDefinitionId or the ChargeDefinitionCode.
	`ERROR'!=Mpa.MessageTypeCode &&`
	
	`(Charge.ChargeDefinitionCode==Mpa.Cha`
	`||`
	`Charge.ChargeDefinitionId==Mpa.Charg`
	`&&`
	`Charge.PricePeriodicityCode==Mpa.Pri`
	`&& Charge.MessageTypeCode!='ERROR'`
	`&&`
	
	`Charge.ParentEntityId==Mpa.ChargePar`
	`&&`
	`Charge.RollupFlag==Mpa.ChargeRollupF`
		Many

Create a new charge component. You match one attribute.	`[ChargeId: {Charge.ChargeId}]`	Most service data objects already include a charge and charge component. If you create a new entity or charge component, then set Cardinality to Many so your join can identify the correct one.

Another Data Set Example

Here's a step that includes several typical data sets:



Note

1. **Name**:

Example	Description
Create Shipping Charges	Text that describes the data set. Make sure the name is unique within the algorithm. Use camel case and make the first letter uppercase. For example, MyDataSet.

Example	Description

2. **Variable Path.** An expression that specifies the source for the data set.

   Here's the format you use to create an expression that references a variable:

   - `AlgorithmVariableName.Attribute`

   where

   - `AlgorithmVariableName` identifies the name of an algorithm variable.
   - `Attribute` identifies an attribute of the service data object.

   Here's the expression that the Header data set uses in the example:

   - `PriceRequest.Header`

     The expression specifies to use the PriceRequest variable of this algorithm and the Header attribute of the service data object.

   Here's the format to use to create an expression that references a function:

   - `algorithmFunctionName(argument1.argument2.argument3 . . .)`

   For example:

   - `checkAllowedCurrency(strategyId.currencyCode)`

   This expression specifies to use the strategyId argument and the currencyCode argument of the checkAllowedCurrency function.
3. **Primary.**
   - **Contains a check mark.** Use this row as the primary data set. Each algorithm step processes the set of records that the primary data set contains. You can set only one primary data set for each step.
   - **Doesn't contain a check mark.** Use this row as a nonprimary data set. A nonprimary data set filters the records that the primary data set processes.
4. **Cardinality.** Specify the cardinality between the primary data set and the nonprimary data set:

Cardinality	Description
One	One primary data set to one nonprimary data set. Establishes an inner join.
Zero or One	Zero or one primary data sets to one nonprimary data set. Establishes an outer join.
Many	Zero or many primary data sets to one nonprimary data set.

5. **Data Set Join.** Set up the join that the step uses to filter the nonprimary data set.

   For example, here's a data set join that filters the Line data set:

   - `[LineId:{Candidate.ParentEntityId}]`

   Note

   - Use a Groovy expression that evaluates to a Boolean value.
   - Use attributes from the nonprimary data set without a qualifier.
   - If you use a data set join, then you can't create an optimized index search, which might degrade performance.
   - If you reference a function in the variable path, then you can't use a data set join.

   If the expression is a literal string, then create a data set join. For example:

   - `[ActiveFlag: 'Y', HeaderId:{Line.HeaderId}]`

6. **Order By.** Sort the records of the data set in ascending order or descending order, according to one or more columns. Here's the format you use:

   - `AttributeName modifier, AttributeName2 modifier`, and so on

   where

   - A comma separates each level of the sort.
   - modifier is `(DESC|ASC) (NULLS FIRST|NULLS LAST)`

   Consider this code:

   -

   It does a three level sort:

   a. Sort the data set records in descending order according to NetPrice. It places records that contain a null value for NetPrice last.
   b. Sort records within the NetPrice sort according to the value of the Discount attribute in the order header in ascending order. It places records that contain a null value for Discount first.
   c. Sort records within the Discount sort according to the value of the NetPrice attribute in ascending order. It places records that contain a null value for NetPrice first.

   If you don't specify a modifier, then Pricing uses Ascending.

7. **Candidate Data Set.** Identify records that are candidates for processing in this algorithm step. Pricing comes predefined to use Candidate as the primary for most data sets.

   For example, if an Order Management user adds shipping charges to order line x, then this step will consider whether to apply shipping charges to order line x, depending on other factors that the algorithm considers, such as pricing strategy, pricing segment, and so on.

   Here are the attributes of the Candidate data set you can use:

Attribute	Description
Variable Path	Candidate uses the CandidateInt.ShippingChargeCandidate variable path. - **CandidateInt.** A variable of this pricing algorithm. It stores a value that determines whether the item is a candidate for a shipping charge. - **ShippingChargeCandidate.** An attribute of the service data object. It specifies whether the item is a candidate for shipping charges.
Primary	Candidate typically identifies the unfiltered set of records that this step processes, so Primary contains a check mark.

8. **ServiceParam Data Set.** Pricing comes predefined to use ServiceParam to identify the service data object, and to identify the attributes of the service data object, that this step uses.

   Here are the values that ServiceParam uses in this example:

Attribute	Description
Variable Path	ServiceParam typically uses the PriceRequest.PricingServiceParameter variable path. - **PriceRequest.** A variable of this pricing algorithm. This variable references the Sales web service. - **PricingServiceParameter.** An attribute of the Sales web service.
Primary	Candidate is typically the primary, and it references nonprimary data sets, such as ServiceParam, to filter the primary data set. So Primary doesn't contain a check mark for a nonprimary data set.
Cardinality	Zero or One specifies that there is zero or one ServiceParam to each Candidate.

9. **Line Data Set.** Pricing comes predefined to use Line to identify the attributes on the order lines that this step examines when it filters records in the primary data set.

   Here are the attributes that the Line data set uses in this example:

Attribute	Description
Variable Path	Line typically uses the PriceRequest.Line variable path.

Attribute	Description
	○ **PriceRequest.** A variable of this pricing algorithm. This variable references the Sales service. ○ **Line.** An attribute of the service data object. It identifies an order line.
Cardinality	Zero or One specifies that there is zero or one Line to each Candidate. In this example, each Candidate can reference only a single order line, so the cardinality is one to one. This cardinality also applies to the header because an order line can reference only one header.
Data Set Join	Line typically uses the `[LineId:{Candidate.ParentEntityId}]` join. ○ **LineId.** Identifies the order line. For example, 101. ○ **Candidate.** References the Candidate data set of this step. ○ **ParentEntityId.**

10. **Header Data Set.** Pricing comes predefined to use Header to identify the attributes on the order header that this step examines when it filters records in the primary record set.

    Here are the attributes that the Header data set uses in this example:

Attribute	Description
Variable Path	Header typically uses the PriceRequest.Header variable path. ○ **PriceRequest.** A variable of this pricing algorithm. This variable references the Sales service. ○ **Header.** An attribute of the service data object. It identifies an order header.
Data Set Join	Header typically uses the `[HeaderId: {Line.HeaderId}]` join: ○ **HeaderId.** An attribute that identifies the header. For example, 101. ○ **Line.** References the Line data set of this step. ○ **HeaderId.**

11. **Charge Data Set.** Pricing comes predefined to use Charge to identify the charges that an order line references.

    Here are the attributes that the Charge data set uses in this example:

Attribute	Description
Variable Path	Charge typically uses the PriceRequest.Charge variable path:

Attribute	Description
	○ **PriceRequest.** A variable of this pricing algorithm. This variable references the Sales service. ○ **Charge.** An attribute of the service data object. It identifies one or more shipping charges.
Cardinality	Charge specifies that there are many Candidates to one Charge.

12. **Local Variable.** Define a variable for use only in the condition.
13. **Default Action.** (Not shown in the illustration. Its below Local Variable on the Edit Algorithm page.) Use Groovy to create the condition to run when the flow doesn't meet any of the other conditions you set up.

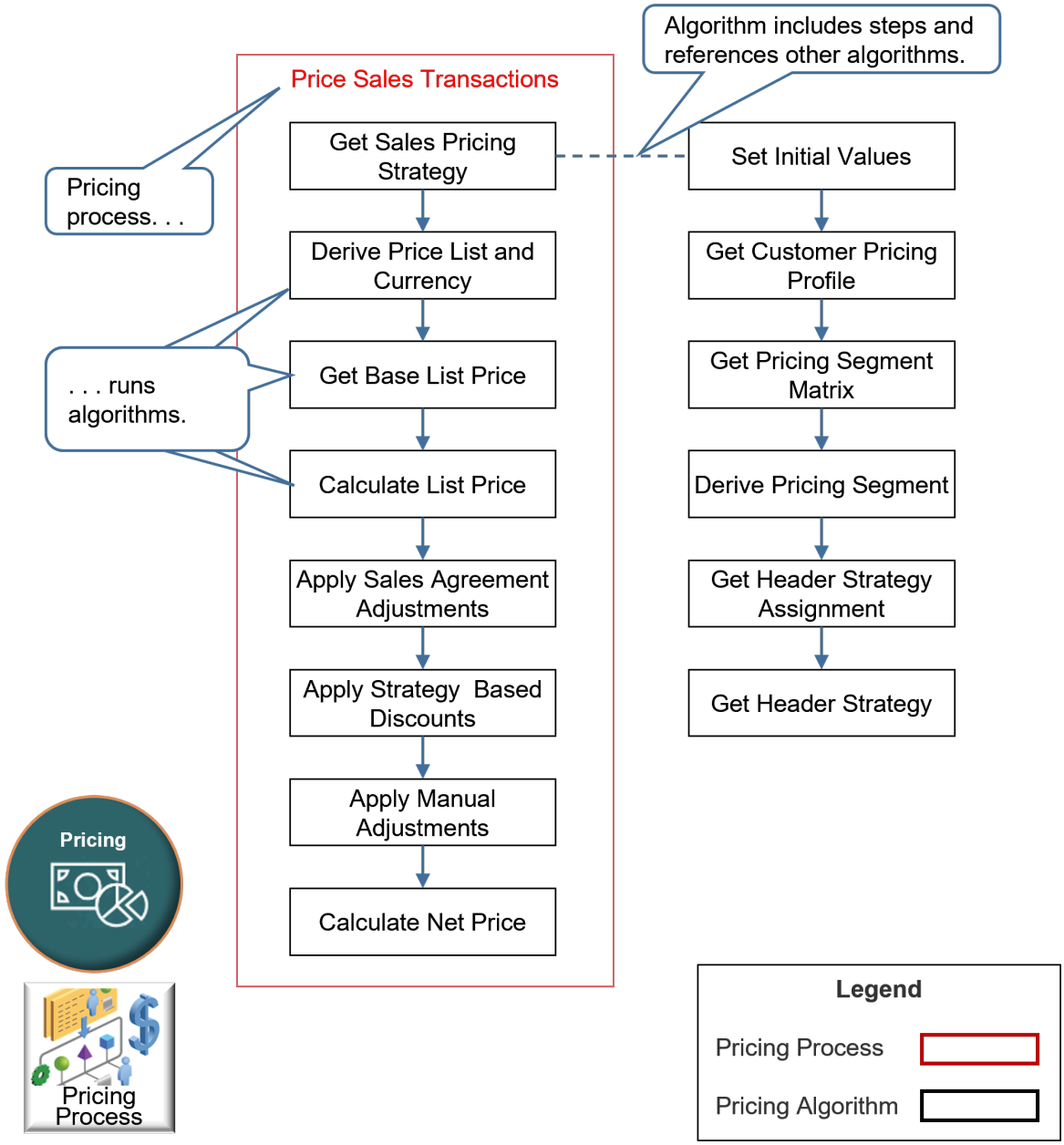
Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Modify Your Pricing Algorithm's Variables
- Overview of Oracle Pricing
- Manage Pricing Algorithms

Pricing Process

A pricing process is an object that runs pricing algorithms to meet the goal of a pricing operation, such as to price a sales transaction.

For example, here's a summary of the predefined Price Sales Transaction pricing process:

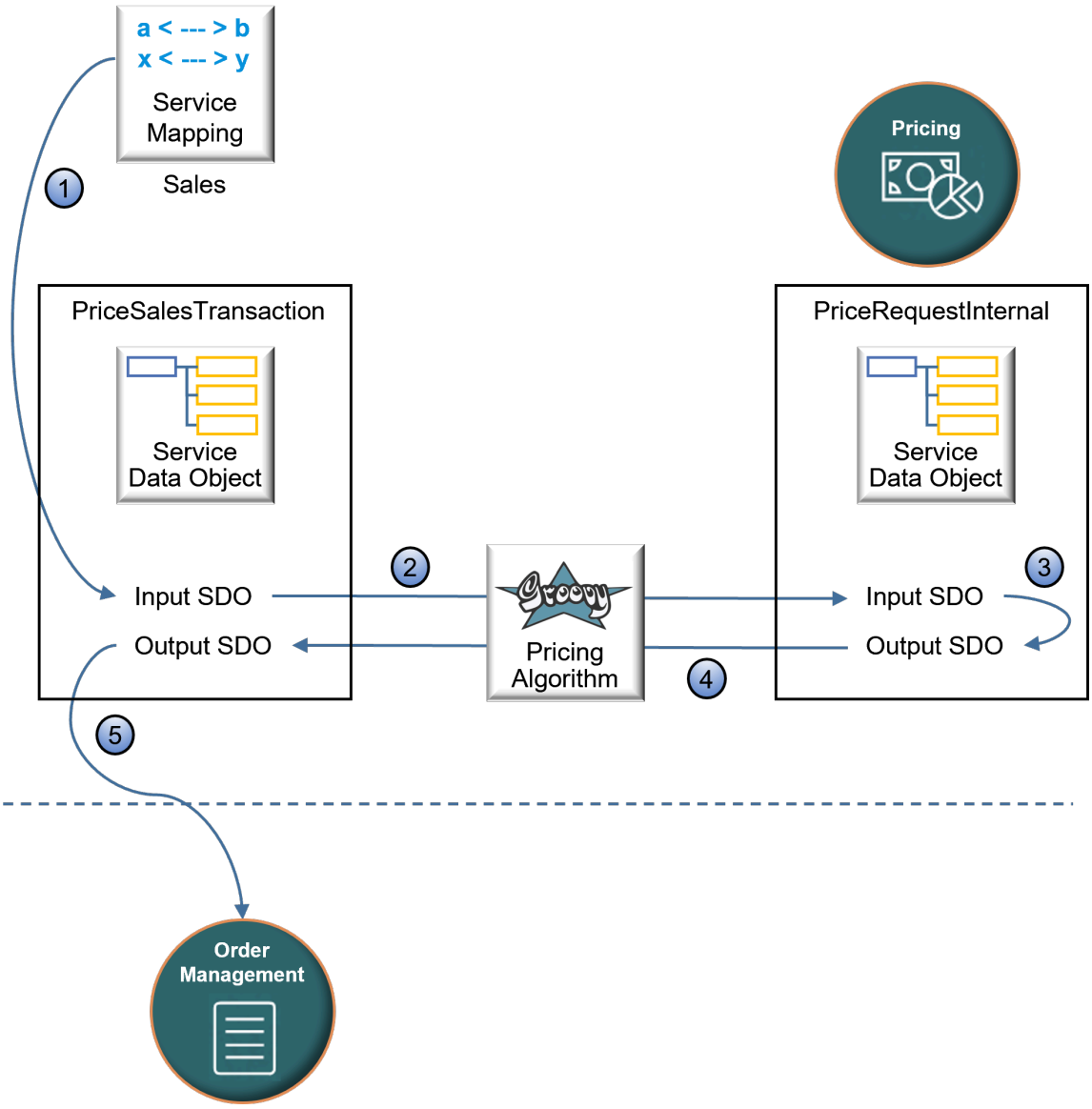


Note

- Pricing comes predefined with a number of pricing processes.
- A pricing process runs pricing algorithms in a sequence.
- The Price Sales Transaction pricing process runs a number of pricing algorithms, such as Get Sales Pricing Strategy.
- Pricing also uses the predefined Calculate Sales Totals and Validate Sales Prices processes.
- Each pricing algorithm contains steps and can reference other pricing algorithms. For example, the Get Sales Pricing Strategy references other pricing algorithms.

All algorithms process the service data object according to the PriceRequestInternal schema in the service mapping. Pricing uses this logic so the values in an entity that one algorithm processes are available to any other algorithm during processing. For example, the chargeCadidate that Get Base List Price calculates is available in Calculate List Price.

Here's an example flow for the Sales service mapping:



Note

1. The Sales service mapping creates the input PriceSalesTransaction service data object.
2. PriceSalesTransaction isn't an internal service data object, so the algorithm implicitly converts PriceSalesTransaction into the input PriceRequestInternal service data object.
3. The algorithm processes the request, then stores results in the output PriceRequestInternal.
4. The algorithm implicitly synchronizes the output PriceRequestInternal to the output PriceSalesTransaction.
5. The Sales service mapping sends the output to Order Management.

   The algorithm discards all the internal entities and attributes it used during processing.

All algorithms use a similar flow.

Detailed Sequence

Pricing comes predefined to use the Price Sales Transaction algorithm and the Calculate Sales Order Totals algorithm to do most of the pricing for a sales order. Here's the detailed sequence it uses:

User Action	Step in Price Sales Transaction	Step in Calculate Sales Order Totals
A. Click Create Order, then enter a value in the Customer attribute.	**B.** Derive Pricing Strategy	**C.** Calculate List Price Total
D. Enter an item in the Select Item field, then search.	**E.** Get Base List Price **F.** Apply Adjustments **G.** Calculate List Price **H.** Apply Discounts **I.** Calculate Net Price **J.** Compute Sales Tax **K.** Get Costs **L.** Calculate Margin	Not applicable
M. Click Add to add an item to an order line.	Not applicable	**N.** Calculate List Price Total **O.** Calculate Discount Total **P.** Calculate Net Price Total **Q.** Calculate Tax Total **R.** Calculate Total Credits **S.** Calculate Shipping Total **T.** Calculate Total PayNow
U. Save, reprice, or submit.	Not applicable	**V.** Repeat steps N through T.
W. Enter an item in the Select Item field, then search.	**X.** Repeat steps **E** through **L**.	Not applicable
Y. Add add an item to an order line.	Not applicable	**Z.** Repeat steps **N** through **T**.
AA. Save, reprice, or submit.	Not applicable	**AB.** Repeat steps **N** through **T**.

For example, assume these steps happen:

1. The Order Entry Specialist enters a value in the Customer attribute:
    - Order Management sends a request to Pricing.
    - Pricing runs the Derive Pricing Strategy step of the Price Sales Transaction algorithm to determine the pricing strategy to use for the customer.
    - Pricing runs the Calculate List Price Total step of the Calculate Sales Order Totals algorithm to set the total price to zero, then sends a reply to Order Management.
    - Order Management displays the strategy in the pricing strategy and segment.
2. The Order Entry Specialist searches for an item when adding an order line:
    - Order Management sends a request to Pricing.
    - Pricing runs the Get Base List Price step through the Calculate Margin step of the Price Sales Transaction algorithm to price the item, then sends a reply to Order Management that includes the item price.
    - Order Management displays the price above the order lines area.
3. The Order Entry Specialist clicks Add to an item to an order line:
    - Order Management sends a request to Pricing.
    - To price the sales order, Pricing runs the Calculate List Price Total step through the Calculate Total PayNow step of the Calculate Sales Order Totals algorithm, then sends a reply to Order Management.
    - Order Management displays the total price for the order line, and the total price for the sales order in the order header.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Pricing Algorithms

Example of How Pricing Algorithms Price Items

Example Pricing Algorithm, Part 1, Get Reference Attributes

Learn how a pricing algorithm prices an item. Use this knowledge to write or modify an algorithm that meets your specific pricing requirements.

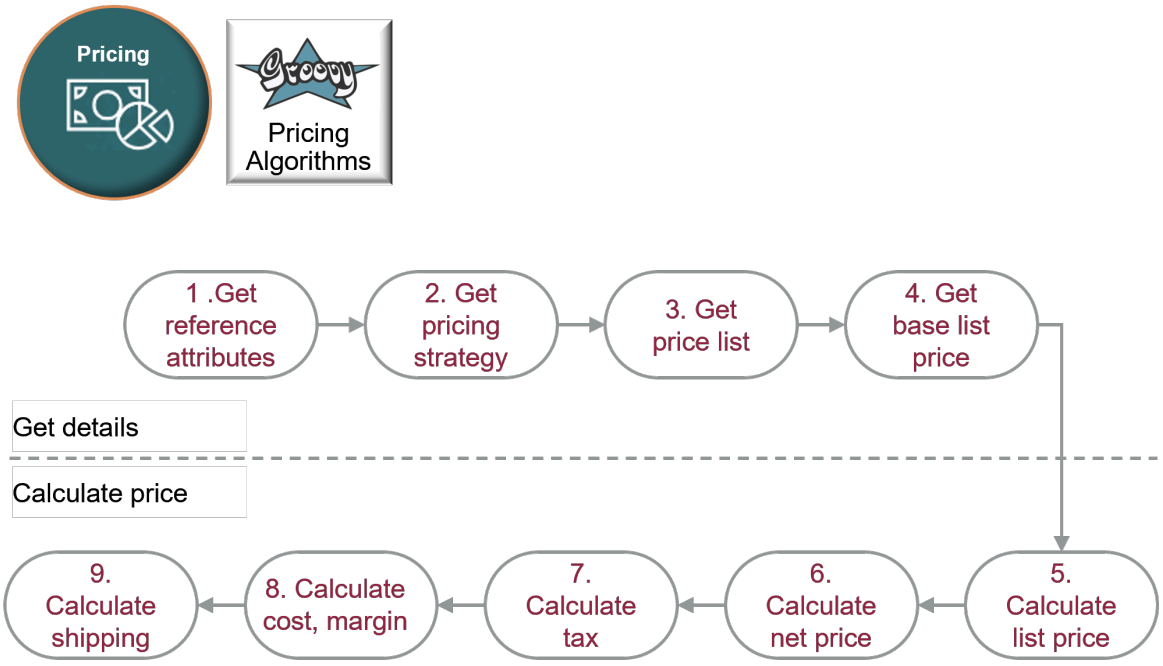
Introduction

Here are some example set ups you can do that meet your specific needs:

Step	Example Usage
1. Get reference attributes.	Get an attribute for the item, such as item weight. Use a descriptive flexfield in Product Management to get a price list.

Step	Example Usage
2. Get pricing strategy.	Use a specific price list for each customer. Use a strategy assignment matrix according to precedence.
3. Get price list.	Use a descriptive flexfield in Trading Community Architecture to determine the price list to use when calculating price.
4. Get base list price.	Make sure the sales order includes a minimum quantity for the item. Override the base list price. Convert a unit of measure.

Here's a summary of the flow that prices an item:



Note

- These steps are a summary of the overall flow. They aren't the names of algorithms.
- Pricing uses profiles, segments, strategies, rules, and other logic to price an item. For details, see *How Pricing Prices Sales Orders*.
- Pricing also uses service mappings, pricing algorithms, and pricing matrixes. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

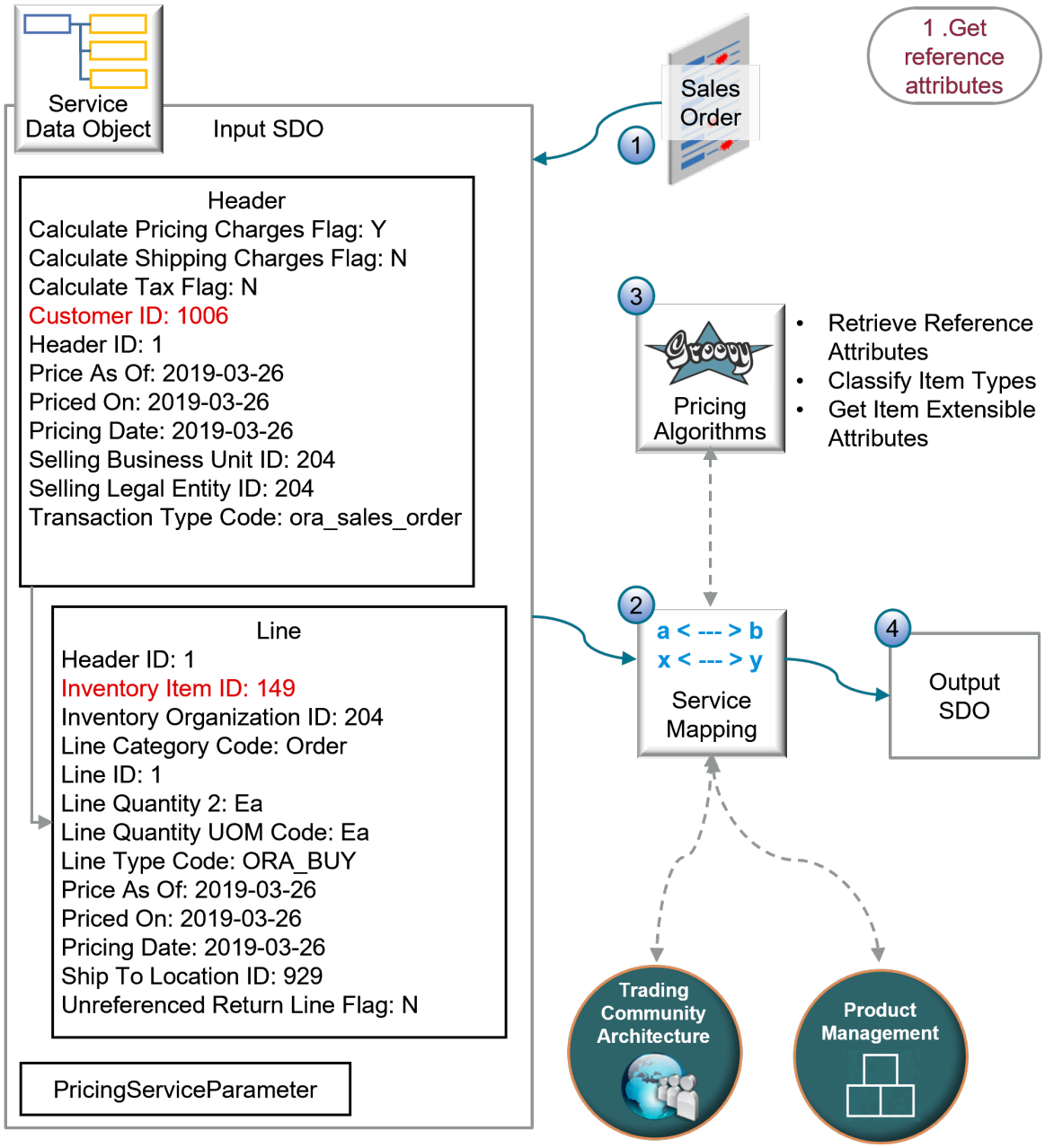
To start, lets consider the first four steps. These steps get the details that pricing needs before it can calculate price.

Get Reference Attributes

Let's take a look at how pricing algorithms price an item. Here's the scenario:

- You need to price an order line that includes item AS54888 Desktop Computer with a quantity of 2 for customer Computer Service and Rentals.
- If your customer orders the AS54888 with a quantity of 1 to 10, give a 10% discount off the base list price. If the quantity is more than 10, give a 20% discount.
- Give an additional $100 discount when Computer Service and Rentals orders the AS54888.
- Don't round the total.

A reference attribute is a kind of attribute that pricing uses to get details from an application. The first step is to get them so pricing can start the flow:



Note

1. Order Management sends a request to pricing to price a sales order. The request includes important attributes that pricing needs to start the process, such as:
    - Customer ID that pricing can use to get customer details from Trading Community Architecture
    - Inventory Item ID that pricing can use to get item details from Product Management
2. The service mapping uses the request to create the input SDO.
3. The algorithms communicate through the service mapping to get customer and item details.
4. The service mapping writes the results to the output SDO.

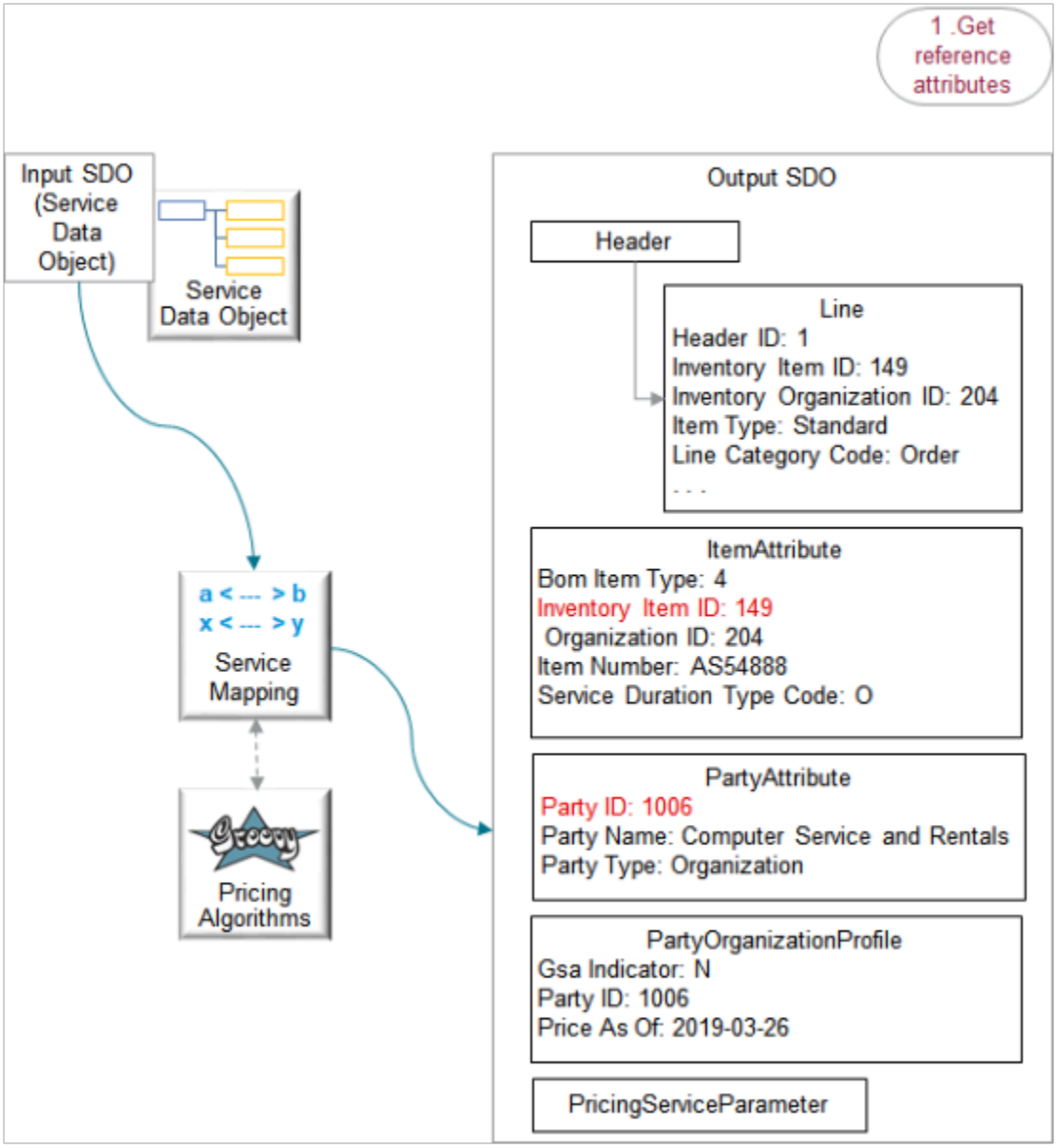
Learn how a service mapping uses Trading Community Architecture and Product Information Mapping. For details, see *Create Your Service Mapping*. You an also learn about party. For details, see *Overview of Displaying Customer Details on Sales Orders*.

Here are the algorithms you use to get reference attributes:

Algorithm Name	Function Name	Description
Retrieve Reference Attributes	getItem	Get the item attributes.
-	getPartyAttr	Get the party attributes.
-	getPartyOrgProfile	Get attributes for the party organization profile.
-	getPartyPersonProfile	Get attributes for the party person profile.
Classify Item Types	getItem	Get item attributes from Product Management.
Get Item Extensible Attributes	getUDAs	Get attributes for user defined items.

These functions are view object lookups. You can use each function to meet your specific needs. For example, if you need to get the weight of the item, use the getItem function to get the value of an extensible flexfield you create, perhaps named Weight.

The service mapping creates the output SDO:



Note

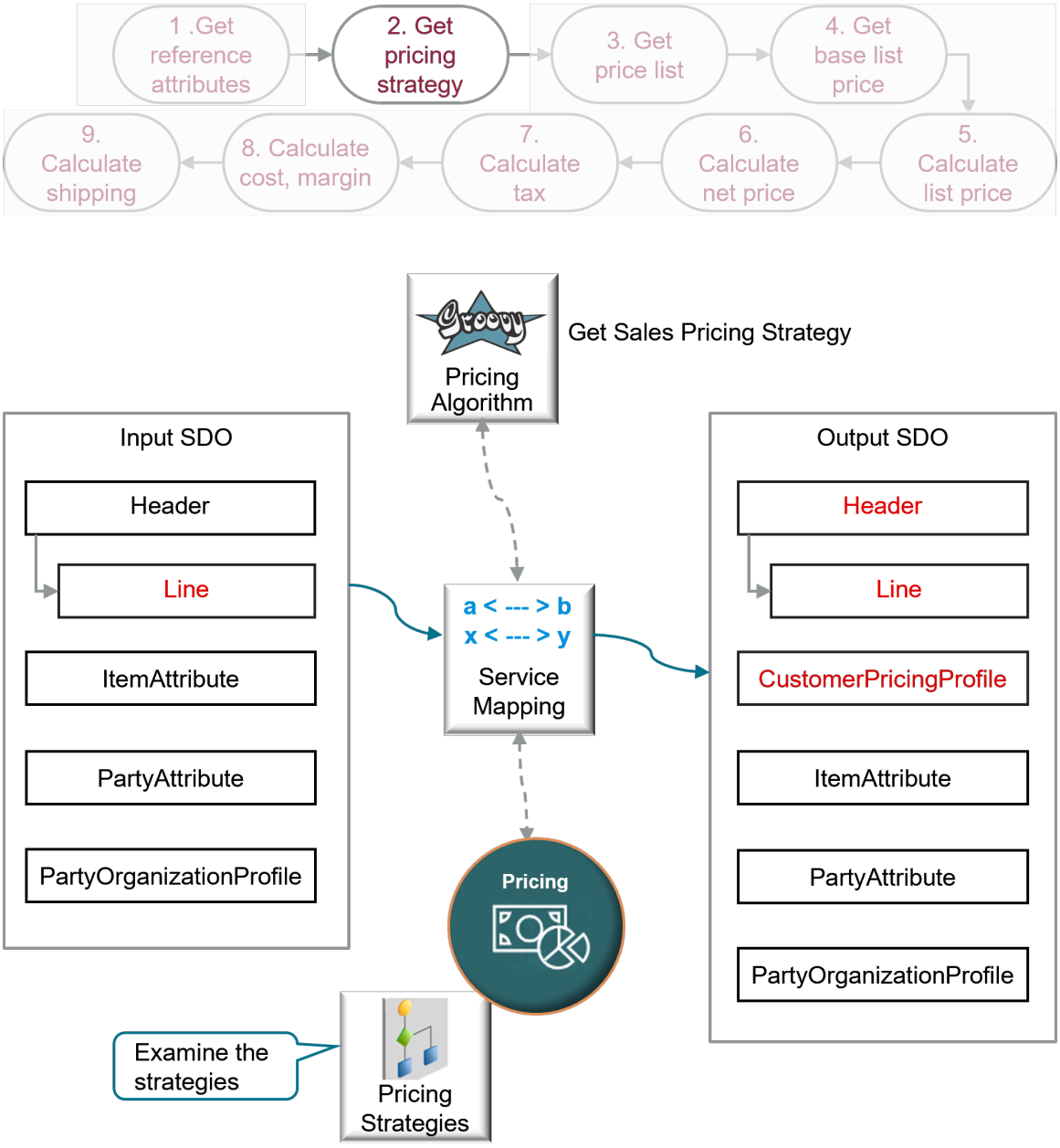
- The service mapping adds the ItemAttribute, PartyAttribute, and PartyOrganizationProfile entities in the output SDO.
- The PartyId for Computer Service and Rentals is 1006, which maps to the CustomerId in the input SDO.
- The InventoryItemId for the AS54888 is 149, and it maps to the InventoryItemId in the input SDO, which is also 149.
- In this example, the output SDO doesn't include a person because PartyType is Organization, not person.

Related Topics

- Pricing Rules

Example Pricing Algorithm, Part 2, Get Pricing Strategy

Determine which pricing strategy to use for your customer for the sales order.



Note

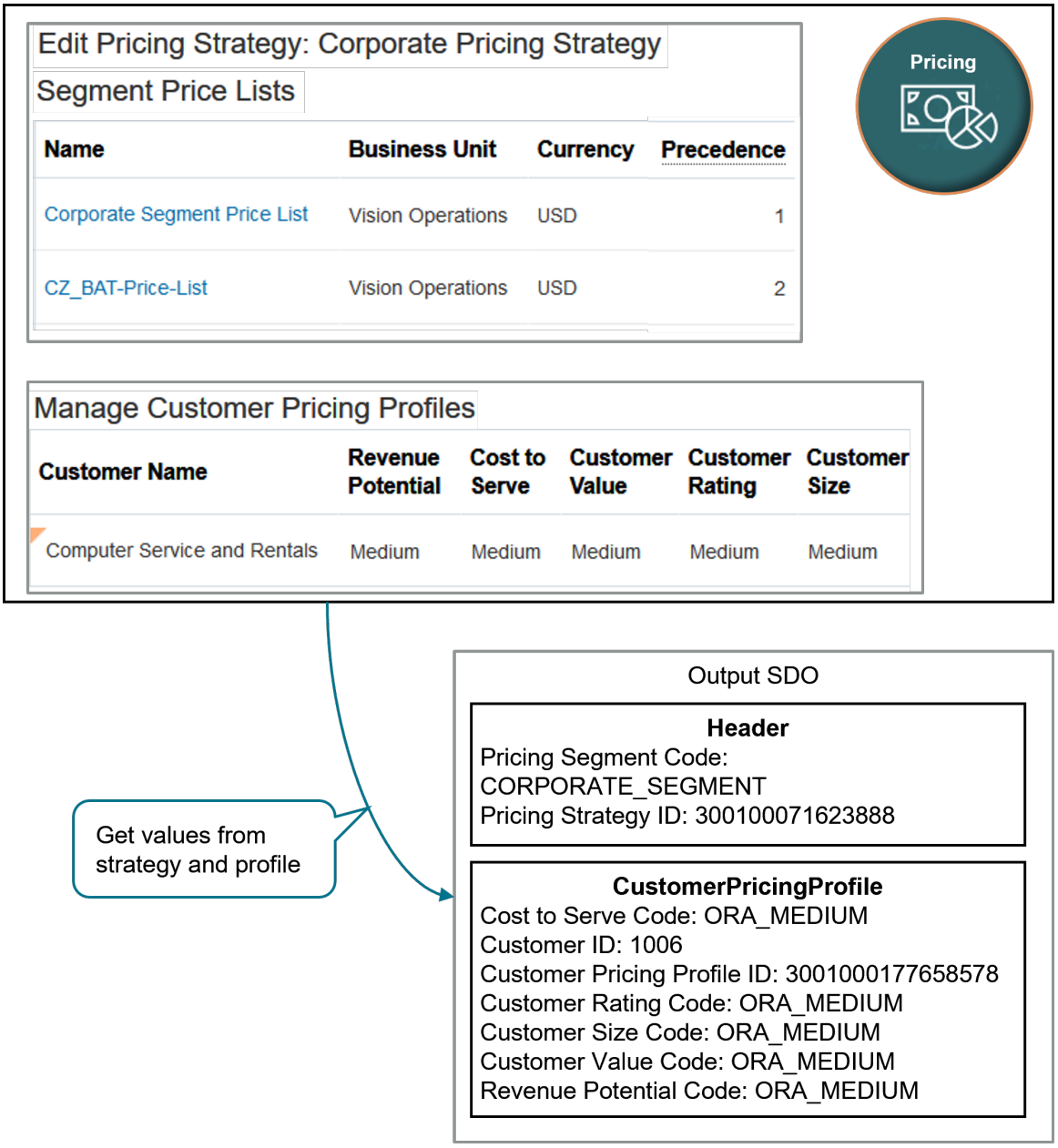
- The output SDO from the Get Reference Attributes step provides the input SDO.
- The algorithm uses some values in the strategy to apply values in the output SDO to the header and the line. For example, it uses the value for the AllowCurrencyOverrideFlag attribute.

- If the algorithm fails for some reason, say, there's a faulty set up in Pricing Administration, then it marks the header as in error.

Here's what the Get Sales Pricing Strategy algorithm does:

What the Algorithm Does	How the Algorithm Does It
1. Get the customer pricing profile so it can identify the header Id.	Examine the profile in Pricing Administration.
2. Determine the pricing segment to use.	Use HeaderId to evaluate the pricing segment matrix, then assign the PricingSegmentCode attribute on the header in the output SDO. In this example, it assigns corporate_segment. It also uses conditions in the segment to create the Pricing Segment Explanation, such as Size=Medium.
3. Identify the strategy assignment.	Examine the strategy assignment matrix according to the pricing context and the type of transaction. In this example, the context is Sales and the transaction is a sales order.
4. Get header values for the pricing strategy.	Evaluate the strategy assignment matrix to identify the strategy, then set PricingStrategyId on the header of the output SDO.
5. Set the pricing strategy on the lines.	Set PricingStrategyId on each of the lines in the output SDO.

The algorithm uses your set ups in the Pricing Administration work area. For example:



Here are details about the attributes in the SDOs. Bold font indicates attributes that the algorithm added to the output SDO during its analysis of your set ups in Pricing Administration. Font that isn't bold indicates values that the algorithm gets directly from the input SDO:

Entity	Attributes in Input SDO	Attributes in Output SDO
Header	-	**AllowCurrencyOverrideFlag: Y** PricingDate: 2019-03-26 @ 08:32:53 **PricingSegmentCode: CORPORATE_SEGMENT**

Entity	Attributes in Input SDO	Attributes in Output SDO
		PricingSegmentExplanation: The applicable pricing segment for this transaction is Pricing Segment=Corporate Segment Precedence=1 because Revenue Potential=Medium Customer Size=Medium Cost To Serve=Medium Customer Value=Medium Customer Rating=Medium **PricingStrategyId: 300100071623888**
Line	HeaderId: 1 InventoryItemId: 149 InventoryOrganizationId: 204 ItemType: Standard LineCategoryCode: Order LineId: 1 LineQuantity2: Ea LineQuantityUOMCode: Ea LineTypeCode: ORA_BUY PriceAsOf: 2019-03-26 @ 08:32:53 PricedOn: 2019-03-26 @ 08:32:53 PricingDate: 2019-03-26 @ 08:32:53 ShipToLocation Id: 929 UnreferencedReturnLineFlag: N	**AllowCurrencyOverrideFlag: Y** **AppliedCurrencyCode: USD** **DefaultCurrencyCode: USD** HeaderId: 1 InventoryItemId: 149 InventoryOrganizationId: 204 ItemType: Standard LineCategoryCode: Order LineId: 1 LineQuantity2: Ea LineQuantityUOMCode: Ea LineTypeCode: ORA_BUY PriceAsOf: 2019-03-26 @ 08:32:53 PricedOn: 2019-03-26 @ 08:32:53 PricingDate: 2019-03-26 @ 08:32:53 **PricingStrategyId: 300100071623888** ShipToLocationId: 929 UnreferencedReturnLineFlag: N
Customer Pricing Profile	-	**CosttoServeCode: ORA_MEDIUM** **CustomerId: 1006** **CustomerPricingProfileId: 3001000177658578** **CustomerRatingCode: ORA_MEDIUM** **CustomerSizeCode: ORA_MEDIUM** **CustomerValueCode: ORA_MEDIUM** **PriceAsOf: 2018-11-10 @ 08:32:53**

Entity	Attributes in Input SDO	Attributes in Output SDO
		RevenuePotentialCode: ORA_MEDIUM

Note

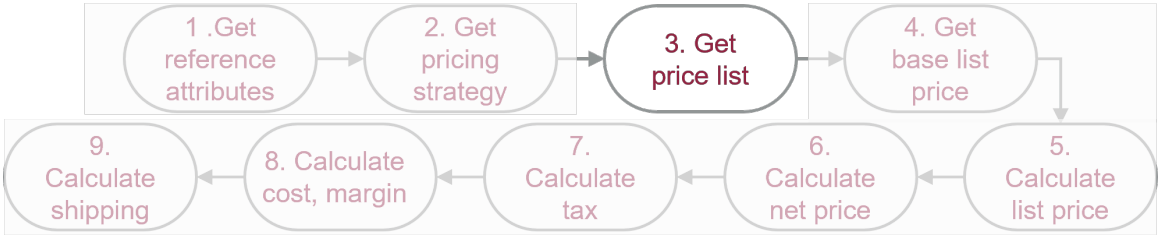
- The value 300100071623888 in PricingStrategyId identifies the Corporate Pricing Strategy.
- The value 3001000177658578 in CustomerPricingProfileId identifies pricing profile Computer Service and Rentals.

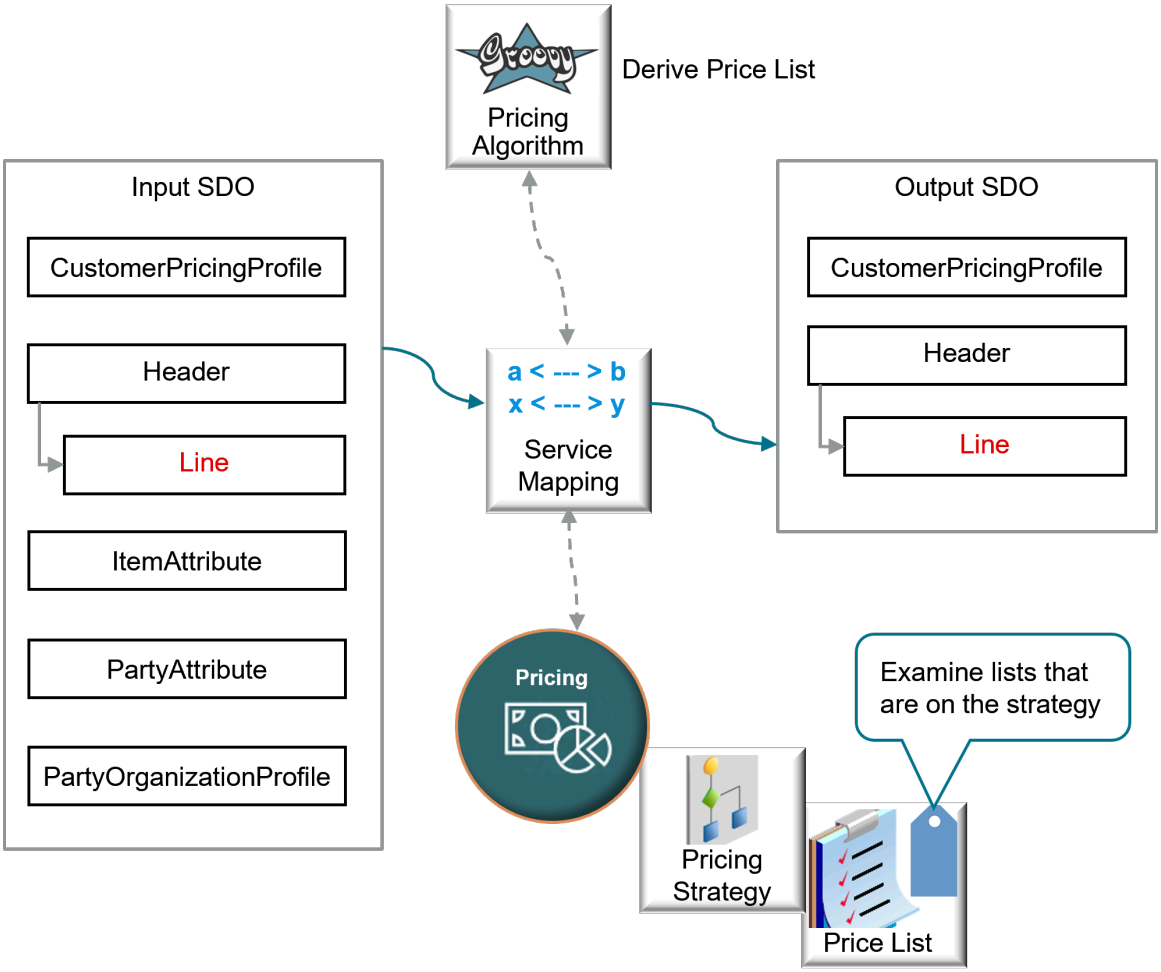
Here are some of the functions you can use with Get Sales Pricing Strategy:

Function Name	Description
GetCustomerPricingProfile	Get values for the customer pricing profile. A typical modification you can do is to get a specific price list for each of your customers. You can use GetCustomerPricingProfile and a descriptive flexfield to implement this kind of modification on the line.
GetPricingSegmentMatrix	Get the type of pricing segment matrix.
getStrategy	Get the pricing strategy.
getStrategyAssignmentMatrix	Get the pricing strategy assignments. A typical modification is to set up a precedence on the assignment matrix, and then run matrix rules according to precedence.

Example Pricing Algorithm, Part 3, Get Price List

Determine which price list to use.





Note

- The output SDO from the Get Pricing Strategy step provides the input SDO.
- Use the Derive Price List algorithm:
  - Get the default price list.
  - If term overrides exist, then apply them.
  - If line overrides exist, then validate them.

Here's what the algorithm does:

- Examine strategies and price lists in the Pricing Administration work area.

- Use the PricingStrategyId in the input SDO, such as 300100071623888, to determine which strategy to examine.
- Examine the price lists on the strategy sequentially according to the Precedence attribute of the price lists on the strategy.
- Use item, unit of measure, and charges in the input SDO to identify price lists that include the same item, unit of measure, and charges on the strategy.
- If it finds a match, then it uses the price list as the default price list and to apply prices for the line. It uses the DefaultPriceListId attribute to identify the price list.
- If a sales agreement applies to the line, and if the agreement overrides the line, then the algorithm uses the agreement price to price the line.
- If a price list override exists on a line, and if the algorithm finds that the price list is valid according to the strategy and date, then it uses the override to price the line.
- If the algorithm doesn't find a matching price list on the strategy, or if the agreement price isn't valid, or if the override price isn't valid, then it creates an error.
- Order Management doesn't come predefined to support overriding the price list. You must manually set it up.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
Line	AllowCurrencyOverrideFlag: Y AppliedCurrencyCode: USD DefaultCurrencyCode: USD HeaderId: 1 InventoryItemId: 149 InventoryOrganizationId: 204 ItemType: Standard LineCategoryCode: Order LineId: 1 LineQuantity2: Ea LineQuantityUOMCode: Ea LineTypeCode: ORA_BUY PriceAsOf: 2019-03-26 @ 08:32:53 PricedOn: 2019-03-26 @ 08:32:53 PricingDate: 2019-03-26 @ 08:32:53 PricingStrategyId: 300100071623888 ShipToLocationId: 929 UnreferencedReturnLineFlag: N	AllowCurrencyOverrideFlag: Y AppliedCurrencyCode: USD **AppliedPriceListId: 300100071623855** In this example, the algorithm didn't find an override, so the applied price list and the default price list are the same. If it found an override, it would set the applied price list to the override price list. DefaultCurrencyCode: USD **DefaultPriceListId: 300100071623855** **DefaultPriceListPrecedence: 1** **FromCurrencyCode: USD**. This is the currency of the price list. **HasAlternatePriceList: Y** HeaderId: 1 InventoryItemId: 149 InventoryOrganizationId: 204 ItemType: Standard LineCategoryCode: Order LineId: 1 LineQuantity2: Ea LineQuantityUOMCode: Ea

Entity	Attributes in Input SDO	Attributes in Output SDO
		LineTypeCode: ORA_BUY PriceAsOf: 2019-03-26 @ 08:32:53 PricedOn: 2019-03-26 @ 08:32:53 PricingDate: 2019-03-26 @ 08:32:53 PricingStrategyId: 300100071623888 ShipToLocationId: 929 UnreferencedReturnLineFlag: N

View Object Lookups in a Hierarchy

Here are some of the functions you can use with the Derive Price List algorithm:

Function Name	Description
getCharges	Get charges for an item or subscription item.
getCoverageCharges	Get charges for a coverage item and covered item.
getItems	Get items for a configuration model.

These functions are hierarchical view object lookups. A view object lookup queries only one lookup. A hierarchical view object lookup queries two objects in the hierarchy. For example, the getCharges function uses view object x to query the price lists in the strategy according to precedence, and view object y to query the price list charges on each price list that view object x returns.

View Object Lookups

Here are some functions that are view object lookups that you can use with Derive Price List:

Function Name	Description
getOverridePriceListName	Get the price list name that OverridePriceListId identifies. If you use a descriptive flexfield to set a specific price for each party that you set up in Trading Community Architecture, then you can use getOverridePriceListName to get the name of the price list to apply instead of using the default value from the strategy.
getStrategy	Get the pricing strategy according to Header.StrategyId.
validateOverrideCurrency	Validate the override currency on the header or the line.
validateOvrPriceList	Validate the override price list on the

Function Name	Description
	line.
validateTermPriceList	Validate the override price list from the pricing term that the sales agreement specifies.

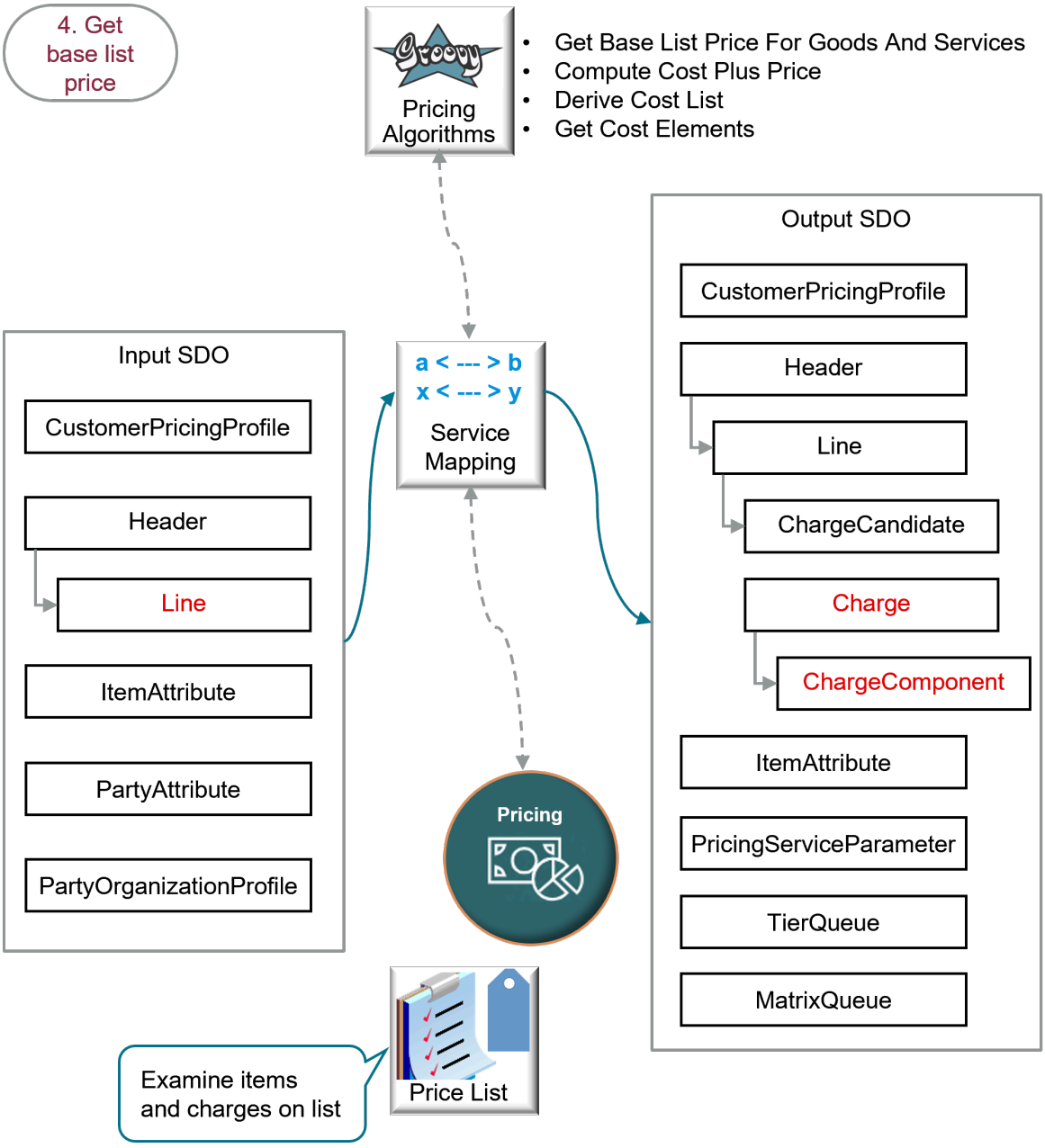
Related Topics

- Pricing Rules

Example Pricing Algorithm, Part 4, Get Base List Price

Create a charge for the item on the price list, then use the charge as the base list price for the item.

The base list price is the price you charge for a quantity of 1 for the item before you apply any tiered pricing that you set up for the item:



The output SDO from the Get Pricing Strategy provides the input SDO.

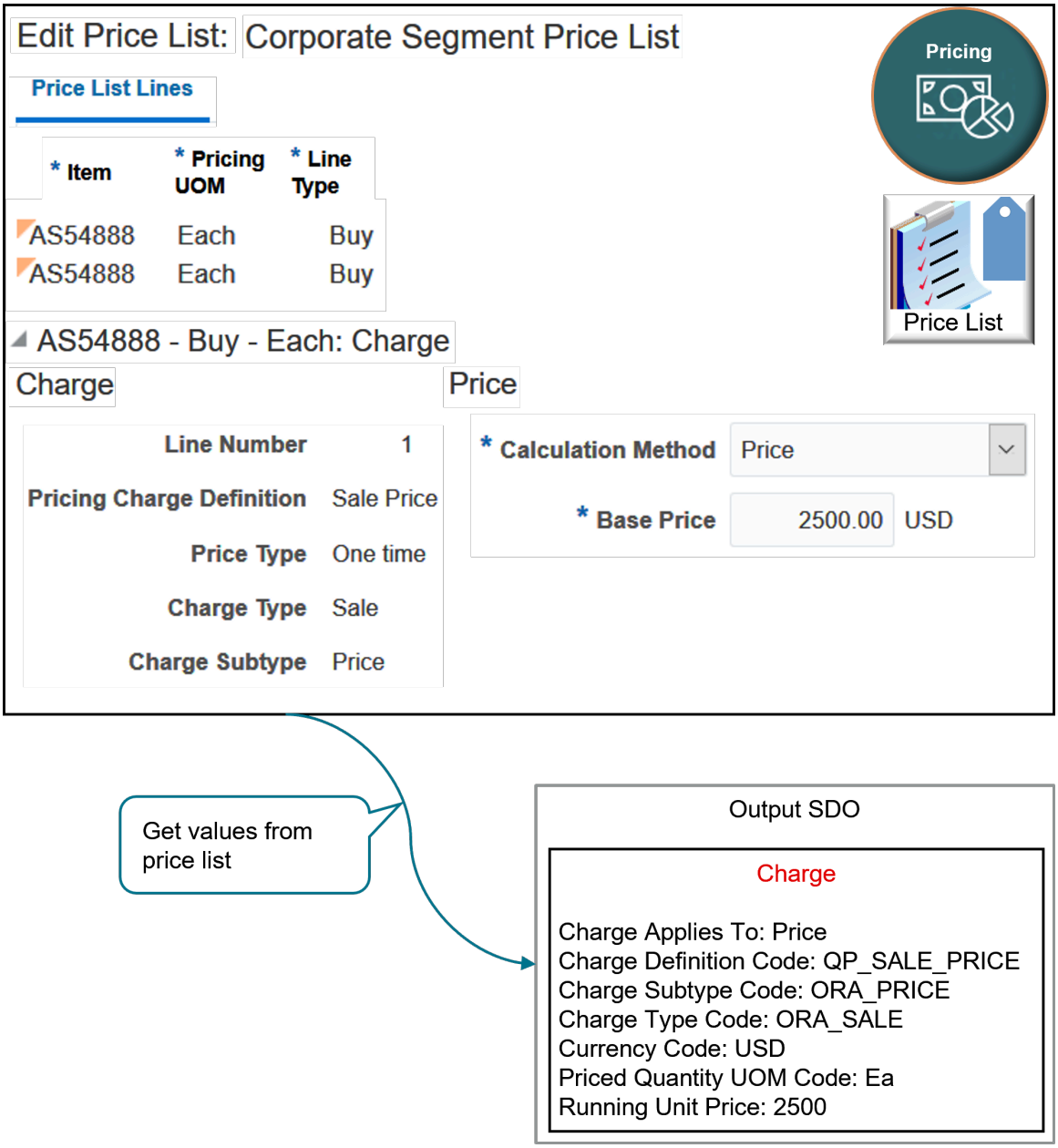
You can use these algorithms to meet your specific needs. Here are some examples:

- Override the base list price according to a value that the Order Entry Specialist enters into an extensible flexfield on the order line.
- Match price list charges for a unit of measure that doesn't match the unit of measure on the order line. Convert the unit of measure on the order line to the unit of measure you need to use.
- Make sure the quantity on the order line is equal to or greater than a minimum quantity that you set for the item.

Here's what the algorithms do:

What the Algorithms Do	How the Algorithms Do It
1. Find charge candidates for coverage items.	-
2. Find charge candidates for component items.	-
3. Find charge candidates for items and component items.	Use values in the input SDO to find charge candidates. For each item, match the item, unit of measure, business unit, and line type. Assume the Corporate Pricing Strategy includes the Corporate Segment Price List. 300100071623855 identifies the list. This price list includes the AS54888 item with a Pricing UOM of Each and Line Type of Buy. Create a charge candidate for each charge that it finds that matches the query. On the Price List page in Pricing Administration, assume you search the Corporate Segment Price List for the AS54888 item. The search returns row 1 and row 2, and each row contains the AS54888 with a Pricing UOM of Each and Line Type of Buy. Each of these rows is a candidate to become the charge.
4. Filter duplicate charge candidates.	If the query finds more than one charge candidate for each line, then filter them according to the item's precedence. An item that has a charge defined takes precedence over other items. For example, if row 1 from your search on the Price List page contains a charge in the Charge area but row 2 doesn't, then the filter will return only row 1.
5. Create charges.	-
6. Calculate cost-plus pricing.	-
7. Create charge component for the base list price.	Use the base price from you set up to set the unit price and calculate the extended amount.

For the 5. Create charges step, create a charge for each candidate that remains after filtering them. For example, if the AS54888 on row 1 and the AS54888 on row 2 each contain a charge, then create a charge for each of them. If no charge candidates match the query criteria, then the algorithm creates an error:



The algorithm uses your set ups in the Pricing Administration work area. If the charge includes:

- Tiered adjustment, then the algorithm populates the TierQueue entity in the output SDO.
- Price adjustment matrix, then the algorithm populates the MatrixQueue entity.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
Line	AllowCurrencyOverrideFlag: Y AppliedCurrencyCode: USD	-

Entity	Attributes in Input SDO	Attributes in Output SDO
	AppliedPriceListId: 300100071623855 DefaultCurrencyCode: USD DefaultPriceListId: 300100071623855 DefaultPriceListPrecedence: 1 FromCurrencyCode: USD. HasAlternatePriceList: Y HeaderId: 1 InventoryItemId: 149 InventoryOrganizationId: 204 ItemType: Standard LineCategoryCode: Order LineId: 1 LineQuantity2: Ea LineQuantityUOMCode: Ea LineTypeCode: ORA_BUY PriceAsOf: 2019-03-26 @ 08:32:53 PricedOn: 2019-03-26 @ 08:32:53 PricingDate: 2019-03-26 @ 08:32:53 PricingStrategyId: 300100071623888 ShipToLocationId: 929 UnreferencedReturnLineFlag: N	
Charge	-	CanAdjustFlag: Y ChargeAppliesTo: Price **ChargeDefinitionCode: QP_SALE_PRICE** ChargeDefinitionId: 3001000070841552 ChargeId: 1 ChargeSubtypeCode: ORA_PRICE ChargeTypeCode: ORA_SALE CompSeqCntr: 1001 CurrencyCode: USD EstimatedPricedQuantityFlag: N

Entity	Attributes in Input SDO	Attributes in Output SDO
		EstimatedUnitPriceFlag: N LineId: 1 NeedsCostPlus: N NeedsMargin: Y ParentEntityCode: LINE ParentEntityId: 1 PriceTypeCode: ONE_TIME PricedQuantity: 2 Ea PricedQuantityUOMCode: Ea PrimaryFlag: N RollupFlag: N **RunningUnitPrice: 2500** TaxIncludedFlag: N
ChargeComponent	-	ChargeComponentId: 1 ChargeId: 1 CurrencyCode: USD **ExtendedAmount: 5000 USD** **PriceElementCode: QP_BASE_LIST_PRICE** PriceValidFrom: 2019-03-26 @ 08:32:53 SequenceNumber: 1000 SourceId: 3001000071623860 SourceTypeCode: PRICE_LIST_CHARGE **UnitPrice: 2500 USD** In this example, the base price is 2,500, the quantity is 2, so the algorithm calculates the extended amount as 5,000.

Here are some of the functions you can use with the algorithms you use to get the base list price:

Algorithm Name	Function Name	Function Type	Description
Get Base List Price For Goods And Services	constructChargeCandidate	Script	Create a data object for the charge candidate according to the view row of the charge definition.

Algorithm Name	Function Name	Function Type	Description
-	getChargeDefinitionName	View object Lookup	Get a single row for the translated name of the charge definition.
-	getCharges	View object lookup	Get price list charges in your pricing setup that match the single price list.
-	getDurationConversionRate	Script	Get the UOM conversion rate for part of the duration.
-	getOKCUomMappings	View object lookup	Get mappings in the contracts okc_ tables to call the contracts API. For details about okc_ tables, go to *Tables and Views for CX Sales and B2B Service*, then see the Contracts chapter.
-	getPriceList	View object lookup	Get your price list.
-	getRootCharge	View object lookup	Get the price list charge of a root item.
-	getUomTranslation	View object lookup	Get the unit of measure table for translations that exist for the UOM.
Compute Cost Plus Price	None	-	-
Derive Cost List	getCostCharge	Hierarchical view object lookup	Get the cost elements.
Get Cost Elements	getCosts	View object lookup	Get costs elements from your setup.

Related Topics

- Pricing Rules

Example Pricing Algorithm, Part 5, Calculate List Price

Learn how pricing algorithms calculate list price.

Here are some examples of how you can set up algorithms to calculate list price that meet your specific needs:

- Prorate a tier adjustment according to billing frequency for a recurring charge.

- Apply precedence when you use an attribute to adjust the price list charge.
- Add the name of the discount rule to your explanation message.
- Round the list price according to business unit.

This step includes three parts:

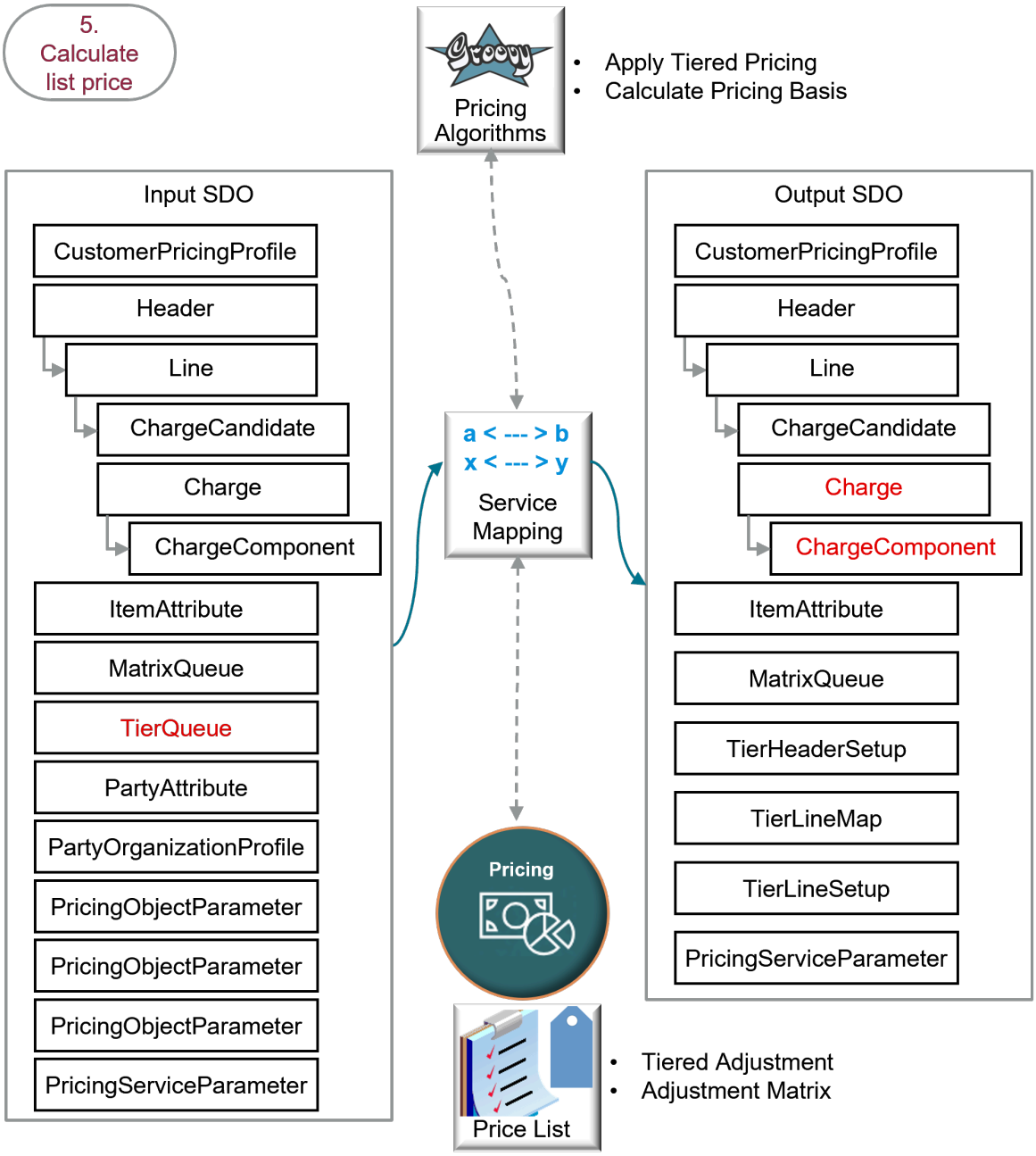
- Apply Tier Adjustment
- Apply Matrix Adjustment
- Finalize List Price

Apply Tier Adjustment

You apply a tier adjustment in our example. For example, prorate a tier adjustment according to the billing frequency of a recurring charge. Use a service mapping to send the billing frequency through an extensible attribute to the algorithm, then use the Apply Tiered Pricing algorithm to extend the duration of the recurring charge.

You apply a tier adjustment that you set up on the price list. Apply it to the base list price. You create tiers according to amount or quantity. For example, create tier 1 that gives a 10% discount off the base list price for quantities 1 through 10, and tier 2 that gives a 20% discount for quantities above 10.

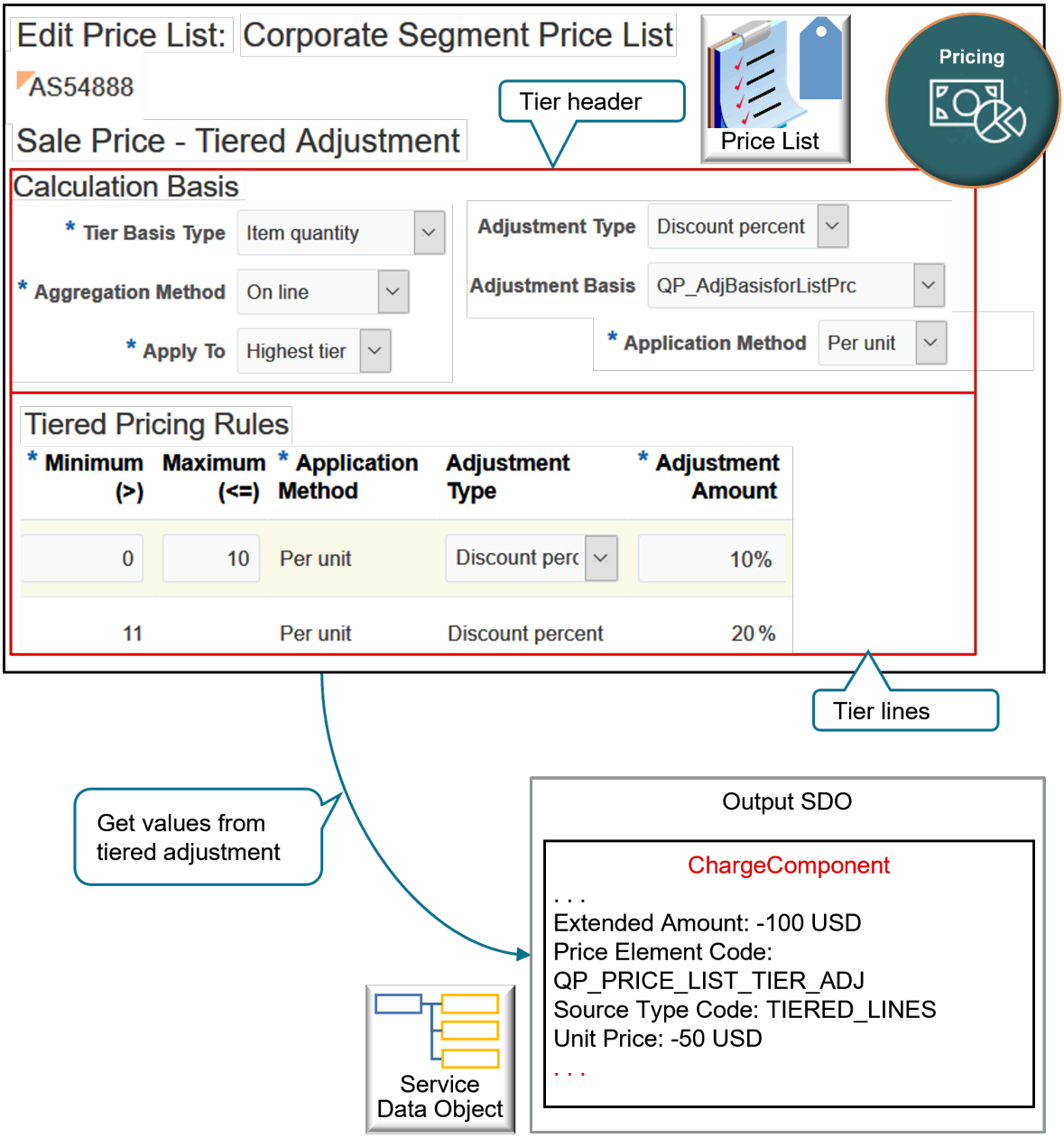
Here are the SDOs:



Note

- The output SDO from the Get Base List Price step provides the input SDO.
- The TierQueue entity contains details about the tier adjustment.

The algorithms use your set ups in the Pricing Administration work area:



Here's what the algorithms do:

What the Algorithms Do	How the Algorithms Do It
1. Get the tier header.	Get details from the tier header. For example, the type of tier basis, such as item quantity. The Calculation Basis area on the Price List page in Pricing Administration contains most of the tier header attributes.
2. Calculate the adjustment amount according to the tier basis.	Assume: • The tier basis is item quantity. • The quantity on the order line is 2.

What the Algorithms Do	How the Algorithms Do It
	• The base list price is 2,500 for Each. • Tier 1 gives a 10% discount on quantities 1 through 10. So, the adjustment amount is 500 (2 multiplied by 2,500 multiplied by 10%).
3. Calculate the accumulation basis.	-
4. Calculate the item quantity according to the quantity tier basis.	-
5. Calculate the item amount according to the amount tier basis.	-
6. Get the tier lines.	-
7. Calculate the adjustment basis for the tier.	-
8. Calculate the tier.	-
9. Calculate the tier adjustment.	-
10. Create the charge component.	-

Here are details about the attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
Charge	-	. . . **RunningUnitPrice: 2450** . . .
ChargeComponent	-	ChargeComponentId: 2 ChargeId: 2 ExplanationMessageName: QP_PRICE_LIST_TIER_ADJ **ExtendedAmount: -100 USD** IsInternal: N **PriceElementCode: QP_PRICE_LIST_TIER_ADJ** PriceValidFrom: 2019-03-26 @ 08:32:53

Entity	Attributes in Input SDO	Attributes in Output SDO
		SequenceNumber: 1001 SourceId: 300100177515265 SourceTypeCode: TIERED_LINES **UnitPrice: -50 USD**

Note

- UnitPrice contains the adjustment amount.
- RunningUnitPrice is BaseListPrice of 2500 minus UnitPrice of 50 equals of 2450.
- ExtendedAmount is UnitPrice of 50 multiplied by Quantity of 2 equals 100.

Here are some of the functions you can use with the Apply Tiered Pricing algorithm. This algorithm does most of the calculations. It calls the Calculate Pricing Basis algorithm. All functions are View Object Lookup except getMessage, which is a script:

Function Name	Description
getDiscountListName	Get the name of the discount list.
getMessage	Get an error message.
getPriceElement	Get the pricing element according to the price element code.
getPriceListCharge	Get the price list charge according to PriceListChargeId.
getPriceListName	Get the price list name, such as Corporate Segment Price List.
getPriceListNameFromCharge	Get the price list name that's associated with the charge.
getPricingBasisHeader	Get the header of the pricing basis.
getTierHeader	Get attribute values for one tier header. Go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Pricing Tiers**, then click **Get one tier header**.
getTierLine	Get attribute values for one tier line. Go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management > Pricing Tiers > Tier Lines**, then click **Get one tier line**.

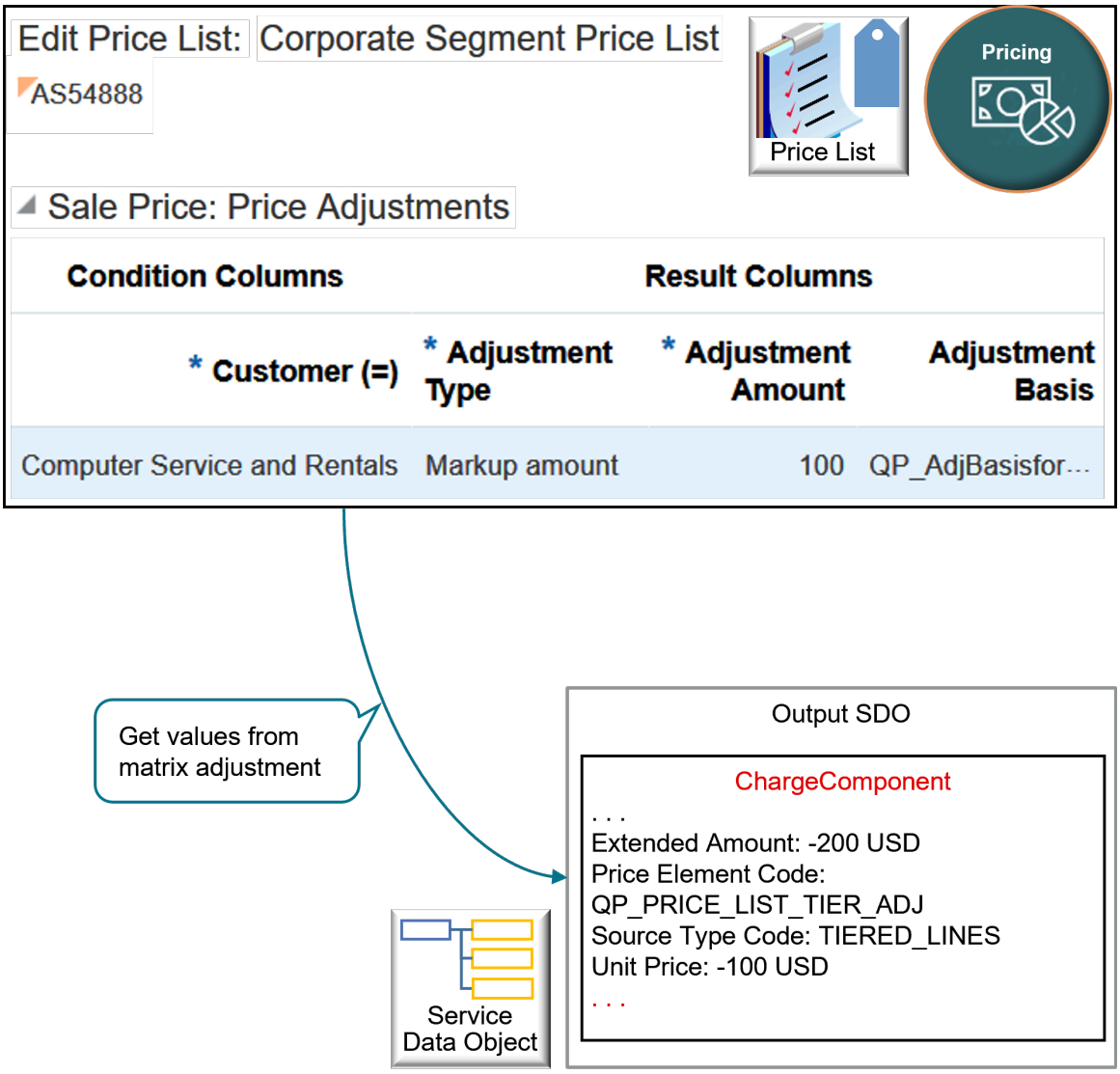
Apply Matrix Adjustment

The matrix adjustment is similar to the tier adjustment:

- Apply a matrix adjustment that you set up on the price list. Apply it on the base list price.

  Set up a matrix adjustment according to the value of an attribute. For example, if the Customer attribute contains Computer Service and Rentals, then give a $100 discount.
- The output SDO from the Apply Tier Adjustment part of the Calculate List Price step is the input SDO in the Apply Matrix Adjustment part.
- The MatrixQueue entity in the input SDO contains the matrix adjustment details.

The algorithm uses your set up in the Pricing Administration work area:



Here's what the algorithm does:

What the Algorithm Does	How the Algorithm Does It
1. Get the attributes.	-
2. Evaluate the matrix rules.	If your add more than one rule in your matrix, then add a precedence condition so the algorithm knows which rule to apply if more than one rule matches the runtime condition.
3. Calculate the matrix adjustments.	-
4. Handle matrix errors. The most common error happens when the condition column isn't part of the algorithm data set.	-

Here are details about the attributes in the SDOs. Bold font indicates attributes that the algorithm added to the output SDO:

Entity	Attributes in Input SDO	Attributes in Output SDO
Charge	-	. . . **RunningUnitPrice: 2350** . . .
ChargeComponent	-	ChargeComponentId: 3 ChargeId: 1 ExplanationMessageName: QP_PRICE_LIST_ATTR_ADJ **ExtendedAmount: -200 USD** **MatrixConditionString: Customer=Computer Service and Rentals** **MatrixResultString: Adjustment Type=Discount Amount Adjustment Amount =100 Adjustment Basis=null** **PriceElementCode: QP_PRICE_LIST_ATTR_ADJ** SequenceNumber: 1002 SourceId: 300100177515271 **SourceTypeCode: MATRIX_RULE** **UnitPrice: -100 USD**

Note

- UnitPrice contains the adjustment amount.
- Assume you create a tier adjustment and a matrix adjustment. The flow applies the tier adjustment first, then the matrix adjustment. The value of RunningUnitPrice after the tier adjustment is 2450. So, RunningUnitPrice now is 2450 minus UnitPrice of 100 equals 2350.
- ExtendedAmount is UnitPrice of 100 multiplied by Quantity of 2 equals 200.

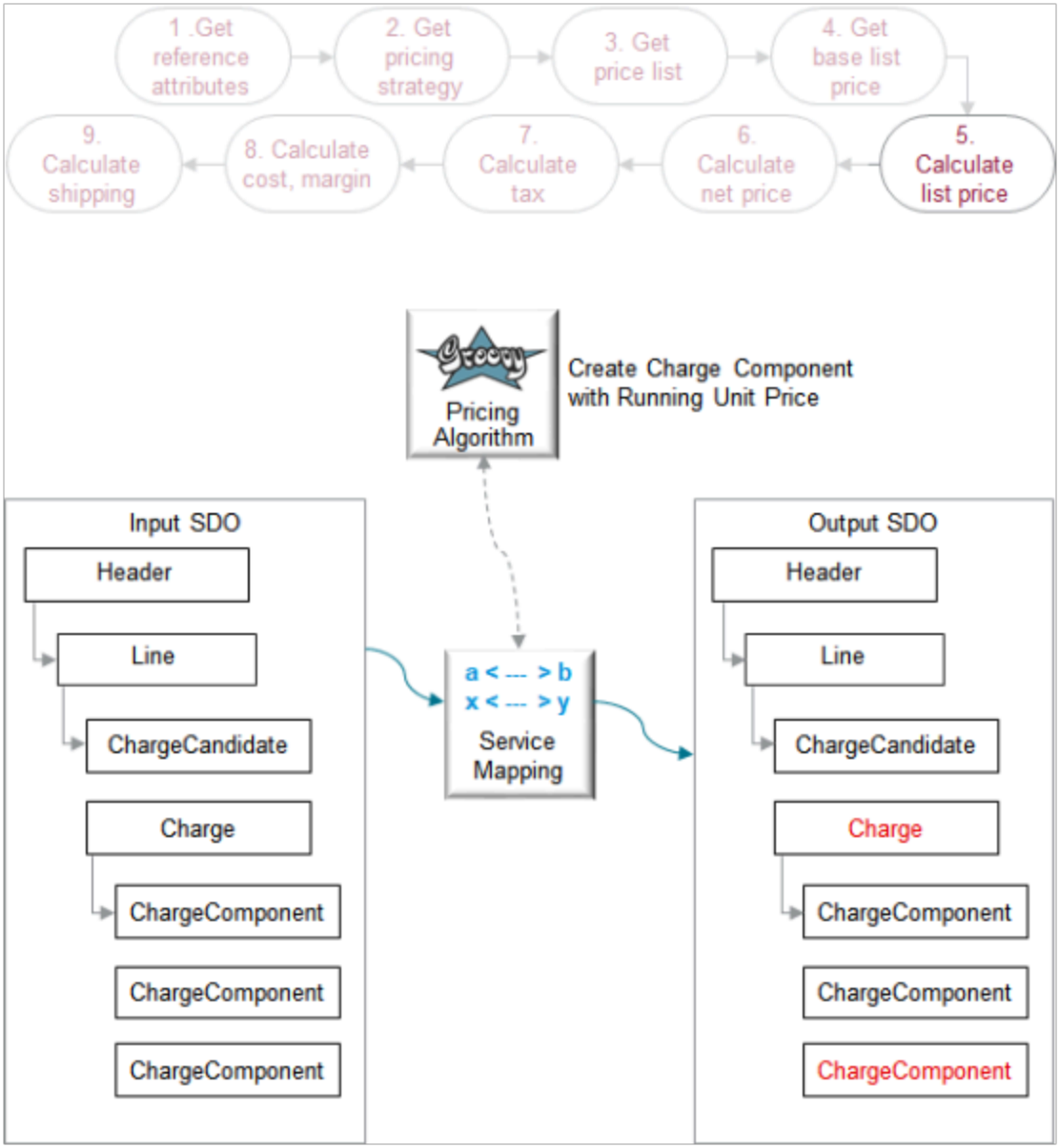
Here are some of the functions you can use with the Apply Matrices algorithm:

Function Name	Description
getAdjustmentBasis	Get the name of the adjustment basis.
getChargeDefinitionName	Get the name of the charge definition.
getDiscountListName	Get the name of the discount list.
getItemExtAttrDimensions	Get the dimensions of the item extensible attributes. Get attribute values for one tier header.
getPriceListName	Get the name of the price list.
getPriceListNameFromCharge	Get the name of the price list that's associated with the charge.

For details, go to *REST API for Oracle Supply Chain Management Cloud*, expand **Order Management**, then click **Pricing Matrixes**.

You can provide an explanation as part of the charge component, then display it on the sales order in Order Management. For example, include the name of the discount rule in the explanation.

Finalize List Price



The output SDO from the Apply Matrix Adjustment part of the Calculate List Price step provides the input SDO.

Here's what the algorithm does:

What the Algorithms Do	How the Algorithms Do It
1. Round the list price and the net price.	Examine, then round RunningUnitPrice for each charge: • Get the rounding rules for the currency that the charge uses. In this example, the charge currency is the same as the price list currency, which is USD.

What the Algorithms Do	How the Algorithms Do It
	• Determine the delta. The delta is the difference between the value that isn't rounded and the value that's rounded. • Apply the delta on another charge component as a rounding adjustment. For example, if RunningUnitPrice is $101, and if the round rule specifies to round to the nearest hundred, then the delta is $1. • Set RunningUnitPrice to the rounded value. • Create a charge component then store the rounded value in it. • Calculate the extended amount according to the quantity.
2. Create the charge component for the net price.	-

Here are details about the attributes in the SDOs. Bold font indicates attributes that the algorithm added to the output SDO:

Entity	Attributes in Input SDO	Attributes in Output SDO
Charge	-	. . . **RunningUnitPrice: 2350** . . .
ChargeComponent	-	ChargeComponentId: 4 ChargeId: 1 CurrencyCode: USD **ExtendedAmount: 4700 USD** **PriceElementCode: QP_LIST_PRICE** **PriceElementUsageCode: LIST_PRICE** SequenceNumber: 1003 SkipRunningPrice: Y TaxIncludedFlag: N **UnitPrice: 2350 USD**

Note

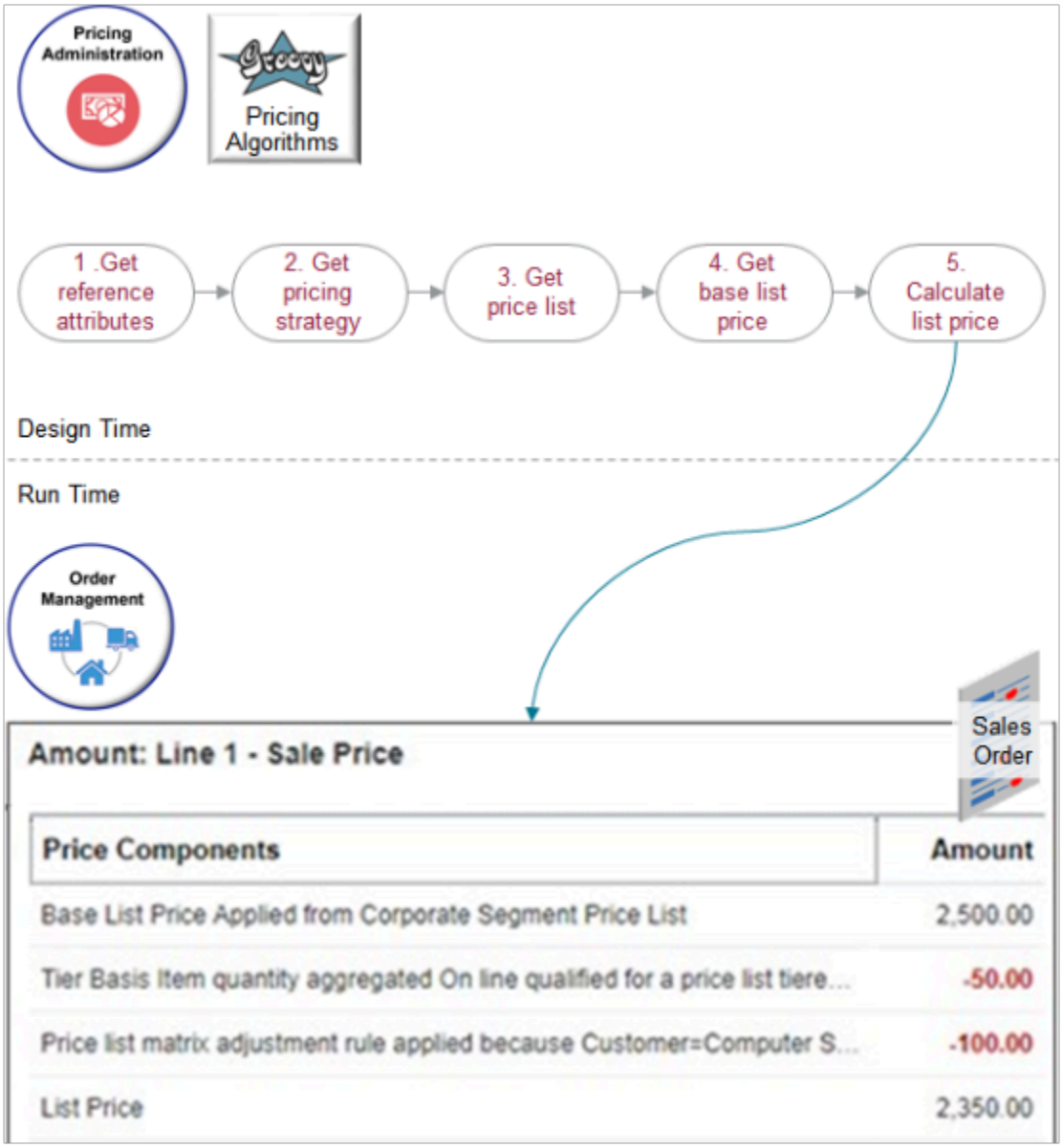
- UnitPrice contains the rounded, final amount.
- RunningUnitPrice is the rounded value.
- RunningUnitPrice contains the same value that the charge component for the list price contains.
- ExtendedAmount is RunningUnitPrice of 2350 multiplied by Quantity of 2 equals 4700.
- PriceElementCode uses QP_LIST_PRICE to set the list price.

Here are some of the functions you can use with the Create Charge Component with Running Unit Price algorithm. You can use it to modify how to round the price. All functions are View Object Lookup except getDurationConversionRate, which is script:

Function Name	Description
getChargeDefinitionName	Get the name of the charge definition.
getDurationConversionRate	Get the conversion rate for the duration.
getOKCUomMappings	Get the mapping for the time unit of measure.
getUomTranslation	Get the translation of the abbreviation, such as Ea, for the unit of measure.

Where Do Pricing Details Display?

The Order Management work area displays each charge component on a separate line in the Line dialog of the sales order:



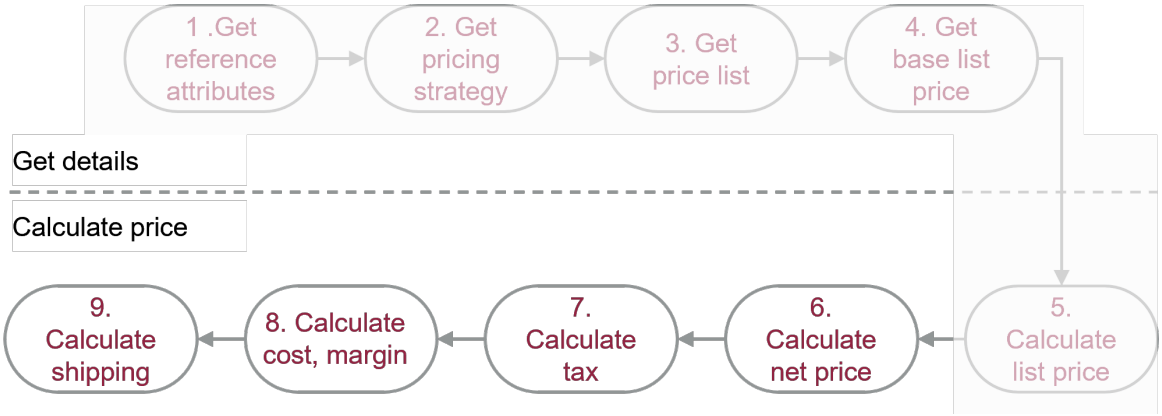
Related Topics

- Pricing Rules

Example Pricing Algorithm, Part 6, Calculate Net Price

Learn how pricing algorithms calculate net price.





Let's see what happens when we calculate net price, tax, cost, margin, and shipping.

We will use a different scenario in the rest of this topic so we can fully illustrate the flow:

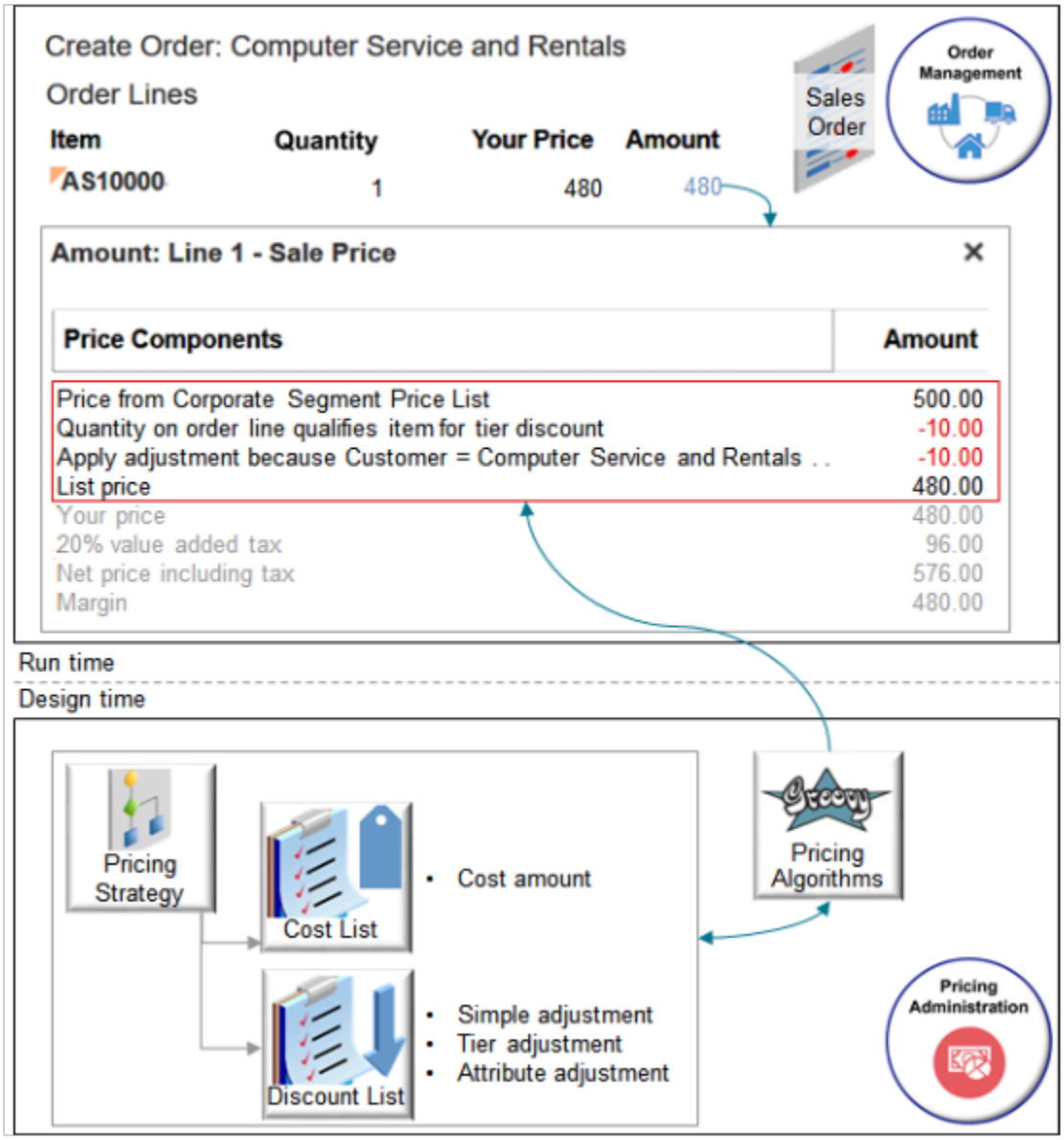
- Calculate pricing for item AS100000
- Provide a $10 discount on the order line for quantities 0 to 10.
- Provide a $10 discount for customer Computer Service and Rentals.

Here are the main steps to calculate net price:

1. Apply pricing terms from the sales agreement.
2. Apply discounts from the pricing strategy.
3. Apply the manual price adjustment.
4. Get the invoice price.

Its more typical to apply discounts from the pricing strategy rather than to apply terms from the sales agreement, so this example starts with the strategy.

Consider an example. You add the AS10000 to an order with a quantity of 1:



Note

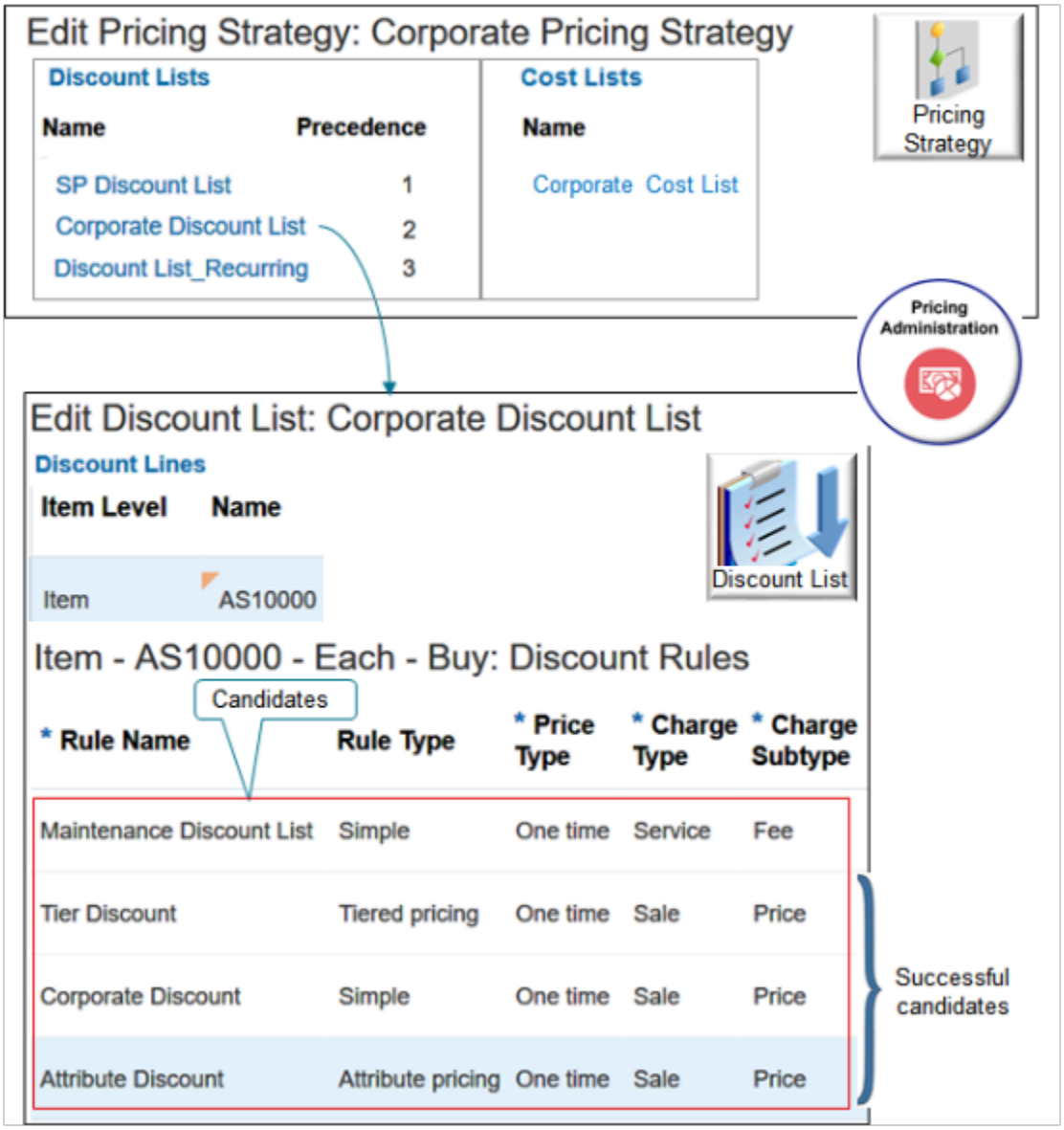
- Pricing algorithms get the base list price for the item from the Corporate Segment Price List.
- Algorithms get simple, tier, and attribute adjustments from the discount list.
- The pricing strategy references the price list and the discount list.
- Algorithms examine the strategy, price lists, and discount lists to calculate pricing.
- Your design time setup in the Pricing Administration work area determines run time pricing on the sales order in the Order Management work area.

Apply Discounts from Pricing Strategy

To calculate net price, you first apply discounts from the pricing strategy. Here are some typical setups you can do to meet your specific needs:

- Apply a discount according to customer, order type, or customer class. Create a matrix that includes a condition according to these and other attributes on the order header or the order line. Use an attribute that describes item, such as Weight, or that describes the customer, such as Business Unit.
- Cascade discounts. Pricing comes predefined to apply simple rules, then tier rules, then matrix rules. But pricing doesn't adjust the calculation after each discount. Assume you start with a value of $480, apply a simple $50 discount, which results in $430. You then apply a 10% tier discount. Pricing applies the 10% on $480, not $430. To cascade discounts so pricing applies 10% on $430, you can create your own element that stores the running unit price.
- Modify the explanation message you display with the discount. For example, include the name of the term rule that you use in a discount matrix adjustment.
- Include the adjustment group.
- Remove a dimension from the adjustment basis.

Assume your set up uses the Corporate Pricing Strategy:



Note

- The Corporate Pricing Strategy contains three discount lists.
- The Precedence for Corporate Discount List is 2.
- Corporate Discount List contains rules for the AS10000 item:
  - There are two simple rules, a tier rule, and an attribute rule.
  - Each rule uses a Sale charge type and Price subtype except Maintenance Discount List, which uses Service and Fee.
- None of the other lists contain rules for the AS10000.

- Here's the set up for the tier rule:

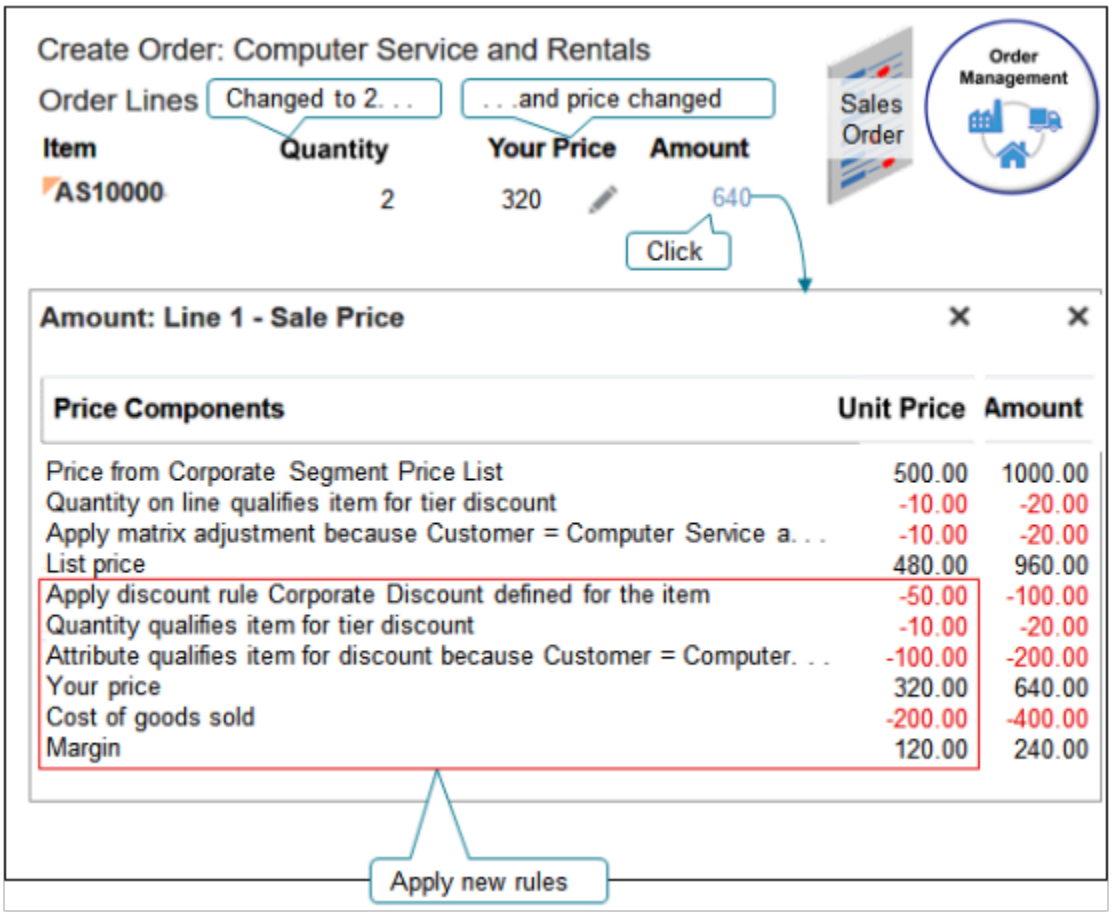
Minimum	Maximum	Application Method	Adjustment Type	Adjustment Amount
0	10	Per Unit	Discount Amount	10.00 USD
10	100	Per Unit	Discount Amount	20.00 USD

- Assume the attribute rule gives a $100 discount when customer Computer Service and Rentals orders the AS10000.

The strategy references the Corporate Cost List. The list contains the AS10000 item, and it uses the Cost Amount attribute to specify the cost for the item:

Attribute	Value
Cost Amount	200

Now, change the quantity from 1 to 2 and see what happens:



Note

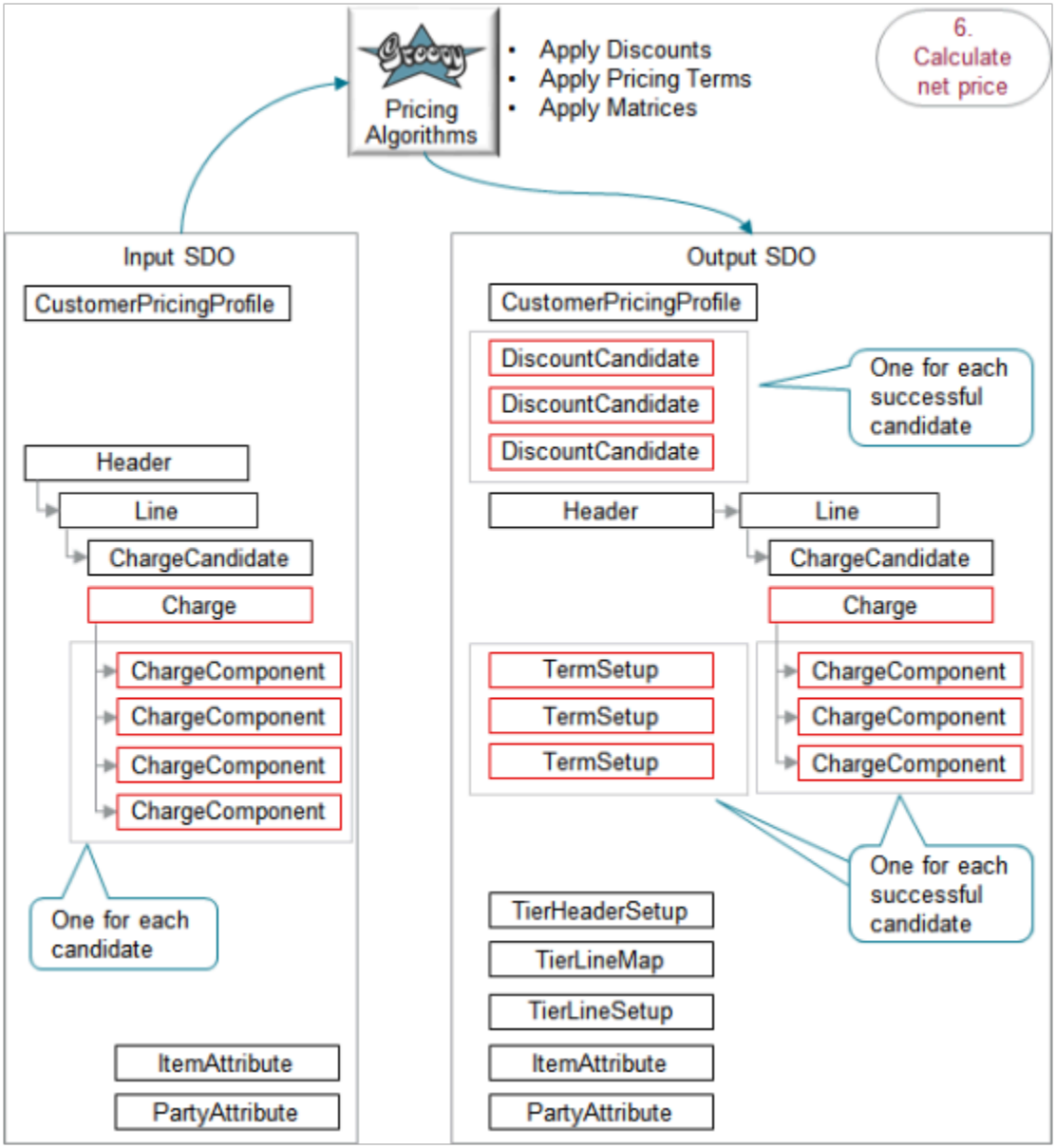
- Your Price equals List Price minus the sum of the three new discounts.

  ```
  320 = 480 - (50 + 10 + 100)
  ```

- The charge breakdown displays the Cost Amount as Cost of Goods Sold.
- Margin equals Your Price minus Cost of Goods Sold.

  ```
  120 = 320 - 200
  ```

Here are the SDOs:



Note

- The Calculate Net Price step provides the input SDO.

- The Line entity in the input SDO represents one order line of a sales order.
- The input SDO includes four ChargeComponent entities, one for each of the discount list candidates.
- The algorithm creates entities for each successful pricing candidate in the output SDO. There are three successful candidates, so the SDO includes three DiscountCandidate entities, three TermSetup entities, and three ChargeComponent entities.
- The service mapping works the same way as described earlier in this topic. The diagram doesn't include it for brevity.

Here's what the algorithms do:

What the Algorithms Do	How the Algorithms Do It
1. Get discount candidates.	Examine the discount rules that are currently active on the pricing strategy. Examine them sequentially according to the value of the Precedence attribute for the list on the strategy. Stop when we find the first list that contains a successful candidate. All active discount rules on the strategy are candidates. In our example, the precedence for Corporate Discount List is 2, so the algorithm examines the rules on SP Discount List first, then Corporate Discount List. Identify the successful discount rule candidates. To be successful, the discount list must: • Contain the same item that the order line contains, such as AS10000. • Contain the same unit of measure that the order line contains, such as Each. • The access set on the discount list must match the business unit on the order header. In our example, SP Discount List has a precedence of 1, so the algorithm examines it first. It doesn't find a candidate, so it examines Corporate Discount List. It finds at least one candidate so it doesn't examine any other lists, such as Discount List Recurring. All four rules in Corporate Discount List meet the criteria, so they're all successful candidates.

2. Qualify charge criteria.	Qualify the rule for each successful candidate.
	
	The input SDO specifies.
	
	`ChargeSubtypeCode: ORA_PRICE`
	`ChargeTypeCode: ORA_SALE`
	
	So, the algorithm filters out the Maintenance Discount List rule because its charge type is Service, not Sale, and its subtype is Fee, not Price.
	
	At this point, only three discount candidates remain.

3. Filter according to item precedence.	If you add a row to you discount list and set Item Level to All Items, and if you also add a specific item to the list, then the algorithm uses the row that contains the specific item and ignores the row that contains All Items. This example doesn't include a row that contains All Items, so this behavior doesn't apply.
4. Get discount candidates for configurable options.	-
5. Convert currency.	If a candidate doesn't use the same currency that the input SDO specifies, then the algorithm converts the currency. For example, if the Tier Discount rule uses RMB but the input SDO specifies USD, then the algorithm converts RMB to USD.

What the Algorithms Do	How the Algorithms Do It
6. Get pricing terms.	The algorithm gets pricing terms from your setup in Pricing Administration, such as on a matrix class that you import. For details, see *Import Batches of Discount Lists That Have Pricing Matrices*. The algorithm also examines terms that you set up in a sales agreement. For details, see *Overview of Setting Up Sales Agreements in Order Management*.
7. Apply pricing terms.	Apply the pricing terms we found. Run the Apply Pricing Terms algorithm. Find charges in the pricing terms that match charges on the order line. Create a separate TermQueue entity for each match we find. In our example, three rules match, so we create a separate term candidate for each of these rules.
8. Get the pricing basis.	Get the pricing basis from the price list that's defined on the strategy. We need the pricing basis so we can apply adjustments to it. Assume the Corporate Pricing Strategy contains the Corporate Segment Price List. This price list contains the AS10000, and the Base Price attribute for the AS10000 on the list contains 480.00.
9. Apply simple adjustments.	Create a chargeComponent entity for each TermQueue that references a simple discount rule. Store the chargeComponent in the Charge entity in the output SDO. In our example, the Corporate Discount rule is the only simple rule that matches the criteria. Assume the rule applies a $50 discount. Calculate the running unit price: 480 minus 50 equals 430.
10. Apply other adjustments.	-
11. Apply tiered adjustments.	Create a chargeComponent entity for each TermQueue that references a tier discount rule. Store the chargeComponent in the Charge entity in the output SDO. In our example, the Tier Discount rule is the only tier rule that matches the criteria. Assume the rule applies a $10 discount. Calculate the running unit price: 430 minus 10 equals 420.
12. Apply matrix adjustments.	Create a chargeComponent entity for each TermQueue that references an attribute discount rule. Store the chargeComponent in the Charge entity in the output SDO. In our example, the Attribute Discount rule is the only attribute rule that matches the criteria. Assume the rule applies a $100 discount. Calculate the running unit price: 420 minus 100 equals 320.
13. Control manual adjustments.	If the pricing term doesn't allow the flow to apply a matrix adjustment, then the algorithm sets the Can Adjust attribute on the charge to N to prevent you from applying a manual adjustment in the Order Management work area.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
Charge	ChargeSubtypeCode: ORA_PRICE ChargeTypeCode: ORA_SALE PriceTypeCode: ONE_TIME RunningUnitPrice: 480.00	RunningUnitPrice: 320.00
ChargeComponent	-	ChargeComponent for the simple Corporate Discount rule. **ChargeComponentId: 5** **ChargeId: 1** **CurrencyCode: USD** **ExplanationMessageName: QP_DISCOUNT_ADJ** **ExtendedAmount: -100 USD** **PriceElementCode: QP_DISCOUNT_ADJ** **PriceElementUsageCode: PRICE_ADJUSTMENT** **PriceValidFrom: 2019-06-03 @ 21:58:27** **SequenceNumber: 1004** **SourceId: 300100177810852** **SourceTypeCode: PRICING_TERM** **UnitPrice: -50 USD** ExtendedAmount equals UnitPrice 50 multiplied by Quantity 2 equals $100. Details for the other charge components contain similar data.

Note

- The list price provides a value of 480 for the RunningUnitPrice in the input SDO.
- The RunningUnitPrice is 480 minus discounts from the three rules equals 320.00.

Here are some of the functions you can use with the Apply Discounts algorithm:

Function Name	Description
getDiscountRules	Get the discount rules that you set up on the strategy for the item.
getPricingTerm	Get the pricing term.
getComponentDiscountRules	Get the discount rules that you set up for configurable options in a configured item.

Here are some of the functions you can use with the Apply Pricing Terms algorithm:

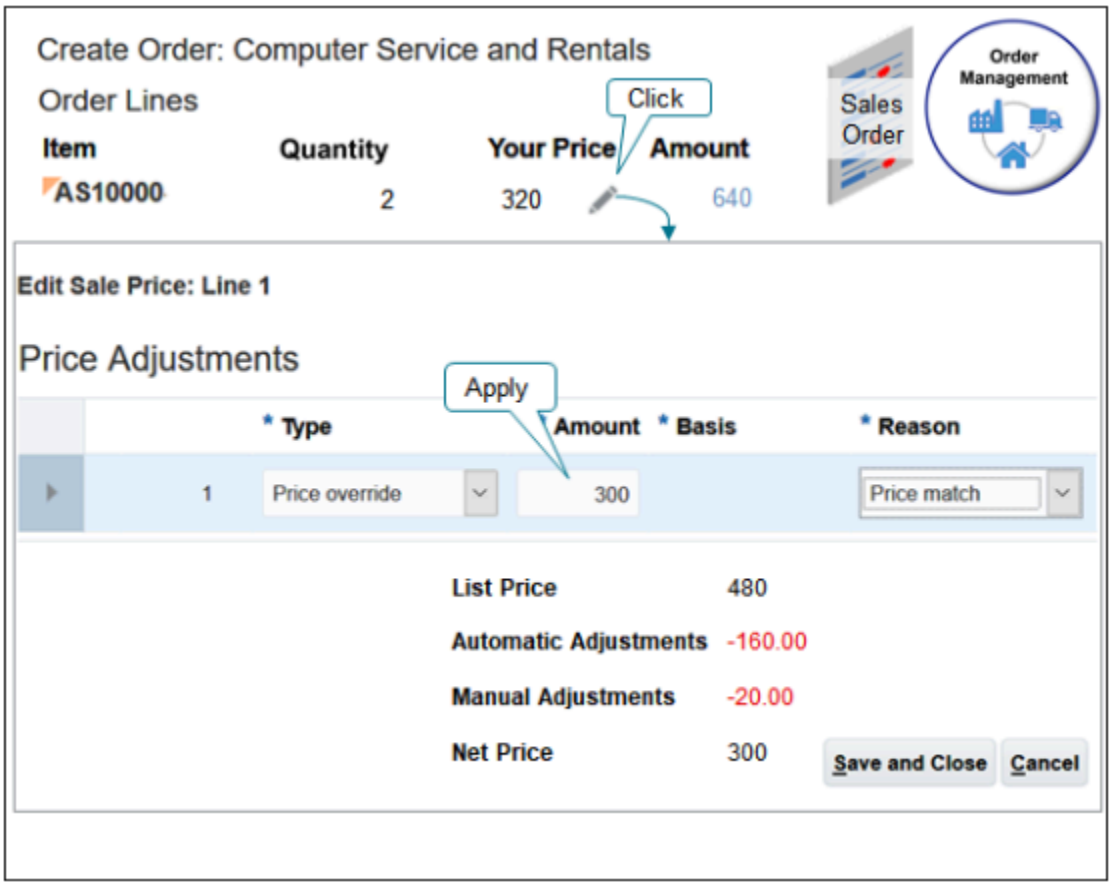
Function Name	Description
getAdjustmentBasisName	Get the translated name of the adjustment basis.
getChargeDefinitionName	Get the translated name of the charge definition.
getContractHeader	Get details from the contract header.
getContractLine	Get details from the contract line.

This step also uses the Apply Matrices and Apply Tiered Pricing algorithms. See the first part of this topic for a description.

Apply Manual Price Adjustments

What happens when you apply a manual price adjustment?

Assume your competitor offers a similar item, so you apply a manual price adjustment to match that price:

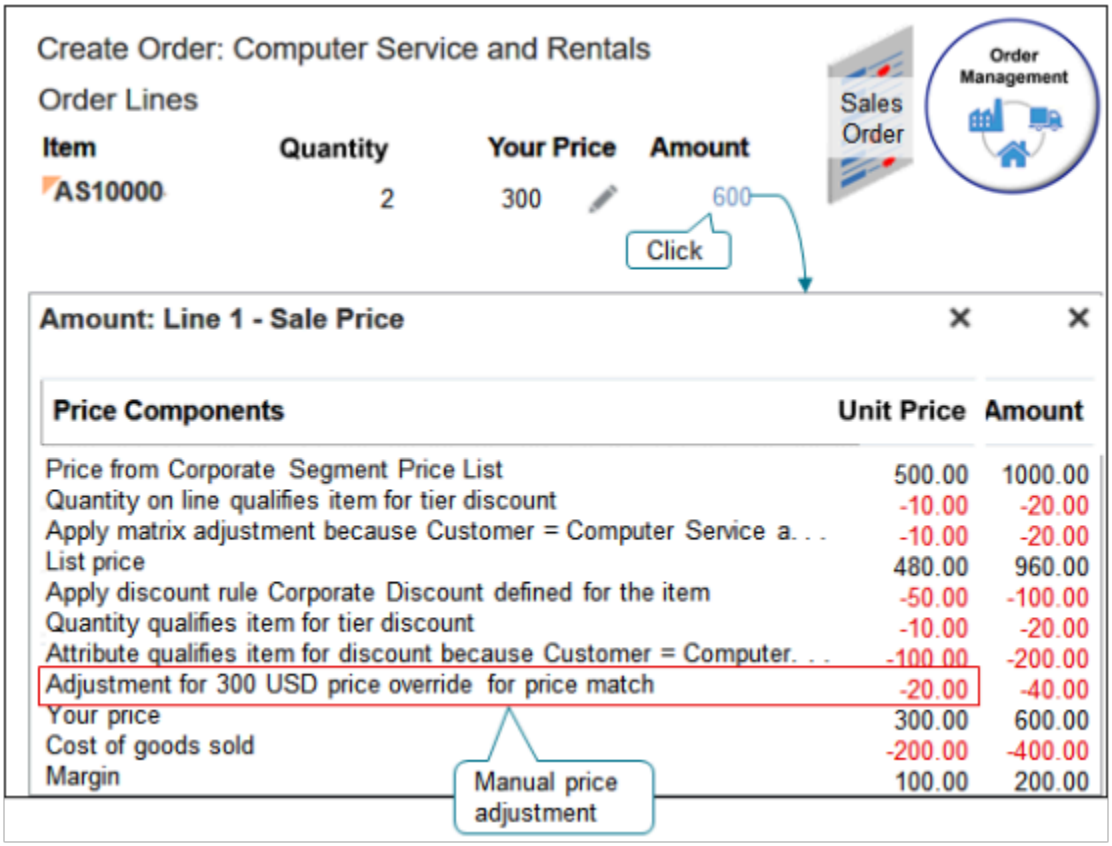


Note

- You click the value in the Amount column.

- You use the Price Adjustments dialog to specify the manual price adjustment.
- You apply a $300 price override.

Examine the breakdown:



Note

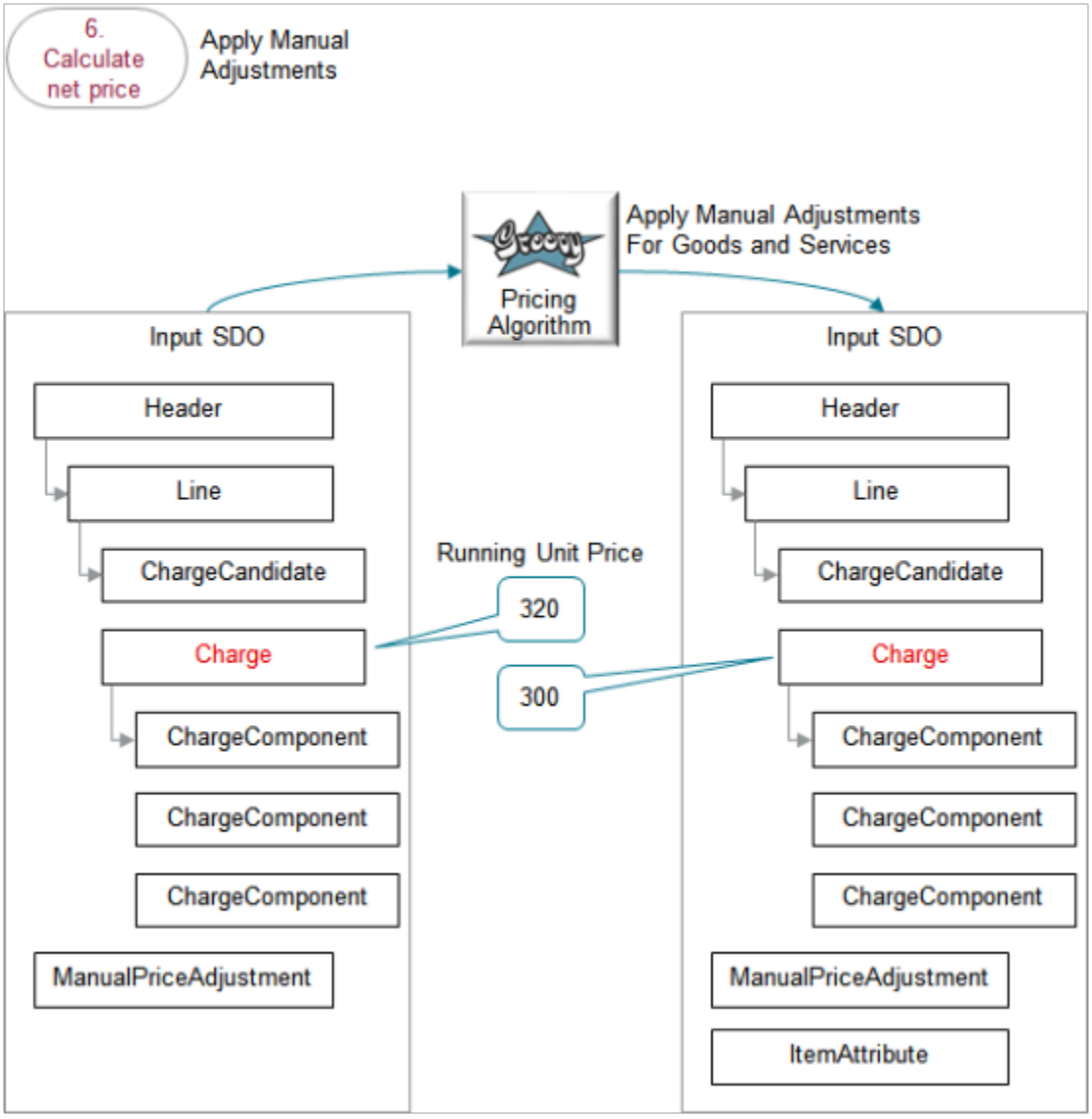
- The override is 300, Your Price before the adjustment was 320, so the algorithm does a minus 20 adjustment.
- Your Price equals List Price minus the sum of the four discounts.

```
300 = 480 - (50 + 10 + 100 + 20)
```

- Margin equals Your Price minus Cost of Goods Sold.

```
100 = 300 - 200
```

Here are the SDOs:



Note

- The output SDO from the Apply Discounts from Pricing Strategy step provides the input SDO.
- The ManualPriceAdjustment entity contains the values you set in the Price Adjustments dialog when you modify the sales order in the Order Management work area.
- Use the Apply Manual Adjustments For Goods and Services algorithm.
- The running unit price for the charge on the input SDO after applying discounts from the pricing strategy but before applying the manual adjustment is $320.
- The running unit price for the charge on the output SDO after applying the manual adjustment is $300.
- The algorithm uses the ManualPriceAdjustment entity to apply the sales price charge on the line.
- The algorithm uses your set ups in the Pricing Administration work area.

Here's what the algorithm does:

What the Algorithm Does	How the Algorithm Does It
Validate manual price adjustments.	Makes sure: • There's a matching charge. • Pricing can manually adjust the charge. • Pricing supports the manual adjustment type. • Pricing can determine the rate for the currency conversion. • The currency type supports a manual price adjustment according to amount.
Convert currency.	If you must do a currency conversion, then determine the conversion rate.
Calculate manual adjustments for rolled up charges.	-
Calculate manual adjustments for native charges.	-

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
ManualPriceAdjustment	**AdjustmentCurrencyCode: USD** **AdjustmentTypeCode: PRICE_OVERRIdE** **AdjustmentValue: 300** **ChargeDefinitionId: 3001000070841552** **ChargeParentEntityCode: LINE** **ChargeParentEntityId: 1** **ChargeRollupFlag: N** Adjustment Type Code identifies the value you select for the Type attribute in the Price Adjustments dialog. Adjustment Value identifies the value that you enter in the Amount attribute in the Price Adjustments dialog.	-
ChargeComponent	-	**ChargeComponentId: 8** **ChargeId: 1** **CurrencyCode: USD** **ExtendedAmount: -40 USD** **PriceElementCode:** **QP_CUSTOM_ADJUSTMENT** **PriceElementUsageCode:** **PRICE_ADJUSTMENT** **PriceValidFrom: 2019-06-03 @ 21:58:27** **SequenceNumber: 1007** **SourceId: 0** **SourceTypeCode: MANUAL_ADJUSTMENT** **UnitPrice: -20 USD** Unit Price is the amount of adjustment needed to bring the net price to the value that you enter

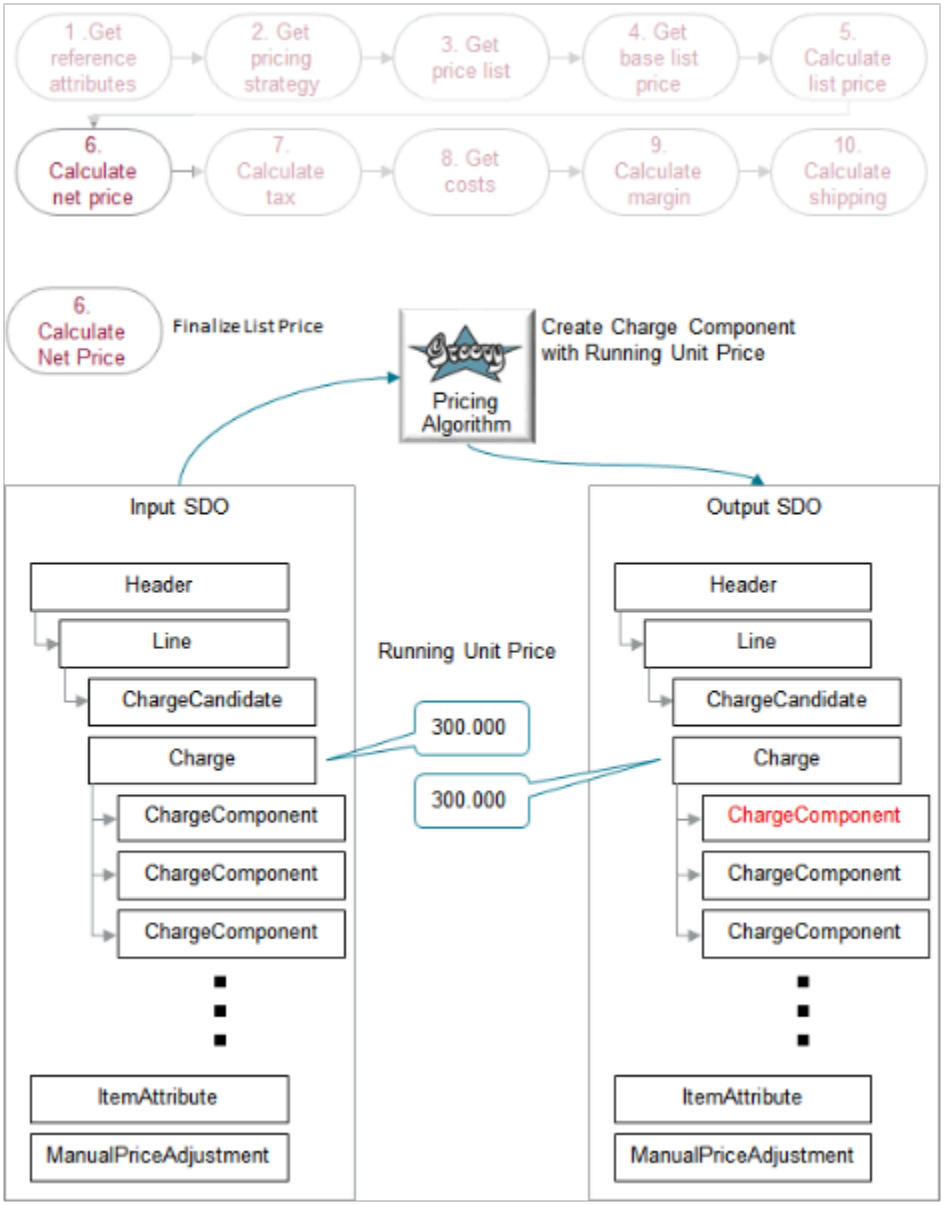
Entity	Attributes in Input SDO	Attributes in Output SDO
		for the Adjustment Value after Pricing subtracts the automatic adjustments. In this example, automatic adjustments are the sum of the discounts that the algorithm calculated. Recall the equation we used before you applied the manual adjustment: • Your Price equals List Price minus the sum of the three new discounts. `320 = 480 - (50 + 10 + 100)` So, if Your Price is 320, and if you enter 300 as a price override for the manual adjustment, then we need to subtract another 20 from 320 to get to the Net Price of 300. There's a quantity of 2 on the order line, so Extended Amount equals manual adjustment 20 multiplied by Quantity 2 equals $40.

Here's a function you can use with the Apply Manual Adjustments For Goods and Services algorithm:

Function Name	Description
getChargeDefinition	Use the charge definition code to get the view object for the charge definition.

Get the Invoice Price

Next, we calculate the invoice price:



Note

- You use the Create Charge Component with Running Unit Price algorithm. For details, see the Finalize List Price section earlier in this topic.
- If you create a rounding rule for the currency that the charge references, then the algorithm rounds according to the rule.
- The running unit price in the input SDO is 300.000.
- The running unit price in the input SDO is also 300.000. No rounding happens in this example.

Here's what the algorithm does:

What the Algorithm Does	How the Algorithm Does It
1. Get the rounding rules.	-
2. Round.	Round the net price according to the rounding rules. Assume the rules in the example round to the nearest hundredth.
3. Calculate.	Calculate the difference between the rounded value and the value that the algorithm hasn't rounded, then store the difference as a rounding adjustment. Assume you create a rule that rounds to decimal precision two, or hundredth. If the input SDO contained 300.473, then the rounded value would be 300.47, and the adjustment for the net price is 47 cents.

Here are details about entities in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO

ChargeComponent	-	`ChargeComponentId: 9`
		`ChargeId: 1`
		`CurrencyCode: USD`
		`ExtendedAmount: 600 USD`
		`PriceElementCode: QP_NET_PRICE`
		`PriceElementUsageCode: NET_PRICE`
		`SequenceNumber: 1008`
		`SkipRunningPrice: Y`
		`TaxIncludedFlag: N`
		`UnitPrice: 300.00 USD`

Related Topics

- Pricing Rules

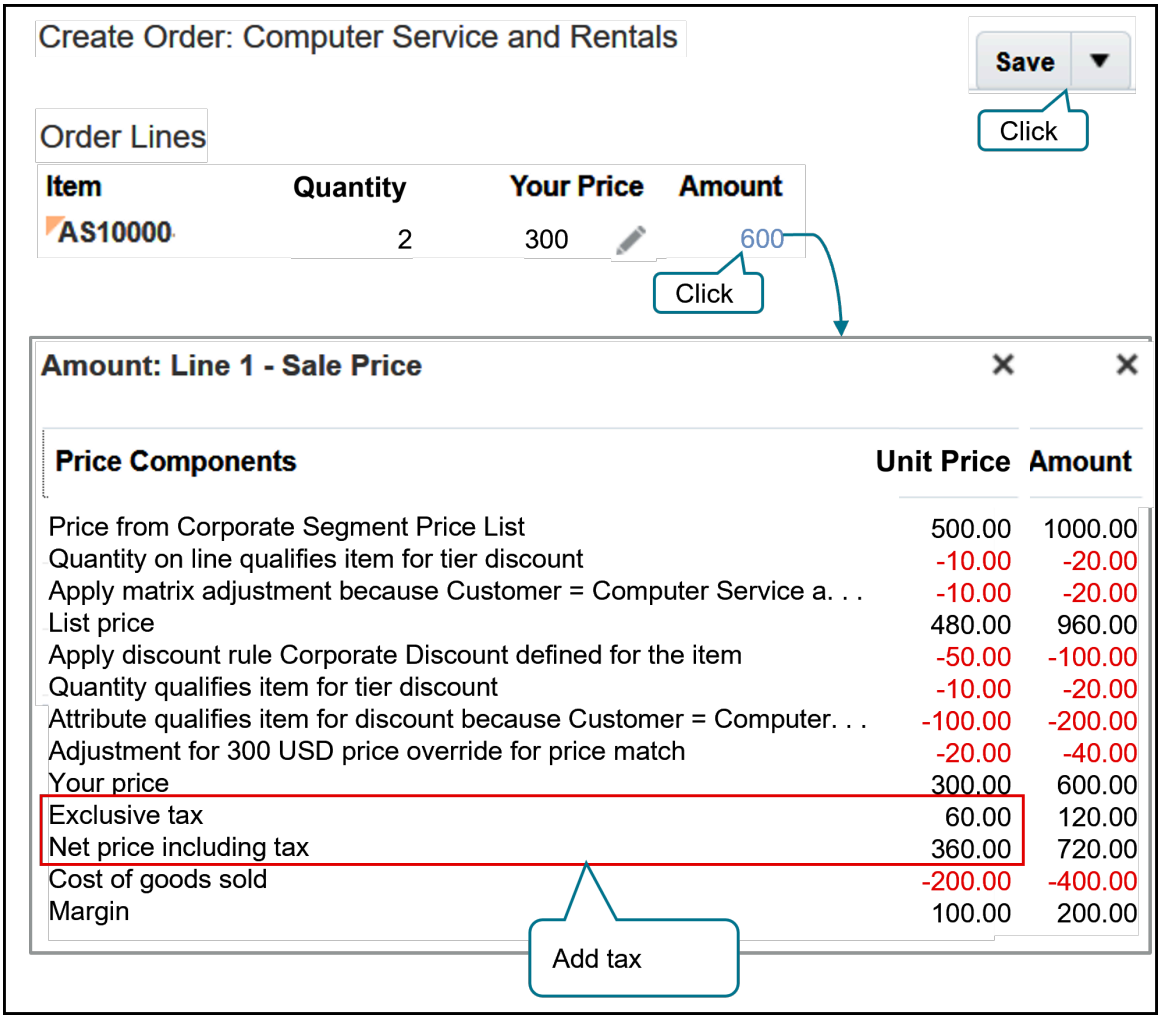
Example Pricing Algorithm, Part 7, Calculate Tax

Learn how pricing algorithms calculate tax.

Here are some ways you can set up tax pricing to meet your specific needs:

- Help make your Order Entry Specialist happy. Use an extensible flexfield and write an order management extension that copies the value from the Exemption Certificate Number attribute on the order header to the order line each time that you add a line. Use this value as the default on the line. This way, the Order Entry Specialist doesn't have to manually enter the value on every order line.
- The tax API uses the product fiscal classification when it calculates tax. Create an interface between the Product Fiscal Classification attribute for the item in Product Management and the tax API.
- Set up a pricing algorithm to create an interface between the Order Date attribute on the sales order and the TransactionDate attribute in the tax API.
- Use the segment of an extensible flexfield to help calculate the assessable value of more than one charge.

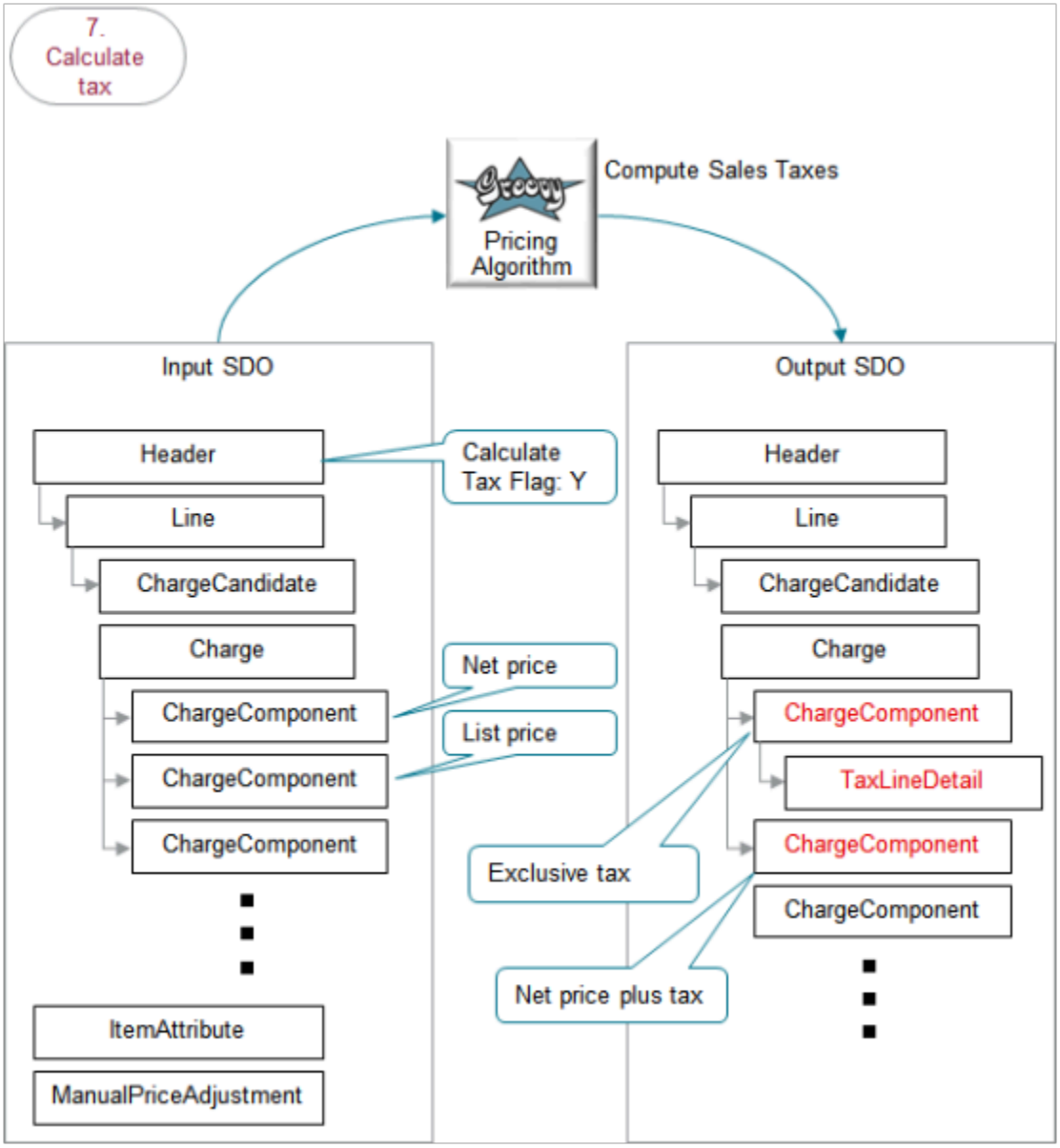
For this example, reprice the sales order in the Order Management work area and see what happens:



Note

- Click **Save**. Pricing reprices the order each time you save or submit the order.
- Click the **amount**.
- Notice the breakdown now includes a line for Exclusive tax and adds the tax to the net price. This tax doesn't affect margin because its exclusive to calculating margin for the item.

Here are the SDOs:



Note

- The output SDO from the Get the Invoice Price step provides the input SDO.
- The CalculateTaxFlag attribute on the header contains Y. This step calculates tax only if CalculateTaxFlag contains Y. For example, if the Quantity attribute on the order line in Order Management contains 0, then Order Management sets CalculateTaxFlag to N, and pricing skips this step.
- The algorithm uses the charge and charge components on the order line in the input SDO to calculate tax.
- Charge components on the input SDO include the net price and list price.
- The TaxLineDetail entity on the output SDO stores tax details for the sales order.
- The three large dots indicate more potential ChargeComponents. The diagram doesn't include them because they're not relevant to this discussion.
- The algorithm uses your set ups in the Pricing Administration work area.

Here's what the algorithm does:

What the Algorithm Does	How the Algorithm Does It
1.Populate tax header.	Use header attributes from the sales order to populate header attributes on the tax.
2. Populate tax lines from charges.	Use values from each charge and each order line to populate values for each tax line.
3. Call the Calculate Tax API.	Call the tax API in Oracle Applications. If Oracle Applications uses a third party tax API, then pricing can use it.
4. Handle tax calculation errors.	Handle any errors that the tax API in Oracle Applications returns to the algorithm.
5. Add the charge component for inclusive tax.	The tax API in Oracle Applications calculates the tax line detail for each tax. For example, it calculates city tax, county tax, state tax, federal tax, and so on. The tax amount includes a boolean attribute on the tax line detail that specifies whether the tax is inclusive. For each charge, the API adds up the tax amounts for the inclusive tax, creates a charge component for them, then stores these tax values in one component charge for the inclusive tax. It stores the tax as the unit price for the total of the tax amounts. The API stores the value in one charge component. It stores the net price excluding the exclusive tax. Its the net price **minus** the total **inclusive** tax.
6. Add the charge component for exclusive tax.	The tax API uses the same logic that it uses to calculate and store exclusive tax, except it stores the net price **plus** the total **exclusive** tax. The tax API might store inclusive and exclusive tax for each charge, depending on how you set up the API. In this example, the sales order has only exclusive tax.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO
ChargeComponent for exclusive tax (parent of TaxLineDetail)	-	**ChargeComponentId: 10** **ChargeId: 1** **CurrencyCode: USD** **ExtendedAmount: 120.0 USD** **PriceElementCode: QP_EXCLUSIVE_TAX** **PriceElementUsageCode: EXCLUSIVE_TAX** **RollupFlag: N** **SequenceNumber: 1009** **SkipRunningPrice: Y** **TaxIncludedFlag: N** **UnitPrice: 60.0 USD** There's only one exclusive tax, so the output SDO includes only one ChargeComponent for exclusive tax.

Entity	Attributes in Input SDO	Attributes in Output SDO
		Unit Price is the tax amount.
TaxLineDetail	-	ChargeComponentId: 10 CurrencyCode: USD HeaderCurrencyCode: USD TaxAmount: 60USD TaxAmountInclusiveFlag: N TaxJurisdictionCode: FUS_STCC JURIS_UES TaxJurisdictionName: FUS_STCC JURIS_UES TaxLineDetailId: 1 TaxRate: 20 TaxRateCode: VAT20 TaxRateId: 300100148455853 TaxRateName: VAT20 TaxRegimeCode: FUS_STCC_REGIME_UES TaxRegimeName: FUS_STCC_REGIME_UES TaxStatusCode: FUS_STCC_TAX_STD-UES TaxableAmount: 300 USD
ChargeComponent for net price plus tax (below TaxLineDetail)	-	ChargeComponentId: 11 ChargeId: 1 CurrencyCode: USD ExtendedAmount: 720.00 USD PriceElementCode: QP_NET_PRICE_PLUS_TAX PriceElementUsageCode: NET_PRICE_PLUS_TAX PriceValidFrom: 2019-05-29 @ 06:35:10 RollupFlag: N SequenceNumber: 1010 SkipRunningPrice: Y TaxIncludedFlag: N UnitPrice: 360.00 USD Unit Price contains the net price of 300 plus tax of 60. Extended Amount is quantity 2 multiplied by Unit Price 360 equals 720.00.

Note

- If the sales order includes exclusive tax, then the algorithm creates one charge component for the exclusive tax and another one for net price plus tax.
- If the sales order includes inclusive and exclusive tax, then the algorithm creates one charge component for the exclusive tax, another one for the inclusive tax, and another one for net price plus tax.

Here are some of the functions you can use with the Compute Sales Taxes algorithm. All functions are view object lookups, except for createChargeComponent and getPartialPeriodDuration, which are scripts:

Function Name	Description
createChargeComponent	Script that creates components for the tax charge.

Function Name	Description
getChargeDefinition	Get the charge definition.
getCurrencyRelatedAttributes	Get the currency attributes.
getLedgerId	Get the value that uniquely identifies the ledger.
getOKCUomMappings	Query the time unit mappings to get details from the contracts okc_ tables.
getPartialPeriodDuration	Script that calls the contracts API to get the duration for part of the time frame when the tax applies.

Related Topics

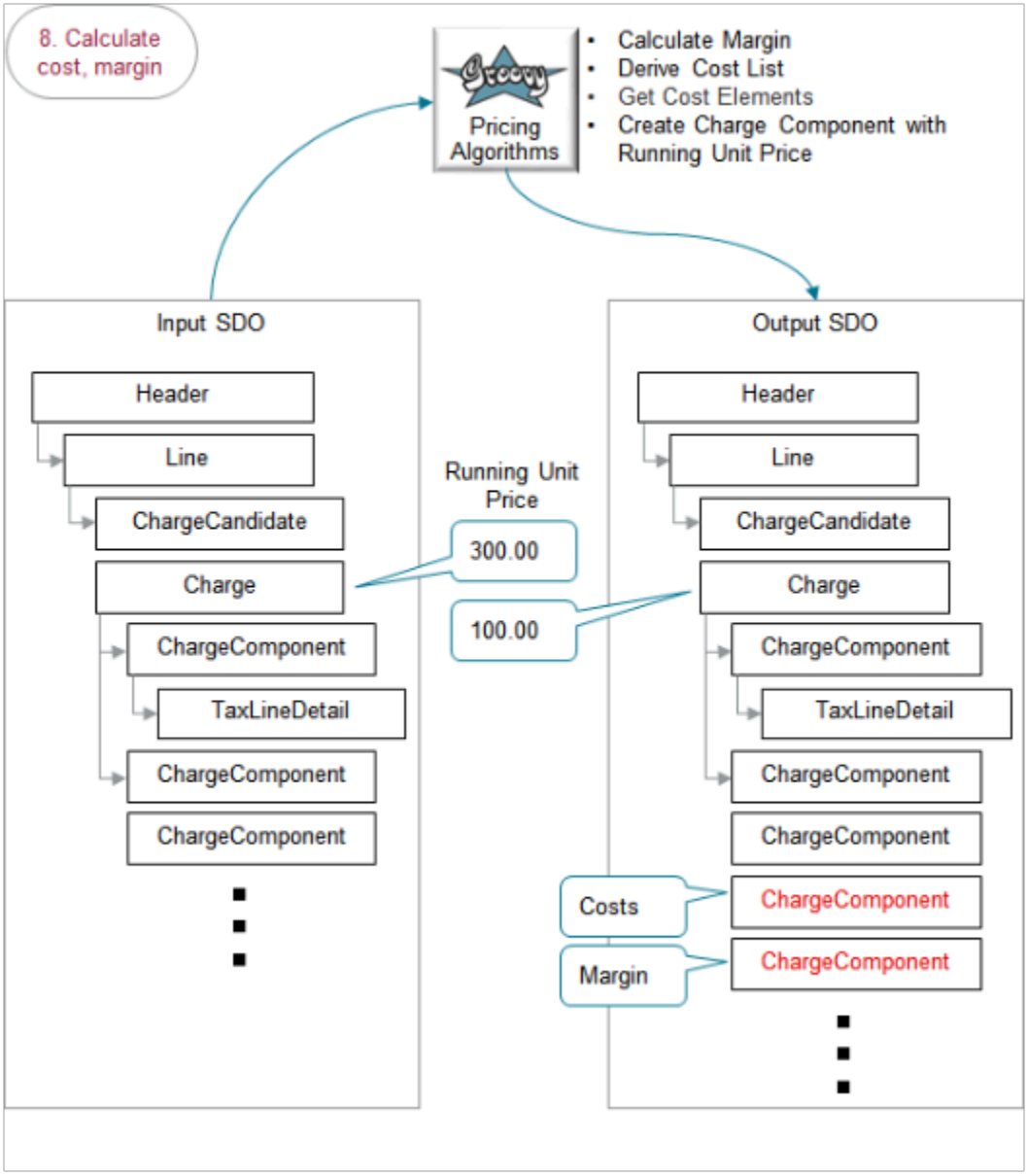
- Pricing Rules
- Manage Tax in Oracle Pricing

Example Pricing Algorithm, Part 8, Calculate Cost and Margin

Now we calculate cost, and then margin.

Before we get into the details, note that you can get actual costs from a source that isn't the cost list. For example, get the average cost from the Oracle Costing Service and use it when you calculate the margin. You can also use the service to calculate markup and base list price. You call the service through Oracle Integration Cloud or Integration Cloud Service. You can call a service that resides outside of Oracle Pricing from the pricing algorithm to calculate costs, margin, markups, or base list price.

Now let's get back to our example. Here are the SDOs:



Note

- The output SDO from the Calculate Tax step provides the input SDO.
- The running unit price on the input SDO is 300 USD.
- The running unit price on the output SDO is 100 USD.

Here's what the algorithms do:

What the Algorithms Do	How the Algorithms Do It
1. Determine cost list.	Get the cost list that the pricing strategy references. Look up costs in sequential order according to precedence. Stop looking as soon as we find the cost list that applies for this customer and item.

What the Algorithms Do	How the Algorithms Do It
2. Get cost elements.	Get cost elements from the cost charges on the cost list. A specific item takes precedence over an All Items specification on the list.
3. Create charge components for costs.	Write each cost that remains after finishing the get cost elements step. Create a separate charge component in the output SDO for each element. This example includes only one charge component because there's only one cost. The algorithm adjusts the running unit price in the output SDO for each charge component it adds.
4. Delete cost elements.	Delete them so they don't cause problems the next time the algorithm runs.
5. Calculate margin.	Calculate the final running unit price.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO

ChargeComponent
 	
(for the costs)
	-	`ChargeComponentId: 12`
		`ChargeId: 1`
		`CurrencyCode: USD`
		`ExtendedAmount: -400.00 USD`
		`PriceElementCode:`
		`QP_COST_OF_GOODS_SOLD`
		`PriceElementUsageCode: COST`
		`SequenceNumber: 1011`
		`SourceTypeCode: COST_LIST`
		`UnitPrice: -200 USD`
		
		Unit Price is the amount of the cost.

ChargeComponent (for the margin)	-	`ChargeComponentId: 13`
		`ChargeId: 1`
		`CurrencyCode: USD`
		`ExtendedAmount: 200.00 USD`
		`PriceElementCode: QP_MARGIN`
		`SequenceNumber: 1012`
		`SkipRunningPrice: Y`
		`TaxIncludedFlag: N`
		`UnitPrice: 100 USD`
		
		Unit Price is the margin.

Here are some of the functions you can use with the algorithms:

Algorithm Name	Function Name	Description
Calculate Margin	-	No functions are available.

Algorithm Name	Function Name	Description
Derive Cost List	getCostCharge	Get the cost charges.
Get Cost Elements	getCosts	Get costs for the item.
Create Charge Component with Running Unit Price	-	See the Finalize List Price step.

Related Topics

- Pricing Rules

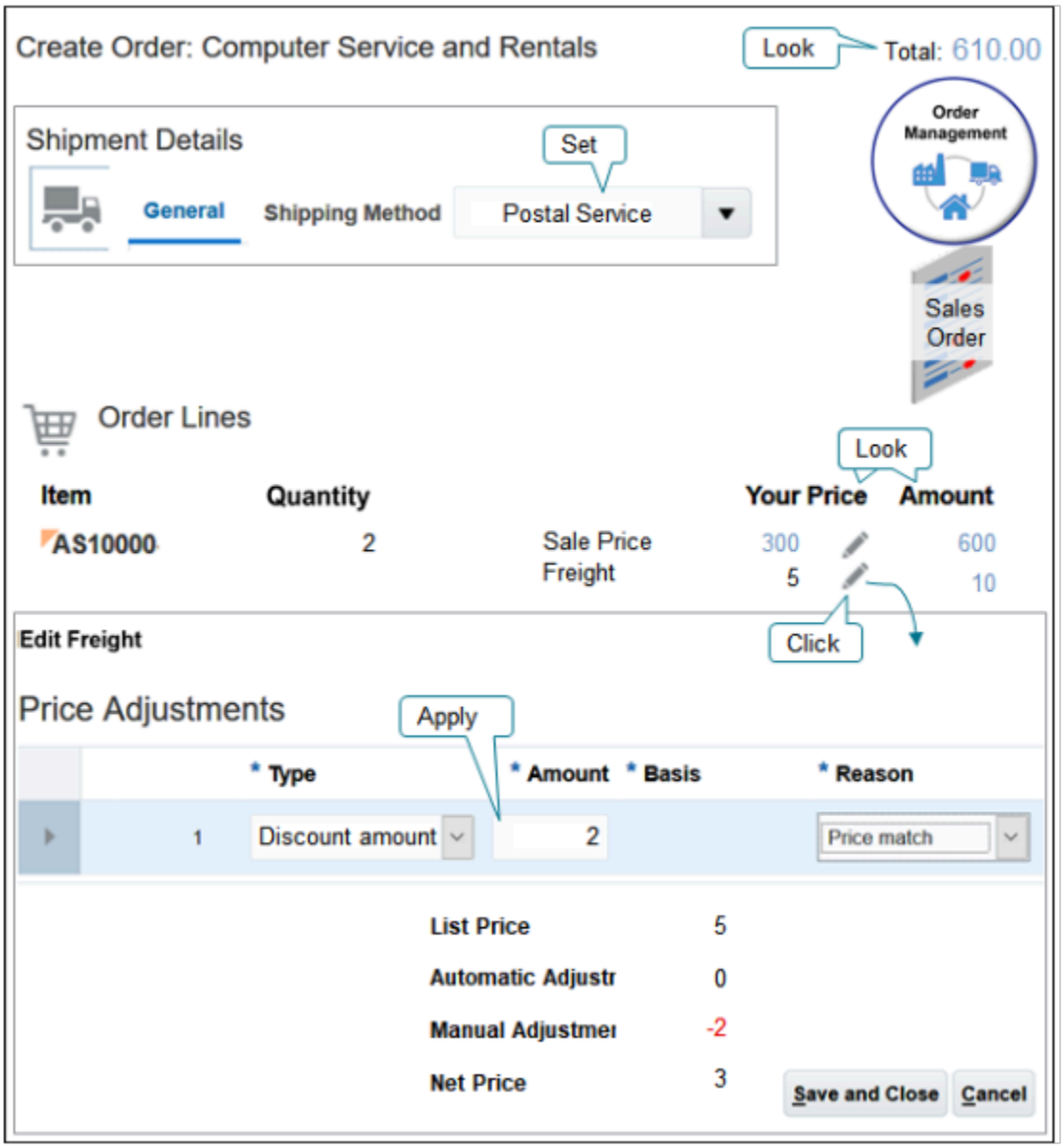
Example Pricing Algorithm, Part 9, Calculate Shipping

Learn how pricing algorithms calculate shipping.

Here are some example set ups you can do on your algorithms to calculate shipping that meet your specific requirements:

- Apply an adjustment according to a matrix. For example, create shipping charges according to the weight of the item, then calculate shipping charge adjustments according to a matrix rule.
- Create flat rate shipping charges according to the value that the Order Entry Specialist enters in an extensible flexfield on the order header or order line. Your algorithm can use the value as an input when calculating the charge.

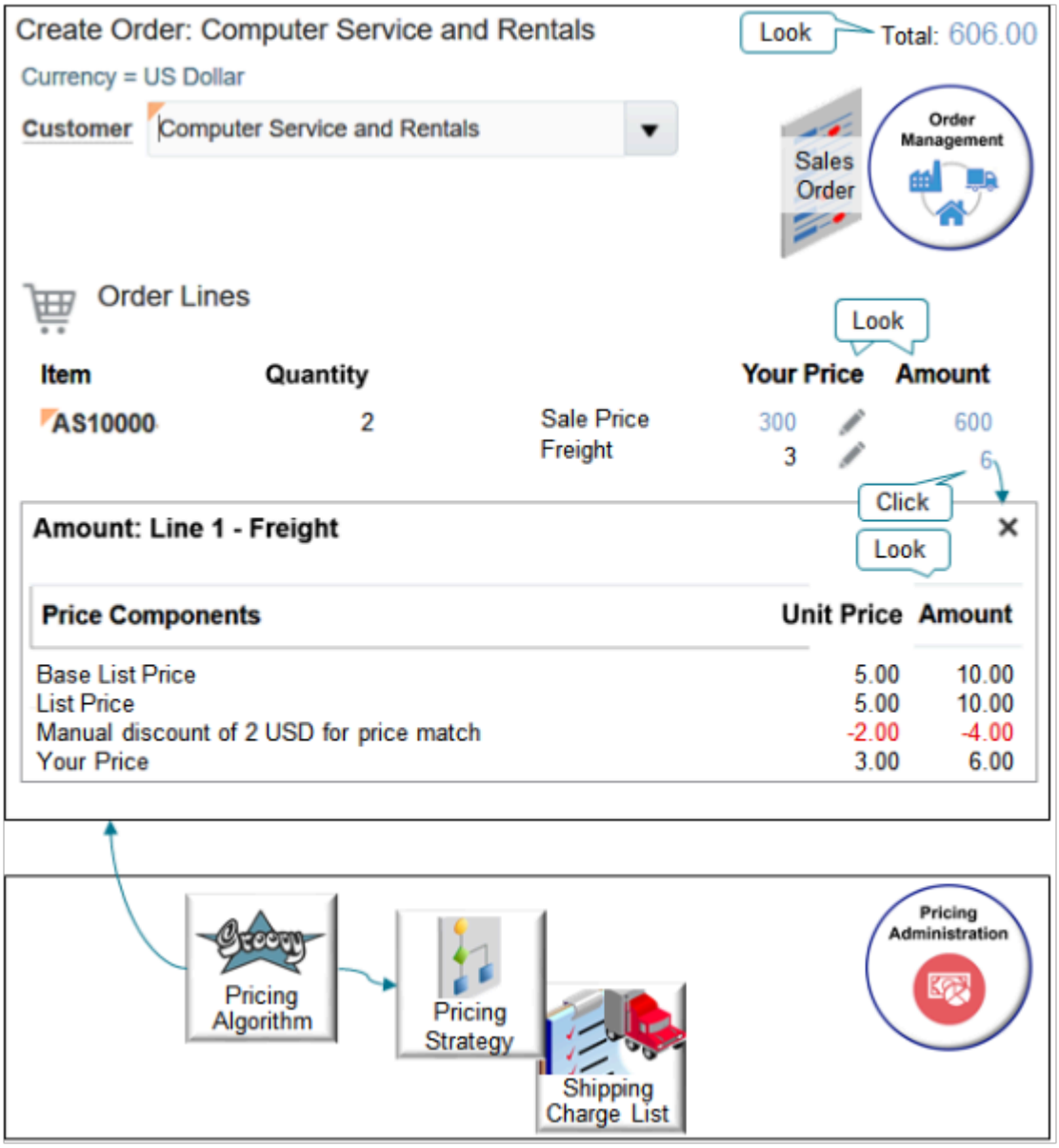
For our example, we are almost done. Let's see how shipping works in the sales order when you manually adjust the shipping charge:



Try it:

1. Open your sales order in Order Management.
2. Click Shipment Details, then set the Shipping method.
3. Notice the order line.
4. Notice that the order line contains a charge for Freight of $5 each, and an extended value in the Amount column of 10 because Quantity is 2.
5. Notice the order total is $610.
6. Click the value in the Amount column.

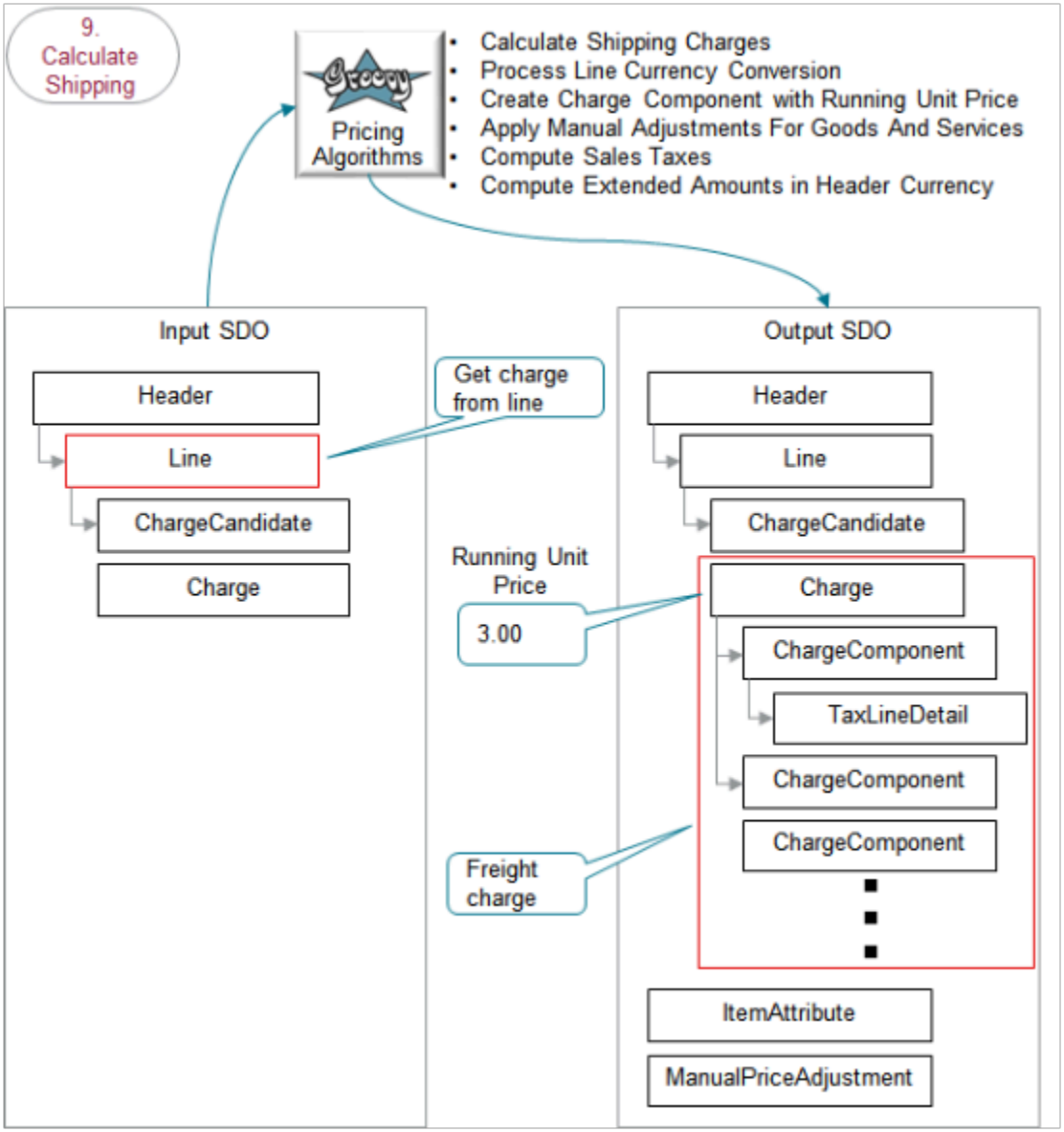
7. In the Price Adjustments dialog, add an amount discount for $2, set Reason to Price Match, then save your work.:



Note

- The value for Freight is now 3.
- The extended value in the Amount column is now 6.
- The order total is $606.

Here are the SDOs:



Note

- The output SDO from the Calculate Cost and Margin step provides the input SDO.
- The algorithms store details in charge components in the freight charge on the output SDO. They use a separate component for each of:
    - Base list price
    - List price
    - Manual adjustments
    - Net price
    - Exclusive tax
    - Net price plus tax

Here's what the algorithms do:

What the Algorithms Do	How the Algorithms Do It
1. Get shipping charge lists.	Examine the shipping charge list that the pricing strategy references. Make sure it includes at least one charge for the item. Calculate the shipping charge for each order line.
2. Convert currency on the order line.	If the shipping charge list doesn't use the same currency that the sales order uses, then convert it so it does use the same currency. For example, if the sales order uses RMB but the shipping charge list uses USD, then covert USD to RMB on the shipping charge list.
3. Create candidates for shipping charges.	Create a separate candidate for each shipping charge that the shipping charge list specifies. Pricing doesn't come predefined to apply flat rate shipping charges. The predefined set up only applies shipping charges for each item.
4. Create charge components for the base list price.	Create a charge component for each shipping charge on the order line according to the base list price that the shipping charge list specifies.
5. Create charge component for the list price.	Create a charge component for each shipping charge on the order line according to the list price that the shipping charge list specifies.
6. Apply manual adjustments.	If the Order Entry Specialist applies a manual adjustment to the shipping charge on the sales order, then the algorithm creates and calculates the charge components it needs to apply the adjustment. It does this for each adjustment that you apply.
7. Create charge component for the net price.	-
8. Calculate tax.	Apply tax rates to the charges.
9. Calculate currency amounts for the order header.	Calculate the order total. Use the currency that the sales order specifies.

Here are some of the functions you can use with the algorithms. All these functions are view object lookups, except for createChargeComponent and getPartialPeriodDuration, which are scripts:

Algorithm Name	Function Name	Description
Calculate Shipping Charges	checkAllowedCurrency	Validate the currency against the list of currencies that the pricing strategy allows.
-	createChargeComponent	Create a charge component.
-	getChargeDefinitionName	Get the name of the charge definition.

Algorithm Name	Function Name	Description
-	getOKCUomMappings	Query the corresponding mappings in the okc_ contracts tables to call the contracts API.
-	getPartialPeriodDuration	Use the Service Start Date and Service End Date on the order line to get the partial price period.
-	getShippingChargeLists	Get the shipping charge lists for the item in the pricing strategy.
-	getShippingCharges	Get the shipping charges.
-	getUomTranslation	Query the unit of measure table for the UOM code, such as Ea.
Process Line Currency Conversion	checkAllowedCurrency	Validate the currency on the order line against the list of currencies that the pricing strategy allows.
Create Charge Component with Running Unit Price Apply Manual Adjustments For Goods And Services Compute Sales Taxes	-	See the earlier sections in this topic that describe these algorithms.
Compute Extended Amounts in Header Currency	getCurrencyPrecision	Get the precision for the currency.

Related Topics

- Pricing Rules

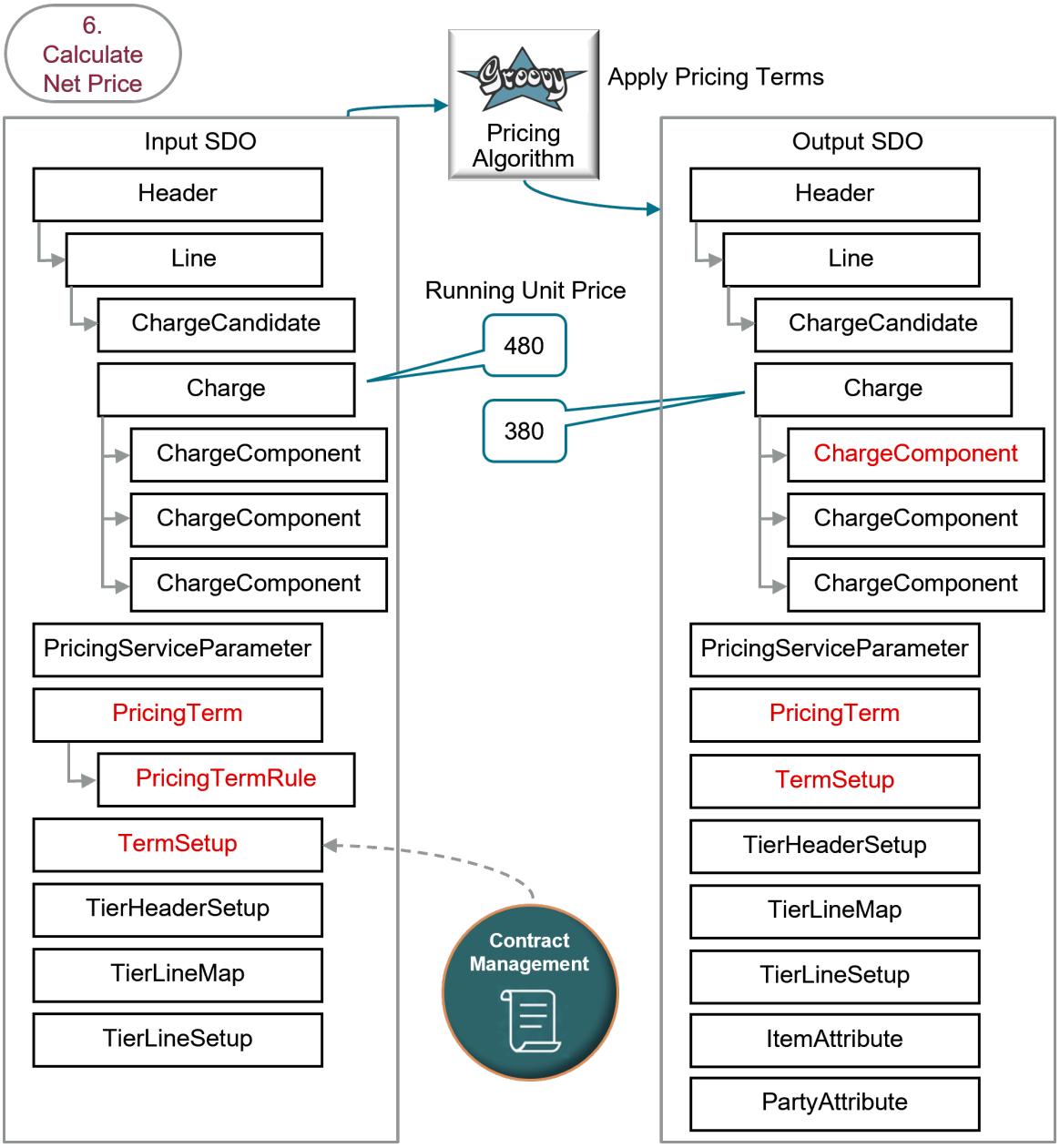
Example Pricing Algorithm, Other Setups

Learn how pricing algorithms do other calculations.

Apply Pricing Terms for Sales Agreement

If you set up Order Management to use sales agreements, then the algorithm uses the agreement to apply pricing terms.

Here are the SDOs:



Note

- The output SDO from the Calculate Net Price step provides the input SDO.
- The running unit price in the input SDO is 480.
- Pricing applies adjustments from the sales agreement before it applies adjustments that the pricing strategy specifies. So, it runs this step immediately before it applies adjustments from the pricing strategy in the Calculate List Price step.
- The TermSetup entity contains the rule that you specify on the sales agreement in Contract Management.
- You use Order Management and Contract Management to set up the agreement, including terms that specify price, such as discounts. For details, see *Overview of Setting Up Sales Agreements in Order Management*.
- The running unit price in the output SDO is 480 minus the 100 discount equals 380.

Here's what the Apply Pricing Terms algorithm does:

1. Find matching charges.
2. Get the pricing basis.
3. Apply simple adjustments.
4. Process other adjustment types.
5. Apply tiered adjustments.
6. Apply matrix adjustments.
7. Process the TermCustomAdjustmentFlag attribute.
8. Process errors in the sales agreement.

The flow is similar to the Calculate List Price step except you can use only a simple rule or matrix rule with a sales agreement. You can't use a tier rule.

For details about the functions you can use, see the Apply Discounts from Pricing Strategy section, earlier in this topic.

Here are details about attributes in the SDOs:

Entity	Attributes in Input SDO	Attributes in Output SDO

ChargeComponent	-	`ChargeComponentId: 5`
		`ChargeId: 1`
		`CurrencyCode: USD`
		`ExplanationMessageName: QP_SALES_AGREEMENT_ADJ`
		`ExtendedAmount: -200 USD`
		`PriceElementCode: QP_SALES_AGREEMENT_ADJ`
		`PriceElementUsageCode: PRICE_ADJUSTMENT`
		`PriceValidFrom: 2019-05-29 @ 05:55:29`
		`SequenceNumber: 1004`
		`SourceId: 300100176714435`
		`SourceTypeCode: PRICING_TERM`
		`UnitPrice: -100 USD`
		The Unit Price attribute contains the $100 discount that the sales agreement specifies.

Related Topics

- Pricing Rules

Setup

Manage Pricing Algorithms

In this example, you remove tax calculations that a predefined pricing algorithm does so the price of each sales order that your users create in Order Management doesn't include tax.

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

Summary of the Steps

1. Examine the current behavior.
2. Create a new version of the algorithm.
3. Modify the pricing algorithm.
4. Verify your set up.

Examine the Current Behavior

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area, then create a new sales order.
3. On the New Sales Order page, add a customer, add an item, then click **Add**:

Attribute	Value
Customer	Computer Service and Rentals
Select Item	AS54888

4. In the Amount column, click the **link**, then notice that the Line dialog includes an amount for tax:

Price Component	Amount
List Price	2,500
Discount	100
Tax	200
Net Price Plus Tax	2,600

Create a New Version of the Algorithm

1. Sign out of Order Management, then sign into Oracle Applications with the privileges that you need to administer pricing.
2. Go to the Pricing Administration work area.
3. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.

4. On the Manage Algorithms page, locate the algorithm you must modify.

   Here's a summary of the attributes:

Attribute	Description
Name	Predefined pricing algorithms provide a wide variety of functionality. To locate the algorithm you must modify, read the links in the Name column, click one that looks promising, then examine the steps. It might be necessary to examine more than one algorithm. If you create a new algorithm, then capitalize each word in the Name attribute and include a space between each word. Suffix the name with the text Custom.
Version	Each predefined algorithm uses version Zero or 1. Versions higher than 1 are versions you created. Pricing uses the latest version in the runtime environment.
Status	Here are the statuses: o **In Progress**. Editable but not available for Order Management. A pricing algorithm you're currently developing is typically In Progress. o **Published**. Not editable and available in the calling application, such as Order Management. o **Inactive**. No longer in use. Not available in the calling application.
Public	You typically use No. Use Yes only if you plan to use the algorithm in more than one calling application, such as in Order Management and Contract Management.

   For this example, you disable tax calculation for a sales transaction, so, click the **row** that displays Price Sales Transactions in the Name column. For most situations, you modify the Price Sales Transactions algorithm to implement the logic you need.
5. Click **Actions > Create Version**.

   Notice that Pricing creates a new version and sets the Status to In Progress.

Modify the Pricing Algorithm

1. In the Name column of the version you created, click **Price Sales Transactions**.
2. On the Edit Algorithm page, examine the step names to verify that this pricing algorithm can implement the behavior you require.

   This algorithm includes steps that calculate tax, such as Compute Sales Tax, so its likely you can modify it to implement the behavior you need.
3. Click the **row** that contains Compute Sales Tax in the Name column, click the **Delete icon**, then click **Save**.
4. Click View Source, then make sure the dialog that displays includes a message like this.

```
The following script has passed syntax check
```

5. On the Manage Algorithms page, click the **row** that contains the highest version of Price Sales Transaction, then click **Actions > Publish**.
   You might receive an error like this.
   ```
   Another user changed row with primary key.
   ```
   The Pricing Administration work area might indicate that it published the pricing algorithm. However, if you sign out, then sign in, it might display as not published. To avoid this problem, click **Actions > Deactivate**, click **Actions > Activate**, then click **Actions > Publish**.

Verify Your Set Up

1. Sign out of Pricing, then sign into Order Management.
2. Create a sales order, add a customer to the order, then add an item.
3. In the Amount column, click the **link**, then make sure the Line dialog doesn't include an amount for tax:

Price Component	Amount
List Price	2,500
Discount	100
Net Price Plus Tax	2,400

Reset Pricing Algorithms to the Predefined Version

You can delete all of your pricing algorithms that are at version 1 and move back to version 0, which is the algorithm's predefined version. If you reset your algorithms to version 0, then you can get the latest predefined versions when you move to the latest update without having to continually promote your algorithms.

Use this feature when you:

- Want to accept new features when you update without having to manage your algorithms.
- Plan to retire your modified algorithms and use only the predefined ones.
- Accidentally did the Promote All action, need to undo your changes, and use the predefined algorithms.
- Need to start over and use only the predefined behavior.

If you do the Delete All Version 1 action, and if all your algorithms remain at version 0 on subsequent updates, then you don't need to do the Promote All action the next time you move to the latest update. Oracle Pricing uses version 0 as the base version on new updates.

Try it:

1. Go to the Manage Algorithms page, then make sure your algorithms are at version 0 or version 1.
2. Click **Actions > Delete All Version 1**.
3. In the warning dialog, click Yes.
4. Notice the details on the Confirmation dialog, then click **OK**.
5. Notice that all your algorithms are now at version 0.

Note

- If any algorithm is at version 1 or higher, then Pricing disables the Delete All Version 1 action. You must remove all extensions before you can use the action.
- You can't reverse the Delete All Version 1 action, so do the action in a test environment before you do it in your production environment.
- If you need to modify an algorithm after you update, then use the Promote All action, and then create a new version.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Algorithm Steps
- Pricing Algorithms

Test and Troubleshoot Pricing Algorithms

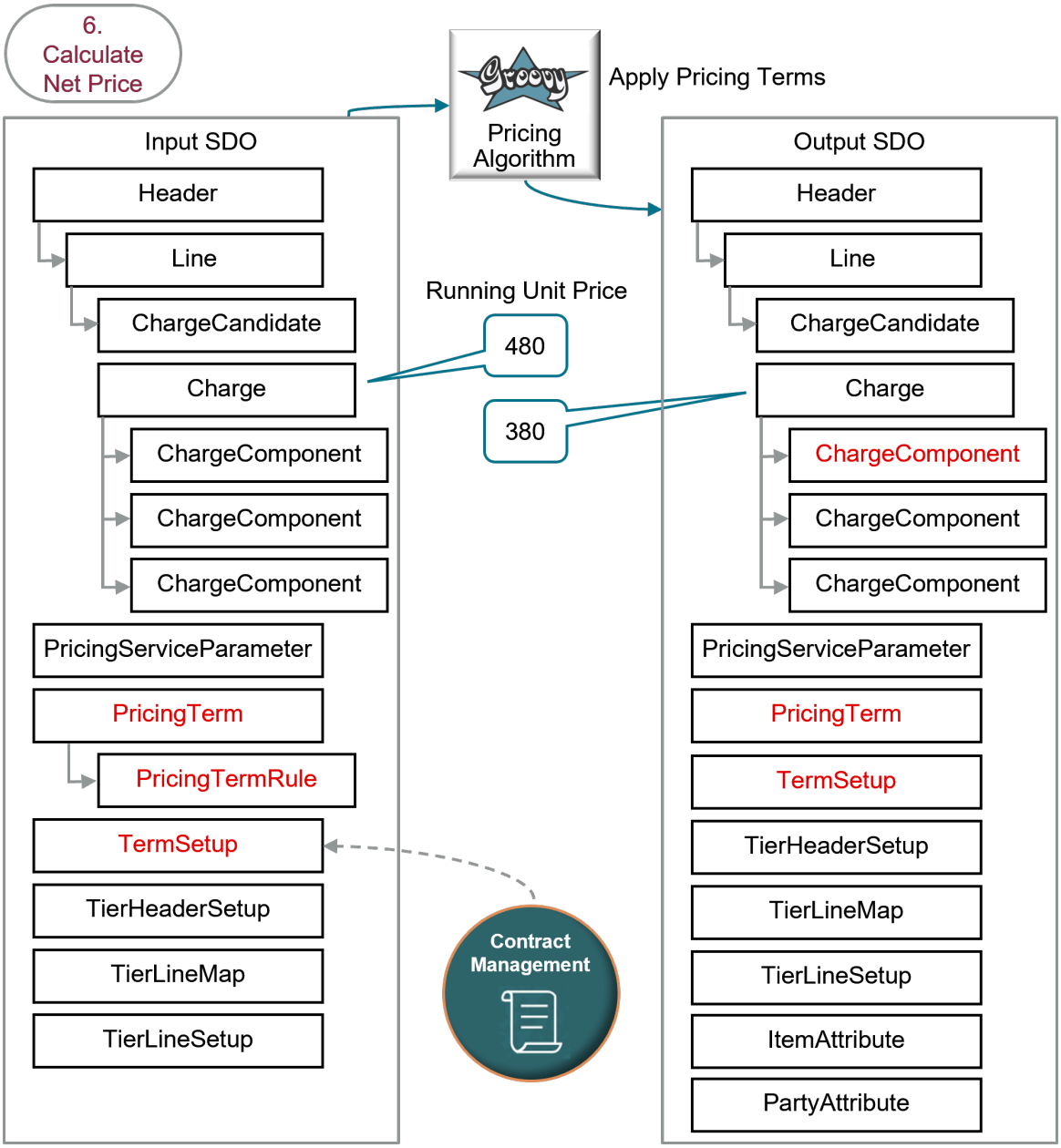
Test your pricing algorithm set up. If it doesn't work, troubleshoot.

Test Your Setup

Test your pricing algorithm set up if you find a problem or prefer to test the set up before you publish:

1. Open your pricing algorithm for editing.
2. Click **Test**, then click **Actions > Add Row**.

3. In the Test Input area, specify inputs to the test:



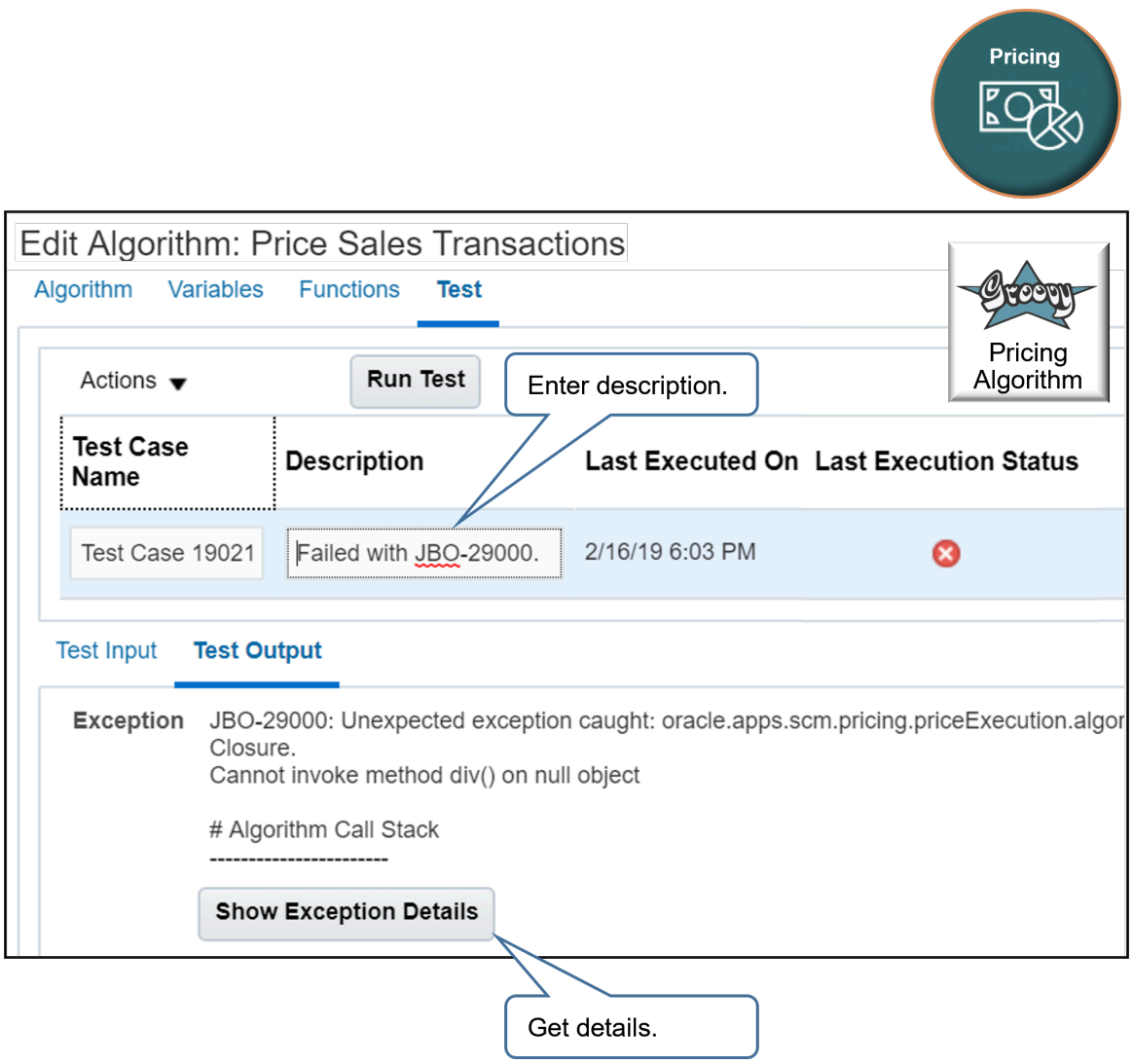
Note

- Examine the predefined set up. Some algorithms come predefined with tests already set up. For example, the Price Sales Transactions algorithm comes predefined with the PriceRequest variable already set up for testing.

  If necessary, modify the Variable Value so it reflects your changes.
- If you created a new variable during set up, or need to test performance of a predefined one, then add it now.
- Add variables that are of type Input, or type Input and Output.

For example, add the Boolean variable GetAllUserDefinedAttributeFlags.

4. In the Variable Value column, add one of:
   - Groovy expression.
   - Literal string.
   - XML formatted code.
   - Check mark for a Boolean data type.
5. Click **Run Test**, then examine the output:



Note

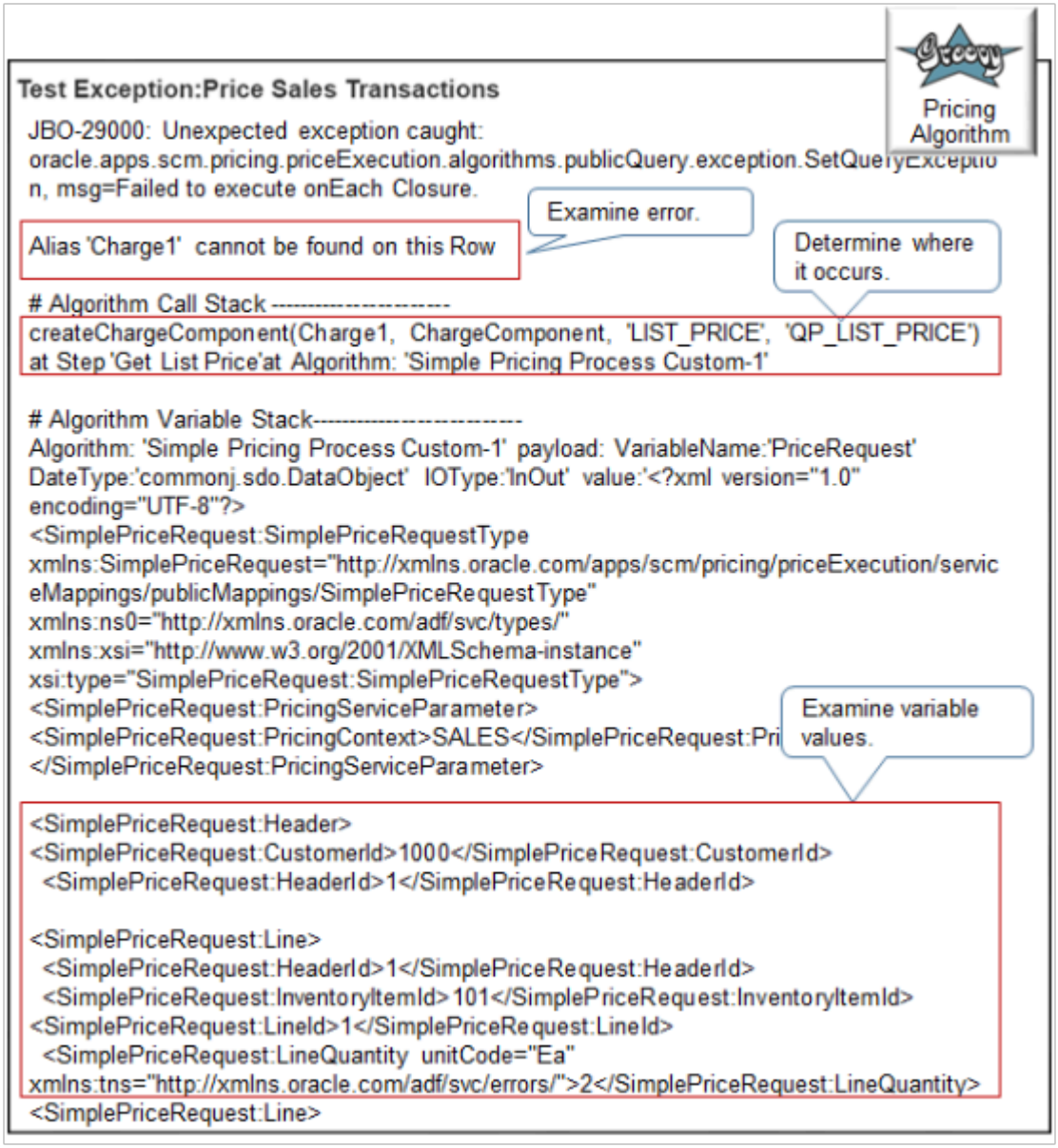
- For future reference and tracking purposes, modify the Test Case Name and Description, as necessary.
- If the test fails, click **Show Exception Details** to troubleshoot your set up.
- Pricing saves your test case across updates for algorithms that are at version 2 or higher. For background, see *Performing Your Quarterly Update (Doc ID 2337485.1)*.

Study some examples that test algorithms. For details, see *Overview of Pricing Use Cases*.

Troubleshoot Your Setup

1. Click **Show Exception Details**.

   Here's the dialog that displays:



2. Examine the error message. It contains the exact Groovy code that causes the problem, such as `Charge1`. Its usually the second line in the details. For example:

```
Alias 'Charge1' cannot be found on this Row
```

3. Determine where the error happens. Its usually immediately below the text Algorithm Call Stack.

   The call stack includes the statement, step, and algorithm that contains the Groovy code that causes the problem. For example:

   ```
   createChargeComponent(Charge1, ChargeComponent, 'LIST_PRICE', 'QP_LIST_PRICE') at Step 'Get List
    Price'at Algorithm: 'Simple Pricing Process Custom-1'
   ```

   In this example, the code happens in the createChargeComponent statement on the Get List Price step of the Simple Pricing Process Custom-1 algorithm.
4. Examine the variable values in the Algorithm Variable Stack section. This section includes variable values that exist immediately before the error happens:
   - Verify data type. For example, InventoryOrganizationId is a numeric value. It must contain numeric data, such as 204, not text data, such as Computer Service and Rentals.
   - Verify data value. For example, if you expect unitCode to equal Ea, but it contains Box, that's not good.
5. Examine changes you made that might cause the error:
   - Examine your recent changes to the algorithm or algorithm step. Use details from the Algorithm Call Stack section to determine where to look first.
   - Examine changes you made to the service mapping. Changes can result in errors in the algorithm. For example, make sure you set up sources and attributes correctly.
   - Examine your code for typographical errors.
   - Make sure you declared all your variables.
   - Make sure you defined all attributes that the algorithm uses.

Here's the complete Test Exception details for this example.

```
JBO-29000: Unexpected exception caught:
 oracle.apps.scm.pricing.priceExecution.algorithms.publicQuery.exception.SetQueryException, msg=Failed to
 execute onEach Closure.

Alias 'Charge1' cannot be found on this Row

# Algorithm Call Stack -----------------------
createChargeComponent(Charge1, ChargeComponent, 'LIST_PRICE', 'QP_LIST_PRICE') at Step 'Get List Price'at
 Algorithm: 'Simple Pricing Process Custom-1'

# Algorithm Variable Stack---------------------------
Algorithm: 'Simple Pricing Process Custom-1' payload: VariableName:'PriceRequest'
 DateType:'commonj.sdo.DataObject' IOType:'InOut' value:'<?xml version="1.0" encoding="UTF-8"?>
<SimplePriceRequest:SimplePriceRequestType xmlns:SimplePriceRequest="http://xmlns.oracle.com/apps/
scm/pricing/priceExecution/serviceMappings/publicMappings/SimplePriceRequestType" xmlns:ns0="http://
xmlns.oracle.com/adf/svc/types/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="SimplePriceRequest:SimplePriceRequestType">
<SimplePriceRequest:PricingServiceParameter>
<SimplePriceRequest:PricingContext>SALES</SimplePriceRequest:PricingContext>
</SimplePriceRequest:PricingServiceParameter>

<SimplePriceRequest:Header>
 <SimplePriceRequest:CalculatePricingChargesFlag>true</SimplePriceRequest:CalculatePricingChargesFlag>
 <SimplePriceRequest:CalculateShippingChargesFlag>false</SimplePriceRequest:CalculateShippingChargesFlag>
 <SimplePriceRequest:CalculateTaxFlag>false</SimplePriceRequest:CalculateTaxFlag>
 <SimplePriceRequest:CustomerId>1000</SimplePriceRequest:CustomerId>
 <SimplePriceRequest:HeaderId>1</SimplePriceRequest:HeaderId>
 <SimplePriceRequest:SellingBusinessUnitId>204</SimplePriceRequest:SellingBusinessUnitId>
 <SimplePriceRequest:SellingLegalEntityId>204</SimplePriceRequest:SellingLegalEntityId>
 <SimplePriceRequest:TransactionTypeCode>ORA_SALES_ORDER</SimplePriceRequest:TransactionTypeCode>
```

```
</SimplePriceRequest:Header>

<SimplePriceRequest:Line>
 <SimplePriceRequest:HeaderId>1</SimplePriceRequest:HeaderId>
 <SimplePriceRequest:InventoryItemId>101</SimplePriceRequest:InventoryItemId>
 <SimplePriceRequest:InventoryOrganizationId>204</SimplePriceRequest:InventoryOrganizationId>
 <SimplePriceRequest:LineCategoryCode>ORDER</SimplePriceRequest:LineCategoryCode>
 <SimplePriceRequest:LineId>1</SimplePriceRequest:LineId>
 <SimplePriceRequest:LineQuantity unitCode="Ea" xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/">2</
SimplePriceRequest:LineQuantity>
<SimplePriceRequest:Line>
```

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Algorithm Steps
- Pricing Algorithms

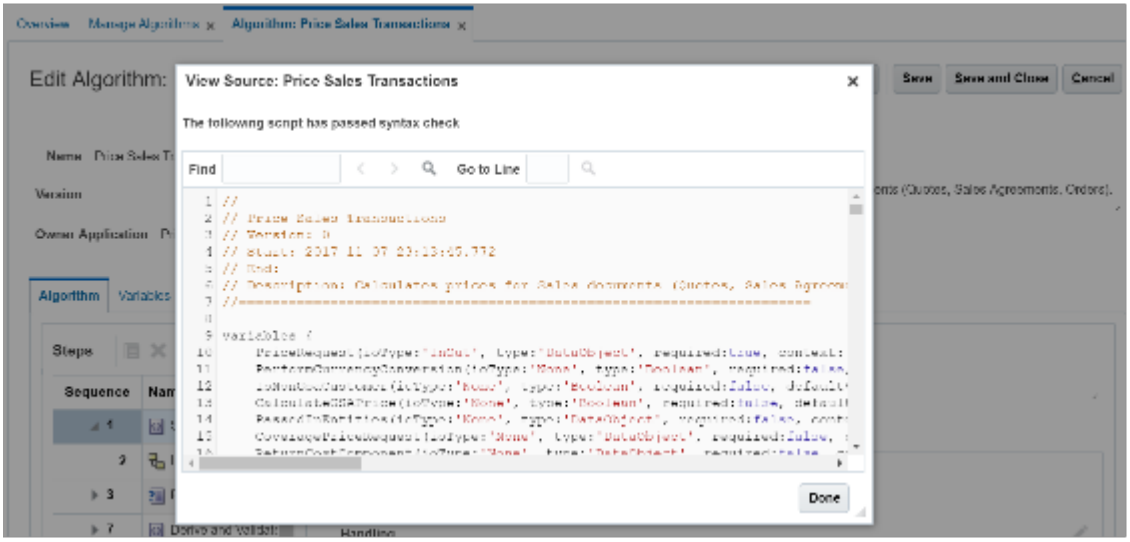
Verify Source Code in Pricing Algorithms

You can click View Source to verify the source code of a pricing algorithm.

1. Go to **Home > Order Management > Pricing Administration > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, locate the algorithm you must modify.
3. Click **Actions > Create Version**.
4. Modify the pricing algorithm so it meets your requirements.
5. Along the top of the Edit Algorithm page, Click **View Source**, then make sure the dialog that displays includes this message.

   ```
   The following script has passed syntax check
   ```

   For example:



Note

- View Source examines and validates your syntax. It then displays the Groovy code that defines the algorithm in a dialog.
- Use it before you publish your pricing algorithm.
- You can also use it to get a complete picture of how you set up the algorithm, or to troubleshoot between versions of the algorithm.
- For example, copy the source code of the working version and of the nonworking version, paste it into a code comparison application, then examine the difference to help identify the code that might be causing the problem.

Here's some code that includes the kind of details that are available when you click View Source.

```
//=====================================================================
// Price Sales Transactions
// Version: 2
// Start: 2016-12-07 22:47:02.528
// End:
// Description: Calculates prices for Sales documents (Quotes, Sales Agreements, Orders).
// Version 2 Comments: Calculates prices for Sales documents (Quotes, Sales Agreements, Orders).
//=====================================================================

variables {
 PriceRequest(ioType:'InOut', type:'DataObject', required:true,
 context:'PricingInternal.PriceRequestInternal')
 PerformCurrencyConversion(ioType:'None', type:'Boolean', required:false, defaultValue:false)
 IsNonGsaCustomer(ioType:'None', type:'Boolean', required:false, defaultValue:true)
 CalculateGSAPrice(ioType:'None', type:'Boolean', required:false, defaultValue:false)
 PassedInEntities(ioType:'None', type:'DataObject', required:false,
 context:'PricingInternal.PriceRequestInternal')
}

// Step "Set Initial Values" - block
step('Set Initial Values'){
 // Step "Initialize" - SubAlgorithm
 step('Initialize'){
 subAlgorithm(name:'Set Initial Values')
 .input ('EnableCache':false,'PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Process and Split Passed In Charges" - if
step('Process and Split Passed In Charges', if:{PriceRequest.Charge.size() > 0}){
 // Step "Process Passed In Charges and Charge Components" - SubAlgorithm
 step('Process Passed In Charges and Charge Components'){
 subAlgorithm(name:'Process Passed In Charges')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
 // Step "Split Passed In Charge Components" - Split
 // Moves all the passed-in charge components that have been retained after processing to a separate SDO.
 step('Split Passed In Charge Components'){
 splitDocument().from(PriceRequest.ChargeComponent).as('Comp').to(PassedInEntities.ChargeComponent).where{true}
 }
 // Step "Split Passed In Charges" - Split
 // Moves all the passed-in charges that have been retained after processing to a separate SDO.
 step('Split Passed In Charges'){
 splitDocument().from(PriceRequest.Charge).as('Charge').to(PassedInEntities.Charge).where{true}
 }
}
```

```
// Step "Derive and Validate Strategy" - block
step('Derive and Validate Strategy'){
 // Step "Derive Pricing Strategy" - SubAlgorithm
 step('Derive Pricing Strategy'){
 subAlgorithm(name:'Get Sales Pricing Strategy')
 .input ('InvokeFromPriceSalesTransactions':true,'PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
 // Step "Validate Pricing Strategy" - SubAlgorithm
 step('Validate Pricing Strategy'){
 subAlgorithm(name:'Validate Pricing Strategy')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Validate Passed Pricing Terms" - block
step('Validate Passed Pricing Terms'){
 // Step "Validate Pricing Terms" - SubAlgorithm
 step('Validate Pricing Terms'){
 subAlgorithm(name:'Validate Pricing Terms')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Validate Override Currencies" - block
step('Validate Override Currencies'){
 // Step "Validate Override Currency" - SubAlgorithm
 // Invoke the sub-algorithm 'Validate Override Currencies'
 step('Validate Override Currency'){
 subAlgorithm(name:'Validate Override Currencies')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Get Charge Candidates For Coverages Goods and Services" - block
step('Get Charge Candidates For Coverages Goods and Services'){
 // Step "Get Price List and Charge Candidates" - SubAlgorithm
 step('Get Price List and Charge Candidates'){
 subAlgorithm(name:'Get Base List Price for Goods and Services')
 .input
 ('CreateChargesFlag':false,'DerivePriceListFlag':true,'ElementParam':'QP_BASE_LIST_PRICE','GetChargeCandidatesFlag'
 .output ('PerformCurrencyConversion':'PerformCurrencyConversion','PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Calculate Pricing Charges For Covered Items" - block
// This step invokes the algorithm to calculate pricing charges for covered items to derive the percentage-
based coverage prices.
step('Calculate Pricing Charges For Covered Items'){
 // Step "If CoverageAssociations Exist" - if
 // Calculate Covered Item Prices if there are CoverageAssociations
 step('If CoverageAssociations Exist', if:{PriceRequest.CoverageAssociation.size() > 0}){
 // Step "Calculate Covered Item Charges" - SubAlgorithm
 step('Calculate Covered Item Charges'){
 subAlgorithm(name:'Calculate Covered Item Charges')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
```

```
 .run()
 }
 }
}

// Step "Calculate Pricing Charges For Coverages Goods and Services" - block
// Call Calculate Pricing Charges where in Get Base
// skip Derive PL and Get Charge Candidates, only Create Charges, but process rest of the ListPrice thru
 Margin Calculations
step('Calculate Pricing Charges For Coverages Goods and Services'){
 // Step "Calculate Pricing Charges" - SubAlgorithm
 // This step invokes the algorithm to calculate pricing charges.
 step('Calculate Pricing Charges'){
 subAlgorithm(name:'Calculate Pricing Charges')
 .input
 ('CreateChargesFlag':true,'DerivePriceListFlag':false,'GetChargeCandidatesFlag':false,'PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Merge If Charges Passed In" - if
step('Merge If Charges Passed In', if:{PassedInEntities.Charge.size() > 0 && finer('Merging back passed-in
 charges and charge components') == null}){
 // Step "Merge Back Passed In Charges" - Merge
 // Merges back the passed in Charge Components into Price Request SDO
 step('Merge Back Passed In Charges'){
 mergeDocument().from(PassedInEntities.Charge).to(PriceRequest.Charge)
 }
 // Step "Merge Back Passed In Charge Components" - Merge
 // Merges back the passed in Charge Components into Price Request SDO
 step('Merge Back Passed In Charge Components'){
 mergeDocument().from(PassedInEntities.ChargeComponent).to(PriceRequest.ChargeComponent)
 }
}

// Step "Calculate Shipping Charges For Goods and Services" - block
step('Calculate Shipping Charges For Goods and Services'){
 // Step "Calculate Shipping Charges" - SubAlgorithm
 // Call 'Calculate Shipping Charges' Algorithm
 step('Calculate Shipping Charges'){
 subAlgorithm(name:'Calculate Shipping Charges')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Populate Charge Component Explanation Messages" - block
// This step populates explanation messages on charge components.
step('Populate Charge Component Explanation Messages'){
 // Step "Populate Explanation Messages" - SubAlgorithm
 // This step invokes the algorithm to populate explanation messages for charge components.
 step('Populate Explanation Messages'){
 subAlgorithm(name:'Populate Charge Component Explanation Message')
 .input ('BaseListPriceElementCode':'QP_BASE_LIST_PRICE','PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Process Returns" - block
step('Process Returns'){
 // Step "Process Returns With Reference" - SubAlgorithm
 // This step creates the refund charges for return lines.
 step('Process Returns With Reference'){
```

```
 subAlgorithm(name:'Process Returns with Reference')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}

// Step "Set Final Values" - block
step('Set Final Values'){
 // Step "Finalize" - SubAlgorithm
 step('Finalize'){
 subAlgorithm(name:'Set Final Values')
 .input ('PriceRequest':PriceRequest)
 .output ('PriceRequest':'PriceRequest')
 .run()
 }
}
```

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Overview of Oracle Pricing
- Manage Pricing Algorithm Steps
- Pricing Algorithms

Modify Your Pricing Algorithm's Variables

In this example, you modify variables to allow your users to set the value for the Calculation GSA price, to set the value for segment price list charges, and to validate that segment prices are greater than or equal to GSA prices.

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

Summary of the Steps

1. Examine the current behavior.
2. Create a new version of the algorithm.
3. Examine the pricing strategy and pricing strategy assignment.
4. Verify your set up.

Examine the Current Behavior

1. Make sure you have the privileges that you need to manage sales orders.
2. Go to the Order Management work area, then create a new sales order.
3. On the New Sales Order page, add a customer and add an item:

Attribute	Value
Customer	Computer Service and Rentals
Select Item	AS54888

4. In the Amount column, click the link, then notice that the Line dialog doesn't include pricing for GSA:

Price Component	Amount
List Price	2,500
Discount	100
Tax	200
Net Price Plus Tax	2,600

Create a New Version of the Algorithm

1. Sign out, then sign in with the privileges that you need to administer pricing.
2. Go to the Pricing Administration work area.
3. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
4. On the Manage Algorithms page, locate the algorithm you must modify.

   To locate the pricing algorithm you must modify, examine the links in the Name column, click one that looks promising, then examine the steps. It might be necessary to examine more than one algorithm.

   For this example, you must allow your users to modify the price for a sales transaction, so, select the row that includes Price Sales Transactions in the Name column.
5. Click **Actions > Create Version**.

   Pricing creates a new version and sets the Status to In Progress.
6. In the Name column of the version you just created, click **Price Sales Transactions**.
7. On the Edit Algorithm page, examine the step names to verify that the algorithm can implement the behavior your require.

   This algorithm includes step names that set the value for the GSA price, such as Calculate GSA Price, so its likely you can modify it to implement the behavior you need.
8. Click **Variables**.
9. In the row that contains CalculateGSAPrice in the Name column, set the value, then click **Save and Close**:

Attribute	Value
Default Expression	true

10. On the Manage Algorithms page, choose the row that contains the highest version of Price Sales Transaction, then click **Actions > Publish**.

Examine the Pricing Strategy and Pricing Strategy Assignment

1. Click **Tasks > Manage Pricing Strategy Assignment**.

2. Click the Strategy Assignment header row, then examine the line that includes GSA Corporate Segment in the Pricing Segment.

   Examine the setup for the customer pricing profile, pricing segment, and pricing strategy assignment. For example, notice how Pricing assigns GSA Corporate Strategy to the Corporate Segment pricing segment for customer Computer Service and Rentals.
3. Examine the price list and verify that it contains these values:

Attribute	Value
Item	AS5488
Pricing Charge Definition	Sale Price
Charge Type	One Time
Charge Sub Type	Price
Calculation Method	Price
Base Price	200

Verify Your Set Up

1. Sign out, then sign into Order Management.
2. Create a sales order, add a customer, then add an item.
3. In the Amount column, click the link, then make sure the Line dialog doesn't include an amount for tax:

Price Component	Amount
List Price	2,500
Discount	100
Net Price Plus Tax	2,400

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Pricing Algorithm Steps
- Overview of Oracle Pricing
- Manage Price Lists
- Manage Pricing Algorithms

Assign Pricing Operations to Pricing Algorithms

Manage a pricing process assignment.

The predefined Price Request service mapping uses the pricing process assignment to identify and run the pricing algorithm at runtime. Oracle Pricing runs the pricing algorithm when it requires the operation you specify. The pricing algorithm calculates the prices, adjustments, totals, or profit margins that the pricing operation requires. For example, you can specify to call the Price Sales Transactions pricing algorithm for the Price Request service.

Here are the predefined pricing operations that Pricing uses most of the time:

- Price Sales Transactions
- Calculate Sales Order Totals

It also uses the Get Sales Pricing Strategy pricing operation. Pricing doesn't currently use the Validate Sales Price pricing operation. The predefined Price Request service mapping includes these operations.

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

Manage a pricing process assignment:

1. Go to the Setup and Maintenance work area, then go to the Manage Pricing Process Assignments task.
2. On the Manage Pricing Process Assignments page, examine the pricing process assignments that come predefined with Pricing.

   To reduce maintenance, use a predefined pricing process assignment instead of creating a new one.
3. If you can't locate a pricing process assignment that meets your requirements, then click **Actions > Add Row**, then set the values:

Attribute	Value
Pricing Operation	Choose the pricing operation.
Process Name	Choose the pricing algorithm that Pricing must run to meet the needs of your pricing operation.

4. Click **Save**.

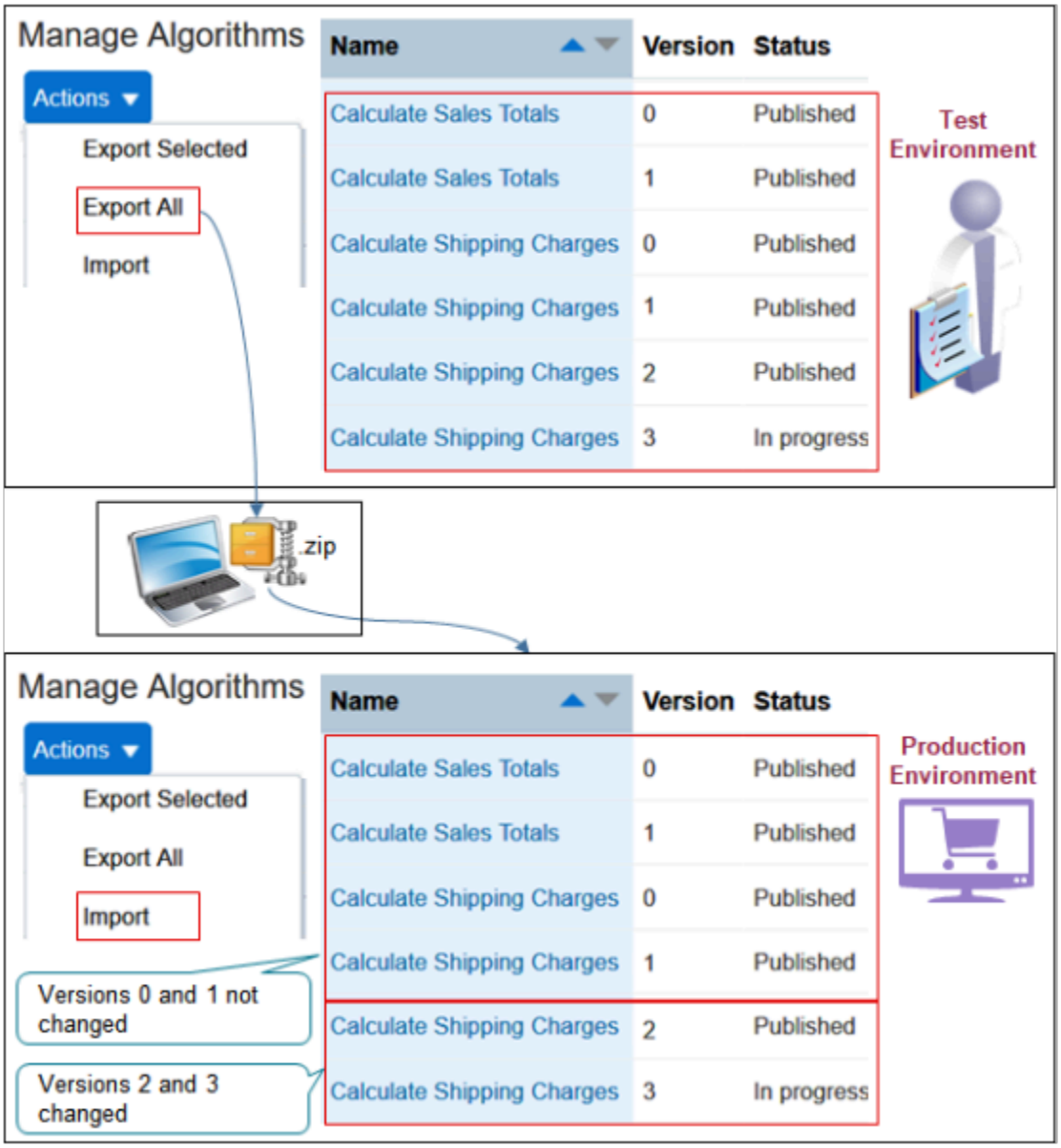
Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Pricing Algorithms

Migrate Pricing Algorithms from Test to Production

You can extend a pricing algorithm to meet your specific business requirements, then use export and import to move it between environments.

You typically develop your algorithms in a test environment. You finish testing then use the Export and Import actions to move it to your production environment. You can also use export to backup your extended algorithms:



Export Your Algorithms

Assume you develop and test algorithms in an environment named Test. You finish testing, then need to export all your algorithms, including predefined ones, into an environment named Production:

1. Sign into your Test environment with the privileges that you need to administer pricing.
2. In the Pricing Administration work area, click **Tasks > Manage Algorithms** under Pricing Configuration.
3. On the Manage Algorithms page, click **Actions > Export All**:

Action	Description
Export Selected	◦ This action exports only the algorithms you select. ◦ It exports all versions of each algorithm you select. For example, if the Calculate Shipping Charges algorithm has versions 0, 1, 2, 3, 4, 5, 6, and 7, and if you select only version 3 of Calculate Shipping Charges, then the export exports all versions of Calculate Shipping Charges. ◦ You can't export only one version or only some versions of an algorithm.
Export All	Exports all versions of all algorithms.
Import	◦ Imports one or more algorithms from the zip file you create when you use the Export Selected or Selected All action. ◦ You can't import only some versions of an algorithm. ◦ If the zip file contains more than one algorithm, you must import all of them. You can't pick and select.

4. In the dialog that displays, select the Save File option, click **OK > Save**:
    - You can use the predefined ExportedAllAlgorithms.zip file name, or change it, but you must save the file as a zip file type.
    - Make a note of the location where you save the file. For this example, assume you created a folder named my_algorithms, and you save the zip file on your local computer at `c:\my_algorithms`.
    - You can use this file as a backup. For example, if your server suffers a catastrophic failure, then you can import the file into a new environment, and pick up where you left off.

Import Your Algorithms

1. Sign out of your Test environment, then sign into your Production environment.
2. Navigate to the Manage Algorithms page, then click **Actions > Import All**.
3. In the dialog that displays, navigate to `c:\my_algorithms`, locate the ExportedAllAlgorithms.zip file you exported earlier, select it, then click **OK**:
    - The import replaces algorithms that currently exist in Production with the algorithms that you exported from Test.
    - If the zip file that you import from Test contains more than one version of an algorithm, then the import removes all versions of the algorithm in Production except for versions 0 and 1, and replaces them with the versions from Test. The import doesn't mess with Versions 0 and 1 in Production. They remain the same.

Assume you import Algorithm_1, Algorithm_2, and Algorithm_3_Custom. 1 and 2 are predefined. 3 is the one you created:

Algorithm Name	Versions in Test	Versions in Production Before You Import	Versions in Production After You Import
Algorithm_1	Version 0, 1, 2, 3 (inactive), 5	Version 0, 1, 2	Version 0, 1, 2, 3 (inactive), 5
Algorithm_2	Version 0, 1, 6, 7 (inactive), 9, 10, 11, 12	Version 0, 1, 2, 3, 6	Version 0, 1, 6, 7 (inactive), 9, 10, 11, 12

Algorithm Name	Versions in Test	Versions in Production Before You Import	Versions in Production After You Import
Algorithm_3_Custom	Version 1, 2 The export exports all versions above version 0 for each algorithm that you create.	Version 1, 2, 3, 4	Version 1, 2

Learn how versions work. For details, see *Promote Pricing Algorithms Into the Latest Update* and *Manage Modifications Through Updates*.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Manage Pricing Algorithm Steps
- Pricing Algorithms

10 Use Cases

Overview of Pricing Use Cases

Use the pricing use cases to help you implement your more advanced pricing set up requirements.

The use cases in this chapter provide stepwise details that walk you through each set up, such as assign a pricing strategy according to order type, create discounts that accumulate or cascade, and so on.

If you don't find the use case you need in this chapter, see *Technical Reference for Oracle Pricing (KB171412)*. Here are some of the use cases that the technical reference contains:

- Convert Units of Measure
- Specify Currency Precision
- Apply Discounts According to Customer Class
- Apply Flat Rate Shipping Charges
- Apply a Percent Discount On Order Header

Strategies

Assign Pricing Strategy According to Order Type

Set up Oracle Pricing so it assigns the pricing strategy according to the type of sales order.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Assume you must assign a different pricing strategy depending on how quickly you deliver the item to your customer. You set up pricing so it assigns pricing strategies according to order type:

Order Type	Pricing Strategy
Standard	Corporate Pricing Strategy
Expedite	Expedite Delivery Pricing Strategy

This example assumes you already defined these pricing strategies.

Summary of the Set Up

1. Set up the order types.
2. Modify the service mapping.
3. Modify the matrix class.
4. Assign the pricing strategy.
5. Test your set up.

Set Up the Order Types

1. Sign into Order Management with administrative privileges.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Parameters**.
3. Go to the Setup and Maintenance work area, then go to the task.
4. On the Manage Order Lookups page, in the Search area, enter the value, then click **Search**:

Attribute	Value
Lookup Code	ORA_ DOO_ ORDER_TYPES

5. In the Lookup Codes area, click **Actions > New** to add each lookup code:

Lookup Code	Meaning
Standard Sales Orders	Standard sales order
Expedited Sales Orders	Sales order that must deliver as soon as possible

   For details, see *Use Order Management to Fulfill Different Types of Sales Orders*.
6. Sign out of Order Management.

Modify the Service Mapping

Modify the Sales service mapping so it can examine the type of sales order. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*:

1. Sign into Oracle Pricing with administrative privileges.
2. Go to **Home > Order Management > Pricing Administration > Manage Service Mappings** under Pricing Configuration.
3. On the Manage Service Mappings page, click **Sales**.
4. On the Edit Service Mappings page, in the Entities area, click **Query by Example**, then query for the Header entity.
5. In the Details area, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	OrderTypeCode_Custom

Attribute	Value
Type	String
Primary Key	Doesn't contain a check mark.
Alternate Key	Doesn't contain a check mark.
Allow Null	Contains a check mark.

You add this attribute so the Pricing Administration work area can examine the value of the order type.

6. Add your new attribute to the service that gets order header details:
    - Click **Services > Query by Example**, and then query for the PriceRequestHeader service.
    - In the Details area, on the Entities tab, click the **row** that includes Header in the Entity column.
    - In the Entities area, click **Actions > Add Row**, then set the values:

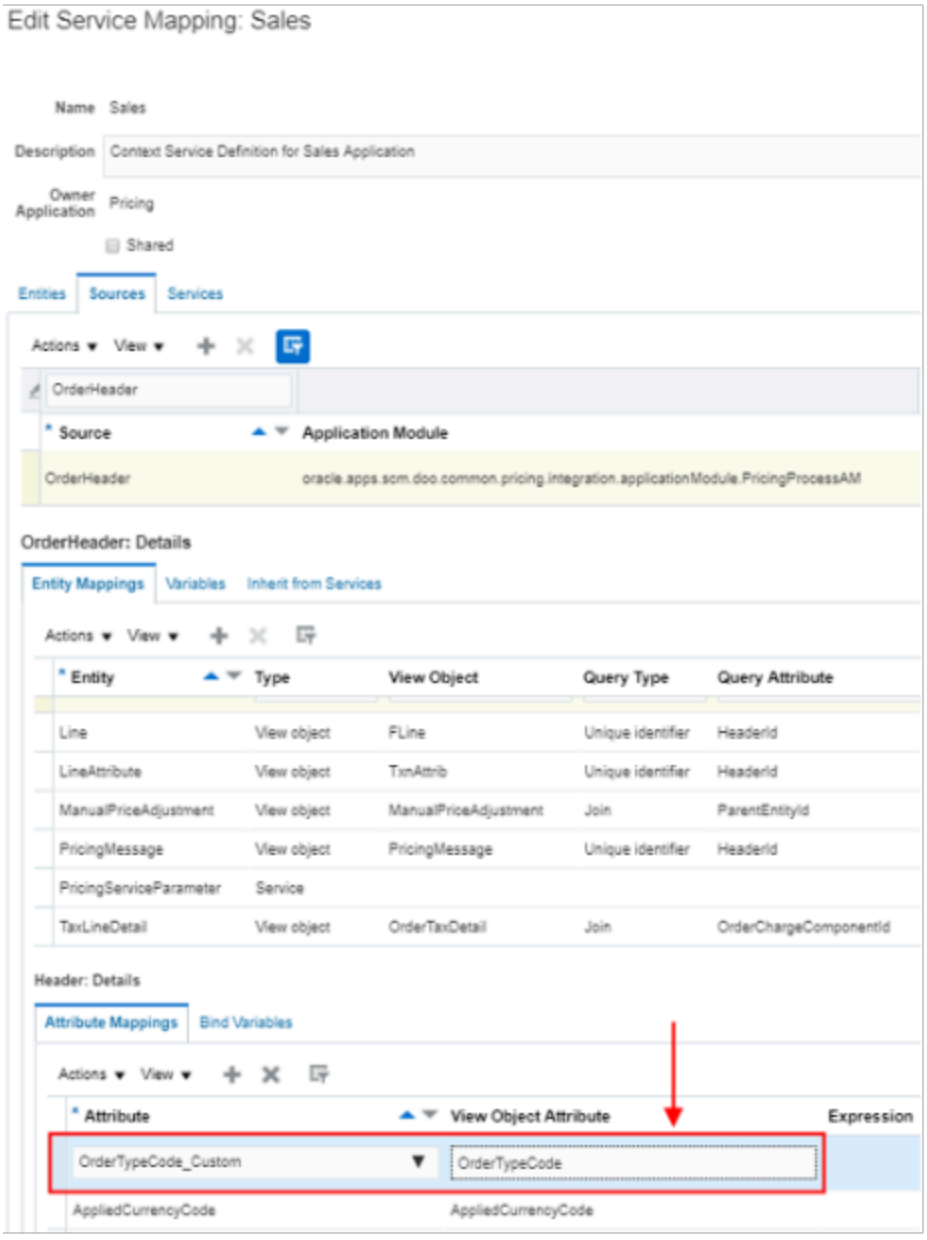
Attribute	Value
Attribute	OrderTypeCode_Custom
Read	Contains a check mark.
Write	Does not contain a check mark.

7. Add your new attribute to the order header source:
    - Click **Sources > Query by Example**, then query for the OrderHeader source.
    - In the Details area, on the Entity Mapping tab, click the **row** that includes Header in the Entity column.
    - On the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	OrderTypeCode_Custom
View Object Attribute	OrderTypeCode The OrderTypeCode is a predefined view object that Order Management uses to communicate the value of the order type. This view object gets values from the ORA_ DOO_ ORDER_TYPES lookup.

Attribute	Value
	The Order Entry Specialist uses the Order Type attribute on the order header in the Order Management work area to set the value for this lookup.

For example:



8. Click **Save and Close**.

Modify the Matrix Class

Modify the matrix class that assigns the pricing strategy so it can reference your new attribute:

1. Click **Tasks > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click **Sales Pricing Strategy Assignment**.
3. On the Edit Matrix Classes page, in the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Order Type Code
Source Code Name	OrderTypeCode
Comparison	=
Compare to Attribute	Header.OrderTypeCode_Custom OrderTypeCode_Custom is the attribute you added to the header entity on the service mapping earlier in this topic.
Allow Null	Contains a check mark.
Null Is Wildcard	Contains a check mark.

4. In the Domain Column, click **Edit Domain**, set the values, then click **OK > Save and Close**:

Attribute	Value
Name	Order Type Code
Domain Type	Lookup
Lookup	ORA_DOO_ORDER_TYPES
Default Value	Leave empty.

Assign the Pricing Strategy

Assign the pricing strategy according to the value of the order type:

1. Click **Tasks > Manage Pricing Strategy Assignments**.

2. On the Manage Pricing Strategy Assignments page, click the row that contains these values:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	All

3. In the Assignment Rules area, click **Edit Rules Table Columns**, add a check mark to Order Type Code, then click **OK**.

   You created the Order Type Code attribute earlier in this topic when you modified the matrix class.
4. In the Assignment Rules area, click **Actions > Add Row**, then set the values:

Attribute	Value
Channel Method	Leave empty.
Transaction Type	Sales Order
Pricing Segment	Corporate Pricing Segment
Order Type Code	Standard Sales Orders
Pricing Strategy	Corporate Pricing Strategy

5. Click **Actions > Add Row**, set the values, then click **Save and Close**:

Attribute	Value
Channel Method	Leave empty.
Transaction Type	Sales Order
Pricing Segment	Corporate Pricing Segment
Order Type Code	Expedited Sales Orders

Attribute	Value
Pricing Strategy	Expedite Delivery Pricing Strategy

Test Your Set Up

1. Sign out of Pricing.
2. Sign into Order Management with the privileges that you need to manage sales orders.
   You must sign out, then sign back in so Pricing can apply your set up.
3. Go to the Order Management work area, then click **Create Order**.
4. Set the Order Type to Standard Sales Orders, click **Additional Order Details > Pricing and Currency**, then verify that the pricing strategy and segment displays Corporate Pricing Strategy.
5. Set the Order Type to Expedited Sales Orders, then verify that the pricing strategy and segment displays Expedite Delivery Pricing Strategy.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Assign Pricing Strategy
- Manage Price Lists
- Pricing Guidelines

Assign Pricing Strategy According to Precedence

Set up a pricing strategy that references different pricing segments according to pricing precedence, but still use a default pricing strategy, such as Corporate Pricing Strategy.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Here's the set up you do so pricing assigns the pricing strategy according precedence:

Pricing Segment	Pricing Strategy	Pricing Precedence
Corporate Pricing Segment	Corporate Pricing Strategy	10
International Pricing Segment	International Pricing Strategy	20
Domestic Pricing Segment	Domestic Pricing Strategy	30

Assume you have already done this up:

- Set up a pricing strategy assignment for the Corporate Pricing Strategy for all of your customers.
- Created pricing segments:
    - Created Corporate Pricing Segment
    - Created Domestic Pricing Segment and assigned it to Computer Service and Rentals
    - Created International Pricing Segment and assigned it to Computer Associates International

  For details, see *Manage Pricing Segments*.
- Created pricing strategies:
    - Corporate Pricing Strategy
    - International Pricing Strategy
    - Domestic Pricing Strategy

  For details, see *Manage Pricing Strategies*.

Summary of the Set Up

1. Modify the matrix class.
2. Assign the pricing strategy.
3. Modify the pricing algorithm.
4. Test your set up.

Modify the Matrix Class

You modify the matrix class that assigns each pricing strategy. You add a result dimension that you can use to determine the precedence that Pricing uses when it assigns each strategy.

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click **Sales Pricing Strategy Assignment**.
3. On the Edit Matrix Class page, in the Result Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Precedence
Source Code Name	Precedence
Allow Null	Contains a check mark.

4. In the Domain column, click **Edit Domain**, set the values, then click **OK > Save and Close**:

Attribute	Value
Name	Precedence

Attribute	Value
Domain Type	None
Data Type	Number
Default Value	10
Default Value Is Fixed	Does not contain a check mark.

Assign the Pricing Strategy

Define an assignment matrix in a pricing strategy assignment. You set up the assignment matrix so it specifies the precedence to use when assigning the pricing strategy.

1. Click **Tasks > Manage Pricing Strategy Assignments**.
2. On the Manage Pricing Strategy Assignments page, immediately under the page title, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	Sales Order

   As an alternative, if you already defined a strategy assignment, then open it for editing, click **Actions > Edit Rules Table Columns**, then add your rules.
3. Click **Create Assignment Matrix**.
4. In the Create Assignment Matrix dialog, add a check mark to **Pricing Segment**, click **Next**, add a check mark to **Precedence**, then click **Finish**.
5. In the Pricing Strategy Assignment Rules area, add rules, then click **Save**:

Pricing Segment	Pricing Strategy	Pricing Precedence
Corporate Pricing Segment	Corporate Pricing Strategy	10
International Pricing Segment	International Pricing Strategy	20

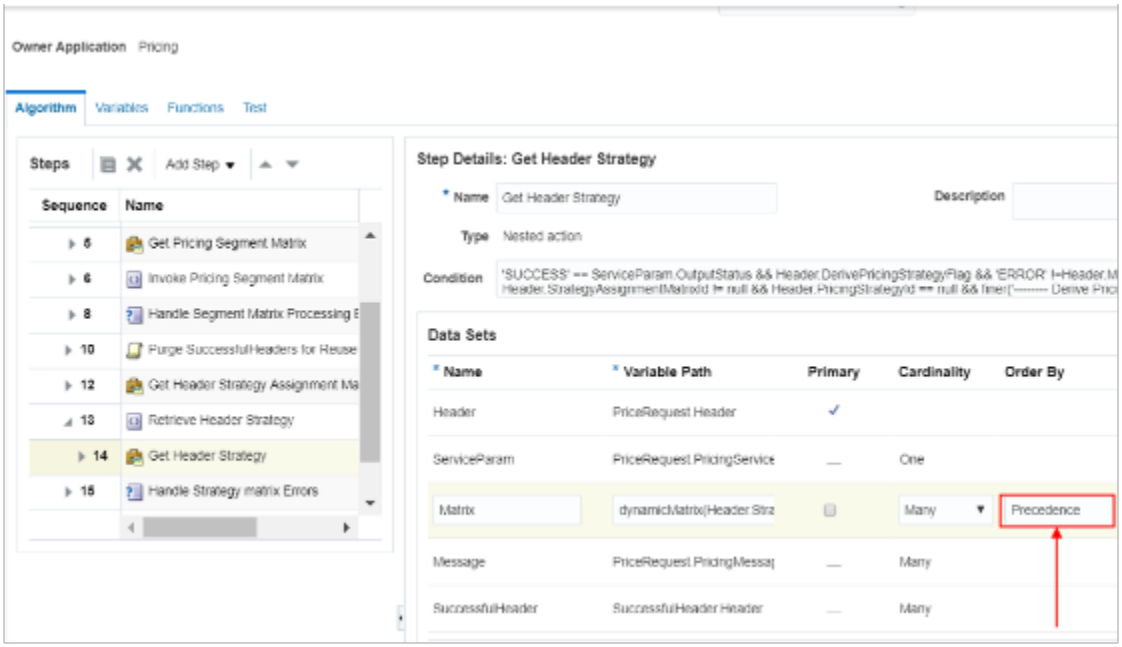
Pricing Segment	Pricing Strategy	Pricing Precedence
Domestic Pricing Segment	Domestic Pricing Strategy	30

Pricing gives the highest precedence to the rule with the lowest value in the Precedence attribute.

Modify the Pricing Algorithm

Modify the pricing algorithm that Pricing uses to determine the pricing segment and pricing strategy it references when it calculates the price for each sales order, sales agreement, quote, and so on.

Here's the modification you make so the algorithm sequences the assignments that Pricing gets from the assignment matrix according to precedence:



Modify the pricing algorithm:

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, query for the Get Sales Pricing Strategy pricing algorithm.
3. Click the **row** that includes the highest version of Get Sales Pricing Strategy, then click **Actions > Create Version**.
4. Click the row that includes the version you just created.
5. On the Edit Algorithm page, set the value:

Attribute	Value
Description	This Process is used to get the Pricing Segment and Pricing Strategy for the Sales Documents (Quotes, Sales Agreements, Orders). Extended on March 1, 2019, in update 19A to determine the pricing segment and pricing strategy to use when calculating price according to precedence.

Attribute	Value

To help manage your extended algorithm, particularly when you update, it's important to document it. Describe what your extended algorithm does. Include the date that you create the algorithm and the update.For background, see *Performing Your Quarterly Update (Doc ID 2337485.1)*.

6. On the Algorithm tab, in the Steps area, click the row that includes the value:

Attribute	Value
Name	Get Header Strategy

It might be necessary to expand steps so you can locate the Get Header Strategy step.

7. In the Data Sets area, click the **row** that includes the value:

Attribute	Value
Name	Matrix

8. Enter the value, then click **Save and Close > Done**:

Attribute	Value
Order By	Precedence

Test Your Set Up

1. Sign out of Pricing.
2. Sign into Order Management with the privileges that you need to manage sales orders.

You must sign out, then sign back in so Pricing can apply your set up.

3. Go to the Order Management work area, then click **Create Order**.
4. Don't set the customer. Instead, click **Additional Order Details > Pricing and Currency**, then verify that the pricing strategy and segment displays Corporate Pricing Strategy.

Recall that you applied Corporate Pricing Strategy to All customers. If you don't specify a value in the Customer attribute, then Pricing uses Corporate Pricing Strategy, by default.

5. Set the customer to Computer Service and Rentals, then verify that the pricing strategy and segment displays Domestic Pricing Segment.
6. Set the customer to Computer Associates International, then verify that the pricing strategy and segment displays International Pricing Segment.

Related Topics

- Pricing Algorithms
- Manage Pricing Algorithms
- Assign Pricing Operations to Pricing Algorithms
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Service Mapping

Assign Pricing Strategy According to Business Unit

Create a strategy assignment that assigns your pricing strategy according to the pricing segment that you assign to your customer and the business unit on the order header.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Assume you already set up profiles, segments and strategies for two of your customers:

Customer	Segment	Strategy
Computer Services	Tier 1	Commercial
Computers Direct	Tier 2	Corporate

For details, see *How Profiles, Segments, and Strategies Work Together*.

But you want to assign a different strategy for each customer according to business unit:

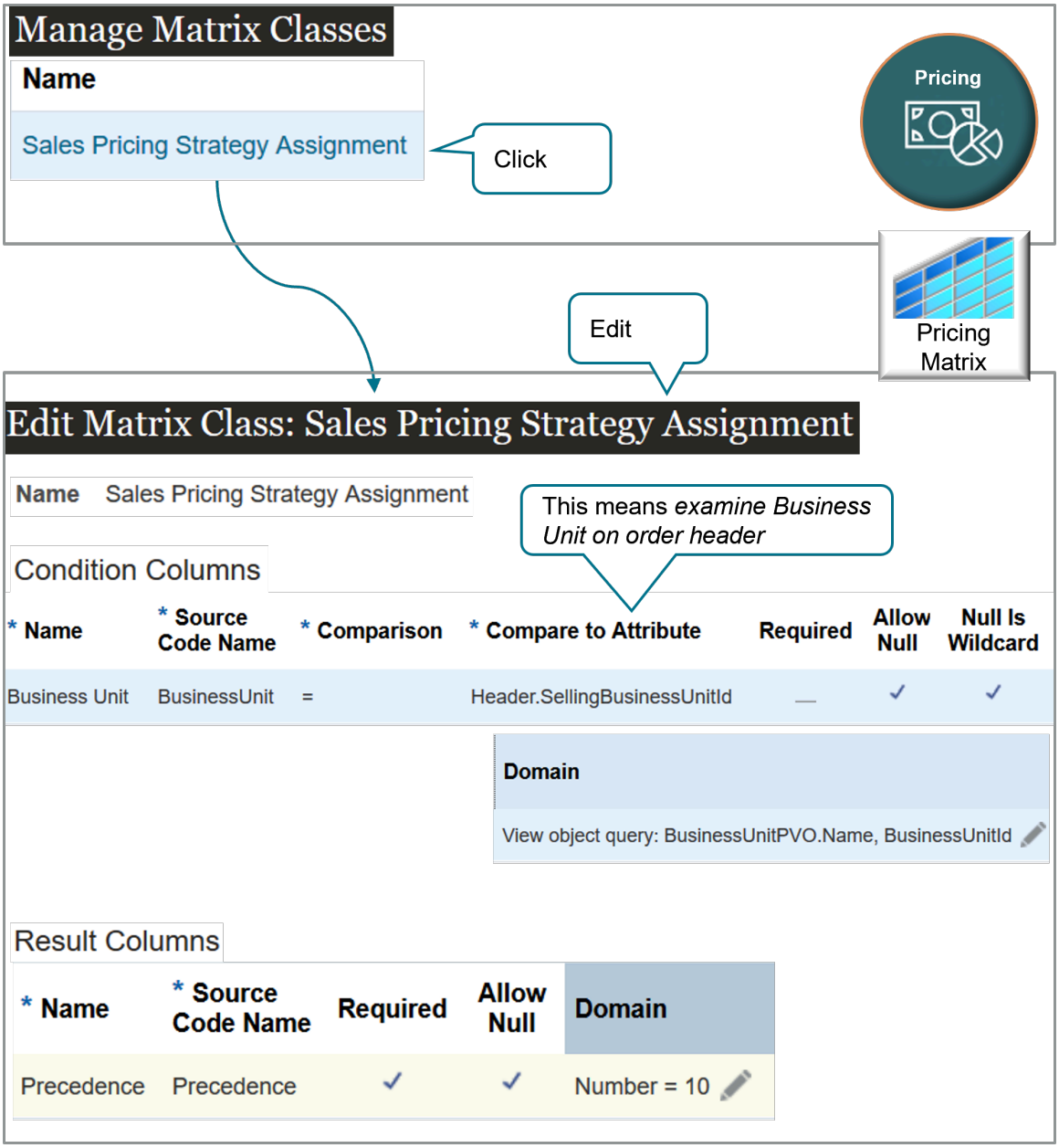
- If the segment is Tier 1 and the business unit is Vision Operations, then assign the Commercial pricing strategy.
- If the segment is Tier 1 and the business unit is Vision Services, then assign the Commercial pricing strategy.
- If the segment is Tier 2 and the business unit is Vision Operations, then assign the Corporate pricing strategy.

Summary of the Set Up

1. Modify the matrix class.
2. Modify the strategy assignment.
3. Modify the pricing algorithm.
4. Test your set up.

Modify Matrix Class

Modify the matrix class so you can set the business unit in the strategy assignment:



Try it:

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, in the Name column, click **Sales Pricing Strategy Assignment**.
3. On the Edit Matrix Class page, in the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Business Unit
Source Code Name	BusinessUnit

Attribute	Value
Comparison	=
Compare to Attribute	Header.SellingBusinessUnitId
Required	Doesn't contain a check mark.
Allow Null	Contains a check mark.
Null is Wildcard	Contains a check mark.
Domain	Click the pencil, then set the values.

To set the Domain, click the **pencil** in the Domain column, then set the values. Set them in the same sequence that this table displays them:

Sequence	Attribute	Value
1	Domain Type	View Object Query
2	Application Module	MatrixDomainAM (oracle.apps.scm.pricing.common.publicModel.applicat
3	Configuration	MatrixDomainAMLocal
4	View Object	BusinessUnitPVO
5	Key Attribute	BusinessUnitId
6	Display Attribute	Name
7	Data Type	Number

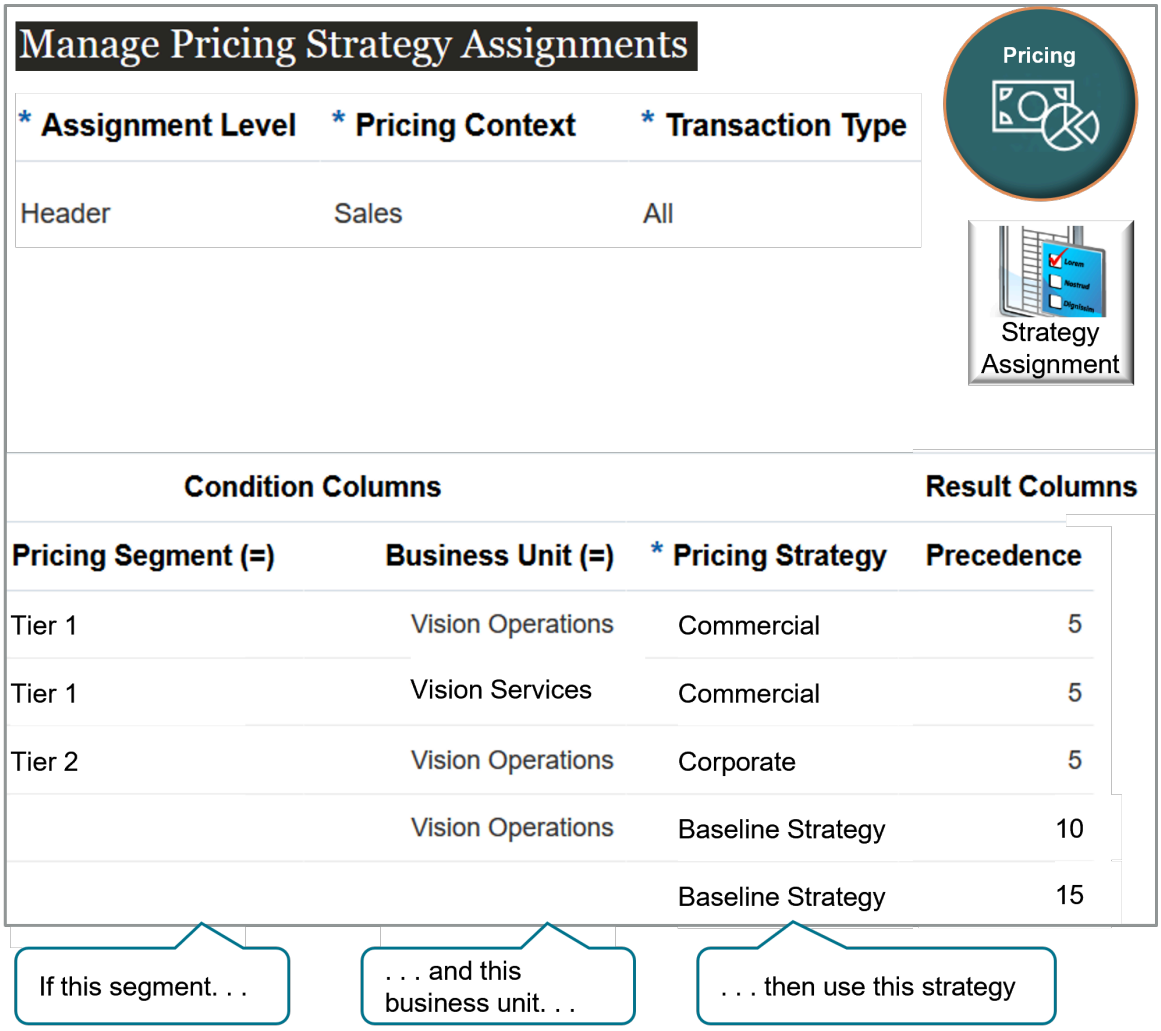
Leave all other attributes empty. Don't add a bind variable.

4. In the Result Columns area, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Name	Precedence
Source Code Name	Precedence
Required	Contains a check mark.
Allow Null	Contains a check mark.
Domain	Number = 10 Set up precedence as a number. Don't set it up as text. Assume you have the values 100, 70, 10. If you set it up as: o **Text** Pricing will sort them as 10, 100, 70, which will fail. o **Numeric** Pricing will sort them in ascending order, 10, 70, 100, which is expected.

Modify Strategy Assignment

Next, modify the strategy assignment:



Try it:

1. Click **Tasks > Manage Pricing Strategy Assignments**.
2. On the Manage Pricing Strategy Assignments page, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	All

3. Click **Create Assignment Matrix**.

4. In the Create Assignment Matrix dialog, in the Select Optional Condition Columns area, set the values, then click **OK**:

Attribute	Value
Pricing Segment	Contains a check mark.
Business Unit	Contains a check mark.

5. In the Pricing Strategy Assignment Rules area, create 5 rows:

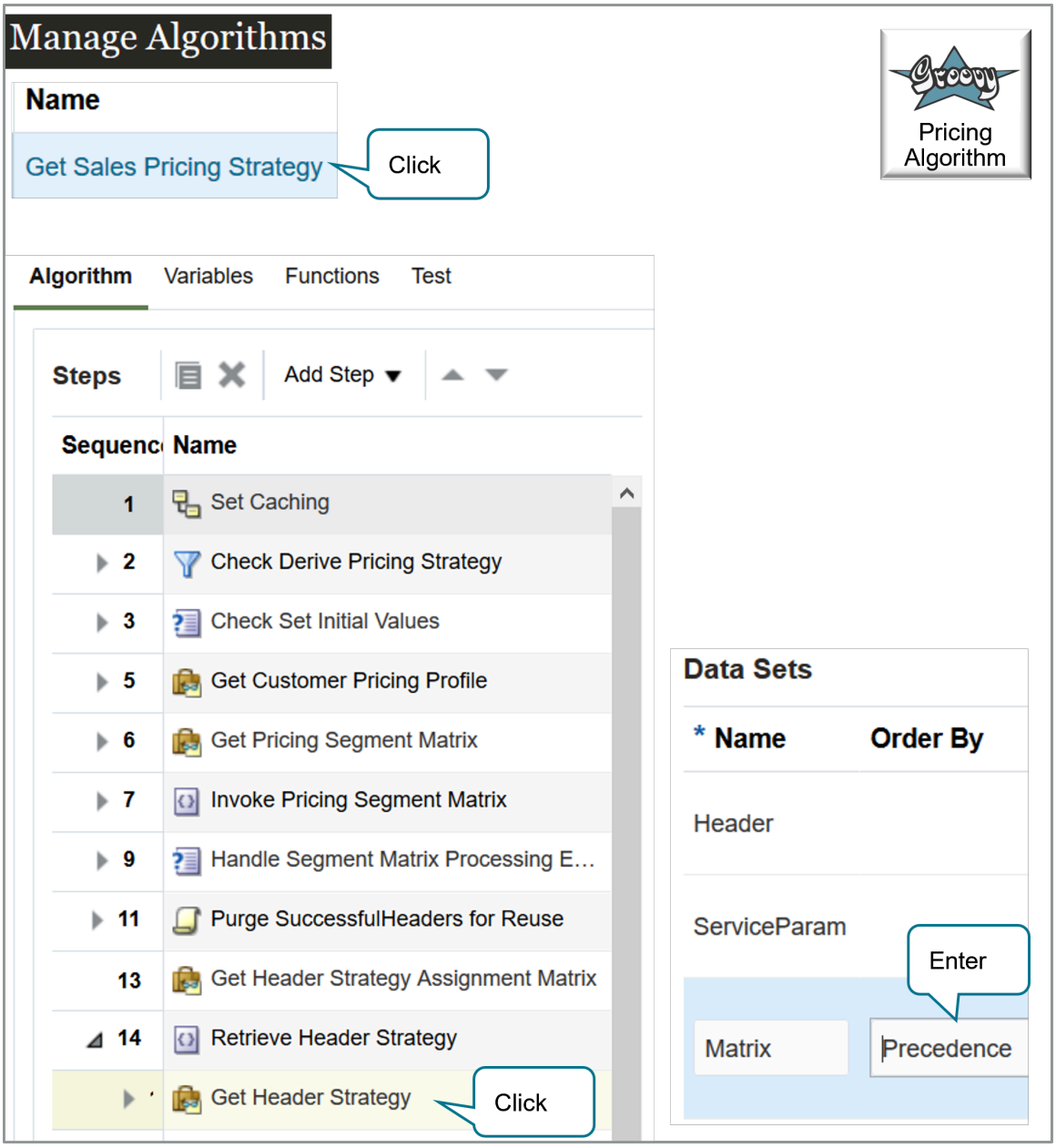
Pricing Segment	Business Unit	Pricing Strategy	Precedence
Tier 1	Vision Operations	Commercial	5
Tier 1	Vision Services	Commercial	5
Tier 2	Vision Operations	Corporate	5
-	Vision Operations	Baseline Strategy	10
-	-	Baseline Strategy	15

Note

- Pricing uses the view object that you set up earlier in the matrix class to populate the list of values that you use to select the business unit. The list contains all the business units that you have set up in you organization.
- The fourth row captures the condition where you haven't set up a pricing segment and you set the business unit on the order header to Vision Operations.
- The fifth row captures the condition where you haven't set up a pricing segment and you don't set the business unit on the order header to any value.
- Baseline Strategy can be any strategy that you use when no conditions apply. This way, Pricing can still price the item and avoid an error condition.

Modify Pricing Algorithm

Modify the pricing algorithm so it examines the rules you just set up in the assignment matrix according to the pricing precedence you set in the matrix:



Try it:

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, filter the **Name** column for Get Sales Pricing Strategy.
3. Click the **row** that has the highest published version of Get Sales Pricing Strategy, then click **Actions > Create Version**.
4. Click the **link** in the Name column of the row that has the highest, In Progress version.
5. On the Edit Algorithm page, on the Algorithm tab, expand the Retrieve Header Strategy step, then, in the Name column, click **Get Header Strategy**.

6. In the Data Sets area, in the row that contains Matrix in the Name column, set the value:

Attribute	Value
Order By	Precedence

7. Click **Save and Close**.
8. On the Manage Algorithms page, click **Actions > Publish**.

Test Your Set Up

1. Open another browser window and sign into Order Management with the privileges that you need to manage sales orders.
2. Go to the Order Management work area, create a sales order, then set the values:

Attribute	Value
Customer	Computer Services
Business Unit	Vision Operations

3. Click **Additional Order Details > Pricing and Currency**, then verify the values:

Attribute	Value
Pricing Segment	Tier 1
Pricing Strategy	Commercial

4. Reset values:

Attribute	Value
Business Unit	Vision Services

5. Click **Additional Order Details > Pricing and Currency**, then verify the values:

Attribute	Value
Pricing Segment	Tier 1
Pricing Strategy	Commercial

Attribute	Value

6. Reset values:

Attribute	Value
Customer	Computers Direct
Business Unit	Vision Operations

7. Click **Additional Order Details > Pricing and Currency**, then verify the values:

Attribute	Value
Pricing Segment	Tier 2
Pricing Strategy	Corporate

Related Topics

- How Profiles, Segments, and Strategies Work Together

Assign Pricing Strategy According to Customer

Assign the pricing strategy according to the value that you set in the Customer attribute on each sales order.

You can assign a different strategy for each customer according to the customer number or the customer name.

Assume you want to assign a different strategy for each customer according to the customer number:

- If the segment is Tier 1 and the customer is Computer Service and Rentals, then assign the Commercial pricing strategy.
- If the segment is Tier 1 and the customer is ABC Telecommunications, then assign the Corporate pricing strategy.

CAUTION: The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Assigning the pricing strategy according to customer is similar to assigning according to business unit. Do the set up described in the Assign Pricing Strategy According to Business Unit topic, but with these following important differences:

1. Get the customer number. Skip this step if you need to assign according to customer name.
2. Modify the matrix class.
3. Modify the strategy assignment.
4. Modify the service mapping.

Get the Customer Number

You will reference the customer number instead of the customer name because the number is unique. Using the number helps to avoid problems when you have the same customer name in more than one customer record.

Use the Manage Customers task to get the customer numbers. Assume you identify the numbers:

Customer	Registry ID
Computer Service and Rentals	2145
ABC Telecommunications	2150

For details, see *Display Account Details on Sales Orders*.

Modify the Matrix Class

Modify the matrix class so you can set the customer in the strategy assignment:

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, in the Name column, click **Sales Pricing Strategy Assignment**.
3. On the Edit Matrix Class page, in the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	Customer
Source Code Name	Customer
Comparison	=
Compare to Attribute	Header.CustomerId
Required	Doesn't contain a check mark.
Allow Null	Contains a check mark.

Attribute	Value
Null is Wildcard	Contains a check mark.
Domain	Click the pencil, then set the values.

To set the Domain, click the **pencil** in the Domain column, then set the values in the Edit Column Domain Values dialog. Set them in the same sequence that this table displays them:

Sequence	Attribute	Value
1	Domain Type	View Object Query
2	Application Module	MatrixDomainAM (oracle.apps.scm.pricing.common.publicModel.applicat
3	Configuration	MatrixDomainAMLocal
4	View Object	CustomerPVO
5	Key Attribute	PartyId For details, see *Overview of Displaying Customer Details on Sales Orders*.
6	Display Attribute	PartyNumber If you want to use the customer name, set this attribute to PartyName instead of PartyNumber.
7	View Criteria	CustomerPVOPartyTypeAndStatusCriteria Also, add these bind variables: o Add the bindPartyStatus bind variable and set it's value to A. o Add the bindPartyType bind variable and set it's value to ORGANIZATION.
8	Data Type	Number

Sequence	Attribute	Value

Leave all other attributes empty.

Modify the Strategy Assignment

Next, modify the strategy assignment:

1. Click **Tasks > Manage Pricing Strategy Assignments**.
2. On the Manage Pricing Strategy Assignments page, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Assignment Level	Header
Pricing Context	Sales
Transaction Type	All

3. Click **Create Assignment Matrix**.
4. In the Create Assignment Matrix dialog, in the Select Optional Condition Columns area, set the values, then click **OK**:

Attribute	Value
Pricing Segment	Contains a check mark.
Customer	Contains a check mark.

5. In the Pricing Strategy Assignment Rules area, create 2 rows:

Pricing Segment	Customer	Pricing Strategy	Precedence
Tier 1	2145	Commercial	5
Tier 1	2150	Corporate	5

6. Continue with the rest of the work described in the Assign Pricing Strategy According to Business Unit topic, such as modifying the pricing algorithm and testing your set up.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Assign Pricing Strategy According to Business Unit
- Overview of Displaying Customer Details on Sales Orders

Lists

Create Discounts That Accumulate or Cascade

Apply one or more discounts that accumulate on a single price list.

You can also apply one or more discounts that cascade. For example, on the running net price.

You apply discounts through a simple rule or through a pricing matrix according to different types of adjustments:

- Discount Amount
- Discount Percent
- Markup Amount
- Markup Percent
- Price Override

Here are some important concepts:

- **Cascading discount**. A calculation that subtracts the discount that each discount rule applies on an item. Here's what cascading discount does:
    - Subtracts discounts in alphabetic, ascending sequence according to rule name.
    - Uses Running Net Price as the basis when it applies each adjustment.
    - Reduces the value of Running Net Price each time it applies the discount for each rule.
    - Doesn't keep the basis constant for each rule. Instead, the discount cascades from rule to rule.
- **Cumulative discount**. A calculation that subtracts the discount that each discount rule applies on an item. Here's what cumulative discount does:
    - Subtracts discounts in alphabetic, ascending sequence according to rule name.
    - Accumulates the total discount amount while it applies each rule.
    - Uses List Price as the basis when it applies each adjustment.
    - Doesn't modify the value of List Price when it applies each discount.
    - Keeps the basis constant for each rule. The discount remains the same from rule to rule.
- **Running net price**. An object that can go up or down in value while Pricing applies discounts and does other calculations on the price of an item.

You can't create a discount that accumulates or cascades on a configured item.

Summary of the Set Up

1. Create price element and pricing basis.
2. Modify pricing algorithms.
3. Modify pricing algorithms for discounts.
4. Test.

CAUTION: The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

1. Create Price Element and Pricing Basis

Create the price element and pricing basis that you will use for the running net price:

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Price Elements**.
2. On the Manage Price Elements page, click **Actions > Add Row**, set values, then click **Save and Close**:

Attribute	Value
Element Code	ELEMENT_FOR_NET_PRICE
Element Name	Running Net Price
Type	Price
Active	Contains a check mark.

3. On the Search page, search for, then open Manage Pricing Bases.
4. On the Manage Pricing Bases page, click **Actions > Create**, set the values, then click **Save and Close > Done > Done**:

Attribute	Value
Name	Running Net Price
Usage	Adjustment Basis
Price Element	Running Net Price
Description	Pricing basis for cumulative and cascading discounts on running net price.
Active	Contains a check mark.

Create Service Mapping Attributes

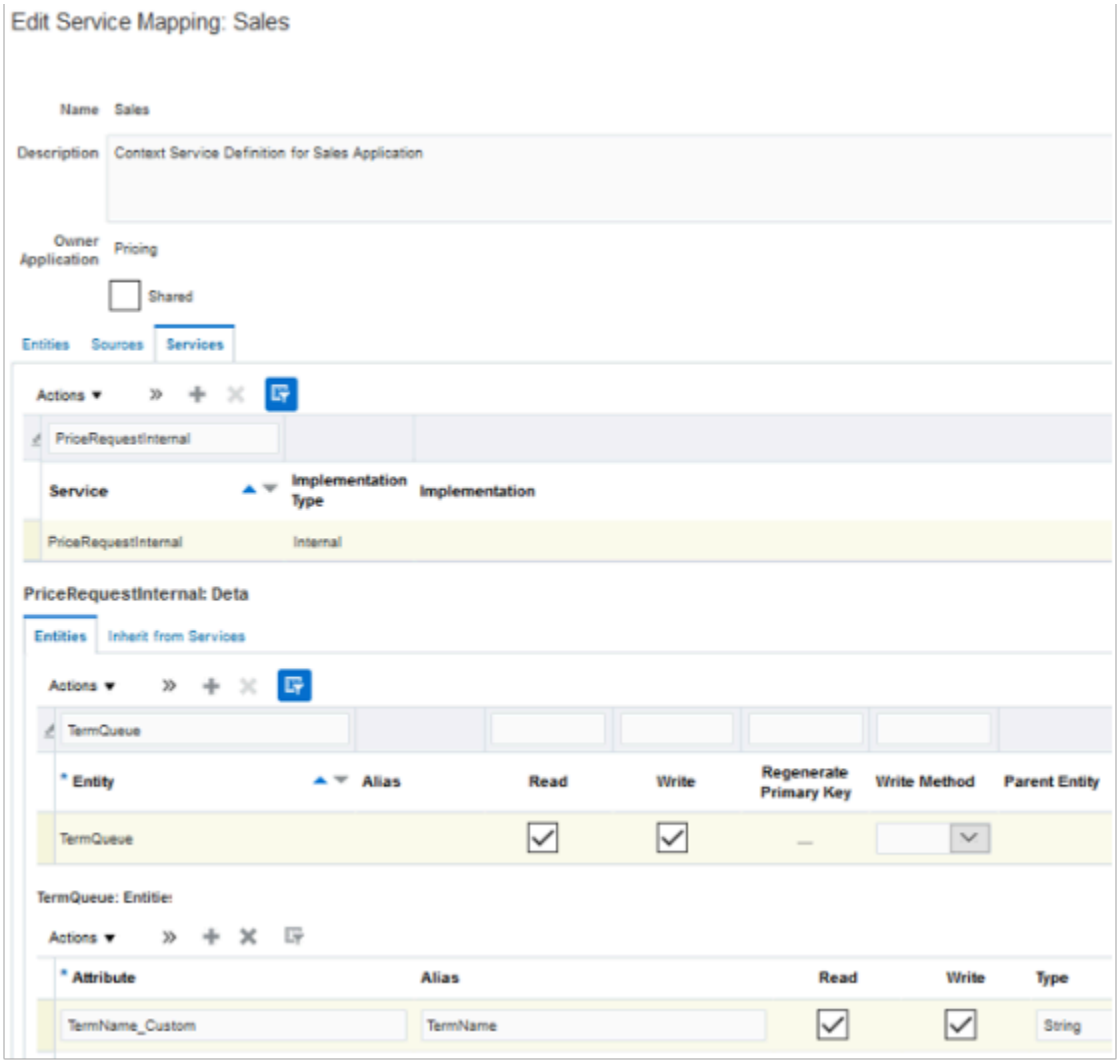
You modify the Sales service mapping differently, depending on whether you use a simple discount or discount according to attribute:

1. Go to **Home > Order Management > Pricing Administration > Manage Service Mappings** under Pricing Configuration.
2. On the Manage Service Mappings page, click **Sales**.
3. On the Edit Service Mapping page, click **Services > Query by Example**, enter the value, then press the Enter key on your keyboard:

Attribute	Value
Service	PriceRequestInternal

Modify Service Mapping for Simple Discount

Here are the modifications that you make in the Sales service mapping:



You add the TermName_Custom attribute to the TermQueue entity of the PriceRequestInternal service.

If you need to add your discount on a tier adjustment, then use the TierLineQueue entity instead of the TermQueue entity. If you need to apply a discount according to an attribute, see *Add Your Own Attributes to Items in Pricing*:

1. In the Details area, in the Entities tab, click the **row** that includes the value:

Attribute	Value
Entity	TermQueue

2. In the Details area, click **Actions > Add Row**, then set the values:

Attribute	Value
Entity	TermQueue

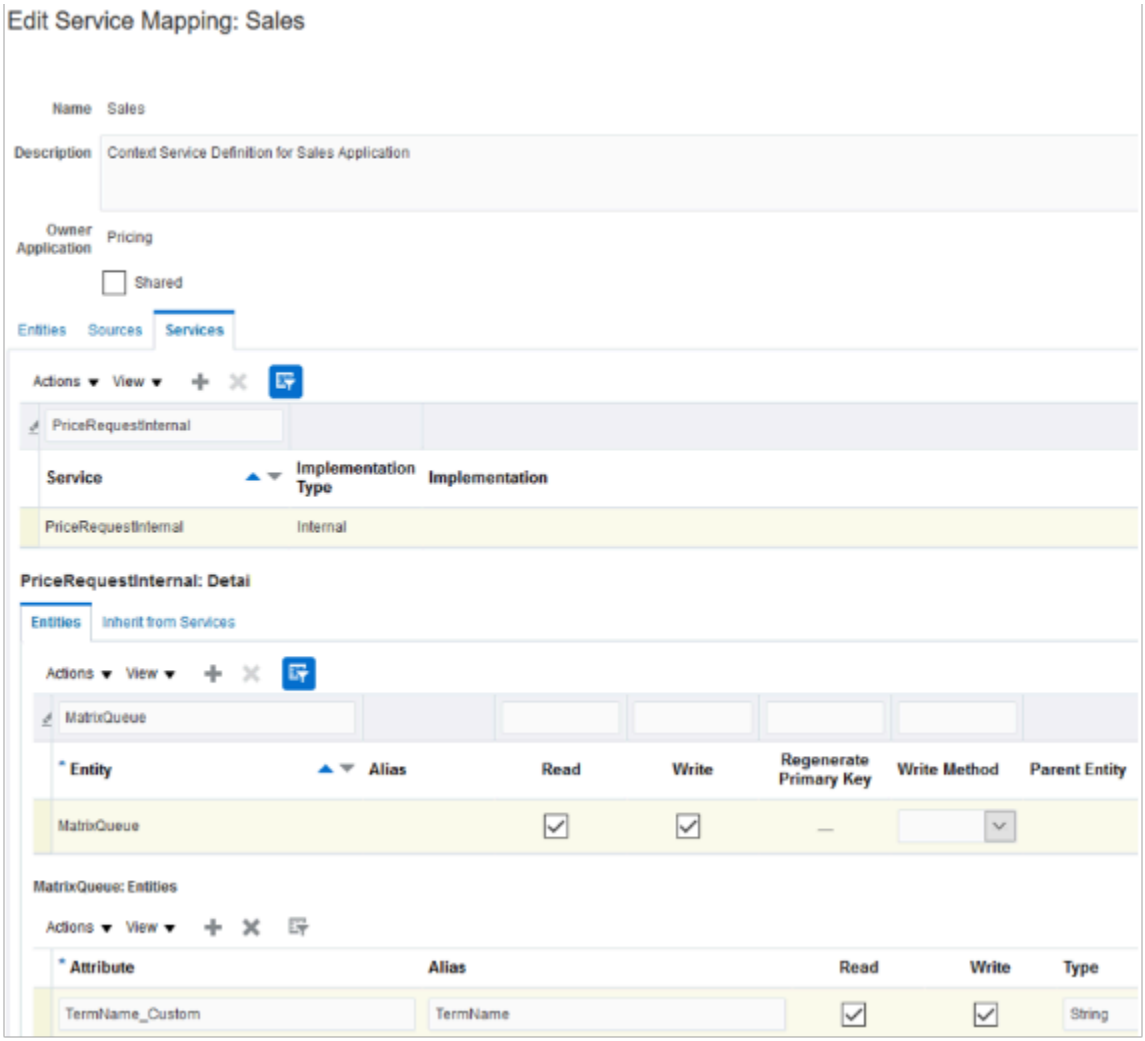
Attribute	Value
Read	Contains a check mark.
Write	Contains a check mark.

3. In the Entities area, click **View > Columns > Show All**, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Attribute	TermName_Custom
Alias	TermName
Read	Contains a check mark.
Write	Contains a check mark.
Type	String

Modify Service Mapping for Discount According to Attribute

Here are the modifications that you make in the Sales service mapping:



You add the TermName_Custom attribute to the MatrixQueue entity of the PriceRequestInternal service.

1. In the Details area, in the Entities tab, click the **row** that includes the value:

Attribute	Value
Entity	MatrixQueue

2. In the Details area, click **Actions > Add Row**, then set the values:

Attribute	Value
Entity	MatrixQueue
Read	Contains a check mark.

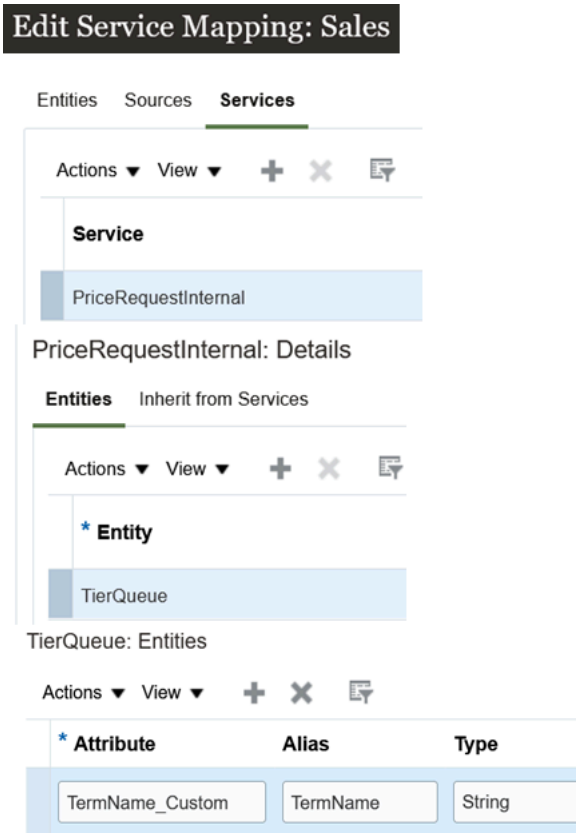
Attribute	Value
Write	Contains a check mark.

3. In the Entities area, **View > Columns > Show All**, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Attribute	TermName_Custom
Alias	TermName
Read	Contains a check mark.
Write	Contains a check mark.
Type	String

Modify Service Mapping for Tier Adjustments

You will add the TierQueue entity.



Try it:

1. Click **Tasks > Manage Service Mappings** under Pricing Configuration.
2. On the Manage Service Mappings page, in the Name column, click **Sales**.
3. On the Edit Service Mapping page, click **Services**.
4. Click **Query By Example**, then query for PriceRequestInternal.
5. In the PriceRequestInternal Details area, on the Entities tab, click **Query By Example**, then query for TierQueue.
6. In the TierQueue Entities area, click **View > Columns**, then make sure Type contains a check mark.
7. Click **Actions > Add Row**, then set the values:

Attribute	Value
Name	TermName_Custom
Alias	TermName
Type	String

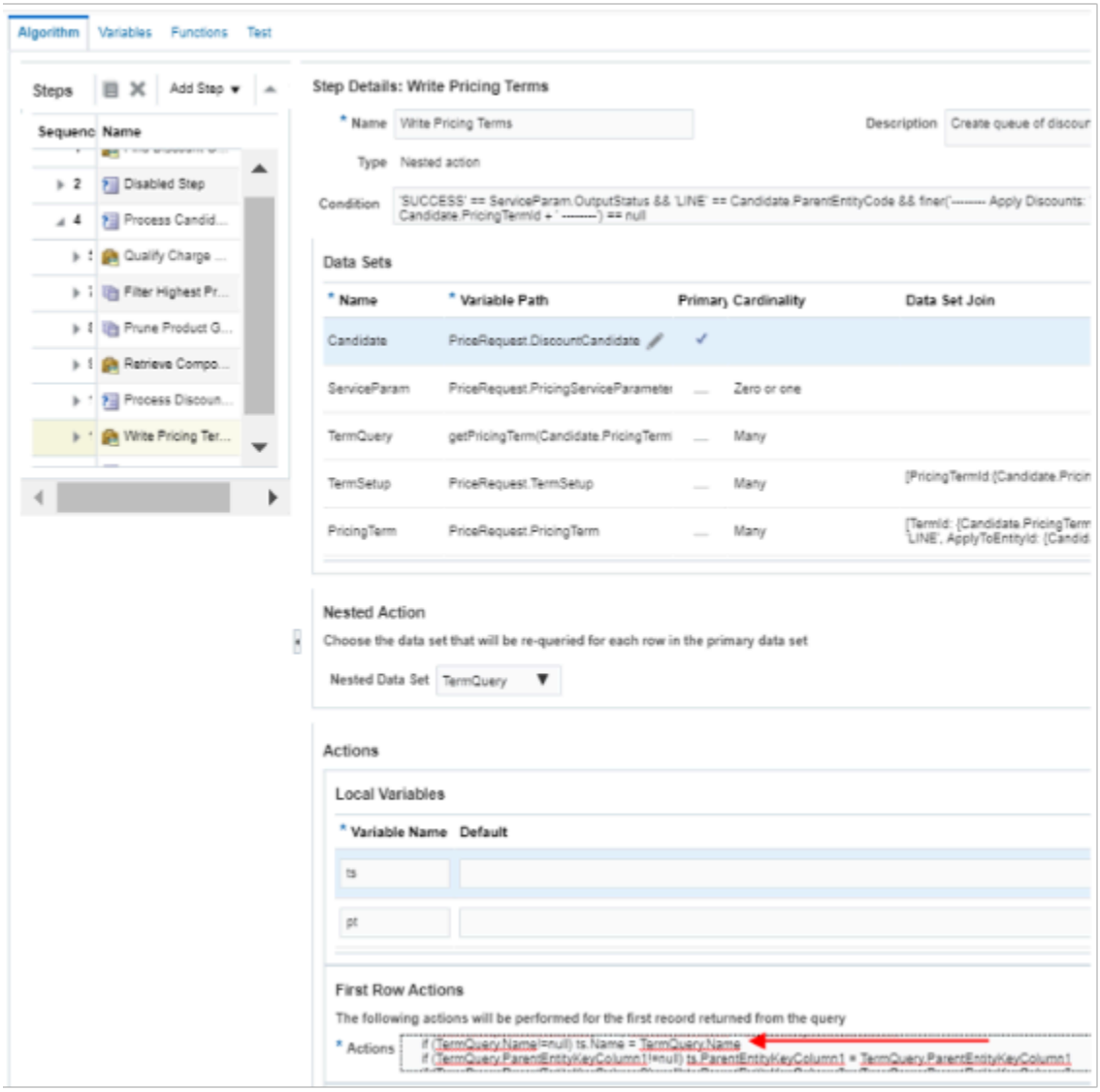
8. Click **Save and Close**.

2. Modify Pricing Algorithms

You modify pricing algorithms that calculate the discounts.

Modify the Pricing Algorithm That Applies Discounts

Here's the algorithm that you modify:



Modify the pricing algorithm that applies discounts:

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, click **Query by Example**, enter the value, then press the **Enter** key on your keyboard:

Attribute	Value
Name	Apply Discounts

3. Click **Action > Create Version**.
4. In the Name column, click the **link** for the version you just created.
5. On the Edit Algorithm page, expand the **steps** until you locate the Write Pricing Term Rules step.
6. Click the **row** that includes Write Pricing Term Rules in the Name column.
7. In the Step Details area, in the First Row Actions area, locate this code:

```
//if (TermQuery.Name!=null) ts.Name = TermQuery.Name
```

8. Remove the forward slashes (//) from the code.

   For example:

   ```
   if (TermQuery.Name!=null) ts.Name = TermQuery.Name
   ```

   This step instructs the algorithm to examine the TermQuery attribute that you set up in the service mapping earlier in this topic.
9. Click **Save and Close > Actions > Publish**.

Modify the Pricing Algorithm That Applies Pricing Terms

Modify the pricing algorithm that applies pricing terms so it can process the running net price:

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, click **Query by Example**, enter the value, then press the **Enter** key on your keyboard:

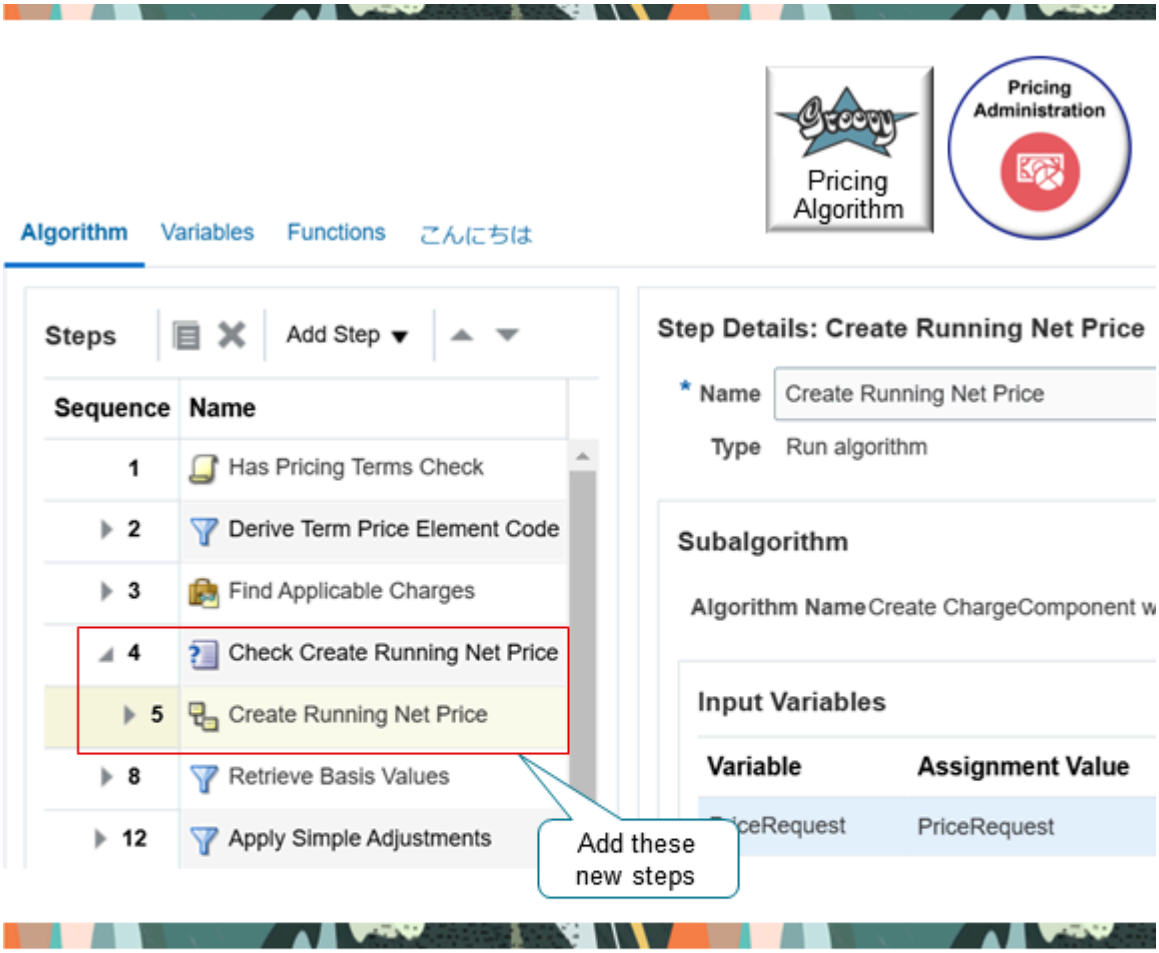
Attribute	Value
Name	Apply Pricing Terms

3. Click **Actions > Create Version**.
4. In the Name column, click the **link** for the version that you just created.
5. Click **Variables > Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Name	RunningNetPriceElem
Data Type	String
Input/Output	None
Default Expression	'ELEMENT_FOR_NET_PRICE' This value identifies the pricing element you created earlier in this topic. You must include the single quotation marks (').

6. Click **Algorithm**.
7. In the Steps area, click the **row** that says Queue Terms Against Lines in the Name column.
8. Add the step that creates the running net price:
   - Click **Add Step > Composite Step**, then click **If**.

Here's the step that you add.



- In the Step Details area, set the values, then click **Save**:

Attribute	Value
Name	Check Create Running Net Price
Condition	`'DISCOUNT_LINE' == TermType` You must include the single quotation marks (').

Attribute	Value

- In the Steps area, expand the **row** you just created, Check Create Running Net Price.
- Click **Add Step > Subalgorithm**.

  You can add a condition only on the step. So, you create the step, then add the subalgorithm that specifies the pricing algorithm to run and the variable to send to this algorithm.
- In the Step Details area, set the values:

Attribute	Value
Name	Create Running Net Price
Description	This step creates a charge component for a running net price.
Algorithm Name	Create ChargeComponent with RunningUnitPrice

- In the Input Variables area, set values:

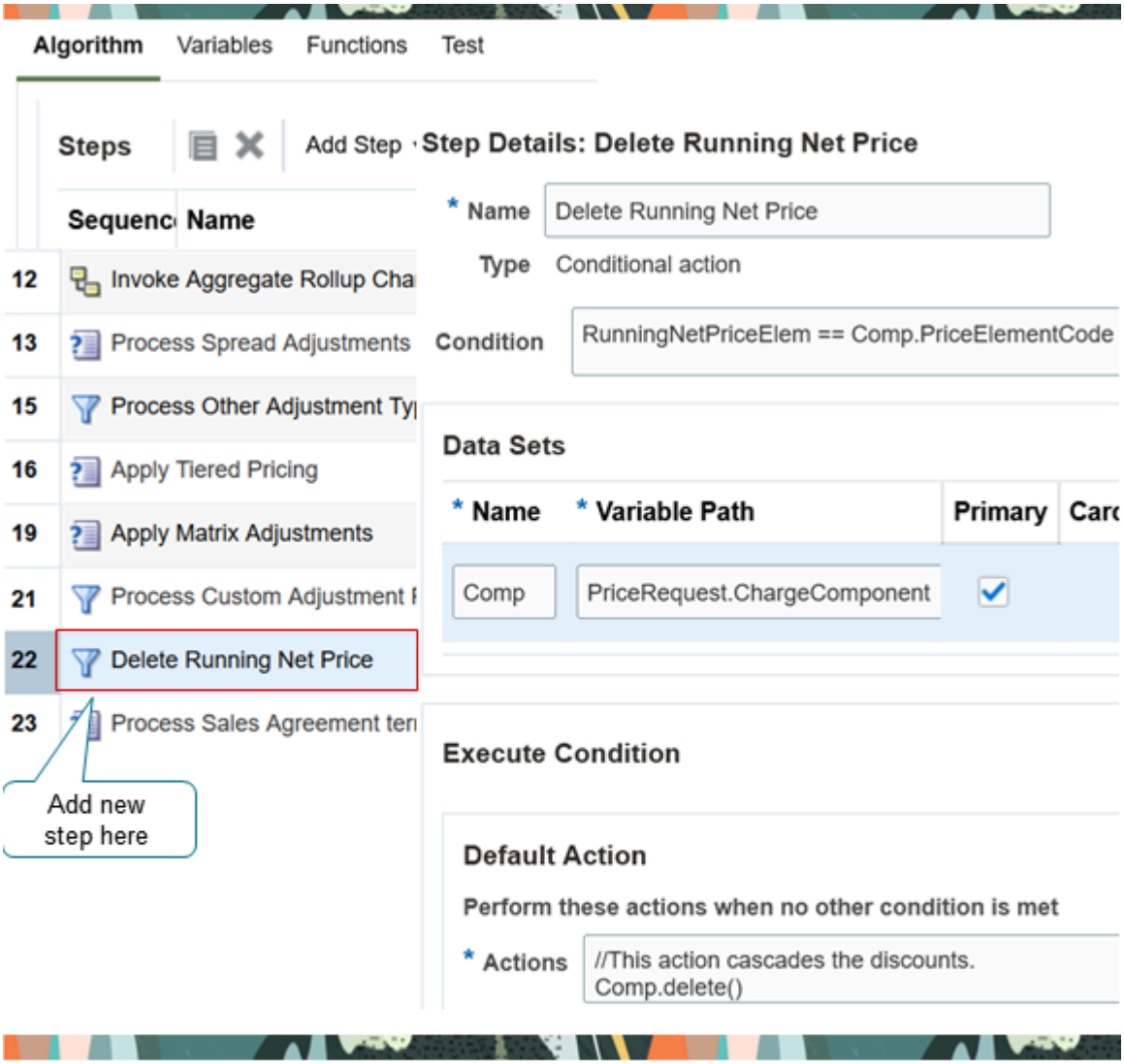
Variable	Assignment Value
PriceRequest	PriceRequest
PriceElementCode	RunningNetPriceElem
IsForCeilingPrice BaseCeilingPriceElementCode	Don't use these variables. They are for Oracle internal use only.
RoundingAdjustmentElementCode	'QP_ROUNDING_ADJUSTMENT'
PerformRounding	false
IsForMargin	false
PriceElementUsageCode	Leave empty
ChargeAppliesToCode	'PRICE'

Variable	Assignment Value
RoundingAdjustmentElementUsageC	'PRICE_ADJUSTMENT'

- In the Output Variables area, set the value:

Variable	Assignment Value
PriceRequest	PriceRequest

9. Add the step that deletes the running net price.



Add the step:

- Click the **step** that has Process Custom Adjustment Flag in the Name column.
- Click **Add Step > Conditional Action**, then set the values:

Attribute	Value
Name	Delete Running Net Price

Attribute	Value
Description	This step deletes a charge component for a running net price.
Condition	`RunningNetPriceElem == Comp.PriceElementCode`

- In the Data Sets area, click **Add Row**, then set the value:

Name	Variable Path	Primary
Comp	PriceRequest.ChargeComponent	Contains a check mark.

- In the Execute Condition area, click **Add Condition > Default Action**, then set the value:

Attribute	Value
Action	`//This action cascades the discounts. Comp.delete()`

- Click **Save**.

10. Make sure the algorithm applies your rules in the sequence that you expect. Do this step when your discount list has more than one rule:
    - Click the **step** that has Process Other Adjustment Types in the Name column.
    - In the Conditional Actions area, in the Then Perform These Actions column, click the pencil.
    - In the Edit Actions dialog, notice the code.

```
finer('\tSetting up queue for applying discount matrix '+TermSetup.AttributePricingMatrixId)

mq = MatrixQ.insert([ParentEntityCode:'CHARGE', ParentEntityId:TermQ.ChargeId])
mq.DynamicMatrixId = TermSetup.AttributePricingMatrixId
mq.ApplyToRollupFlag = ('Y' == TermSetup.ApplyToRollupFlag)
mq.FromCurrencyCode = TermSetup.PricingCurrencyCode
```

    - Locate the line that has `mq.FromCurrencyCode = TermSetup.PricingCurrencyCode`. Its the last line of code.
    - Add a new line after the code that you just located, then add `mq.TermName=TermSetup.Name` to the new line.

      Here's your revised code.

```
finer('\tSetting up queue for applying discount matrix '+TermSetup.AttributePricingMatrixId)

mq = MatrixQ.insert([ParentEntityCode:'CHARGE', ParentEntityId:TermQ.ChargeId])
mq.DynamicMatrixId = TermSetup.AttributePricingMatrixId
mq.ApplyToRollupFlag = ('Y' == TermSetup.ApplyToRollupFlag)
mq.FromCurrencyCode = TermSetup.PricingCurrencyCode
mq.TermName=TermSetup.Name
```

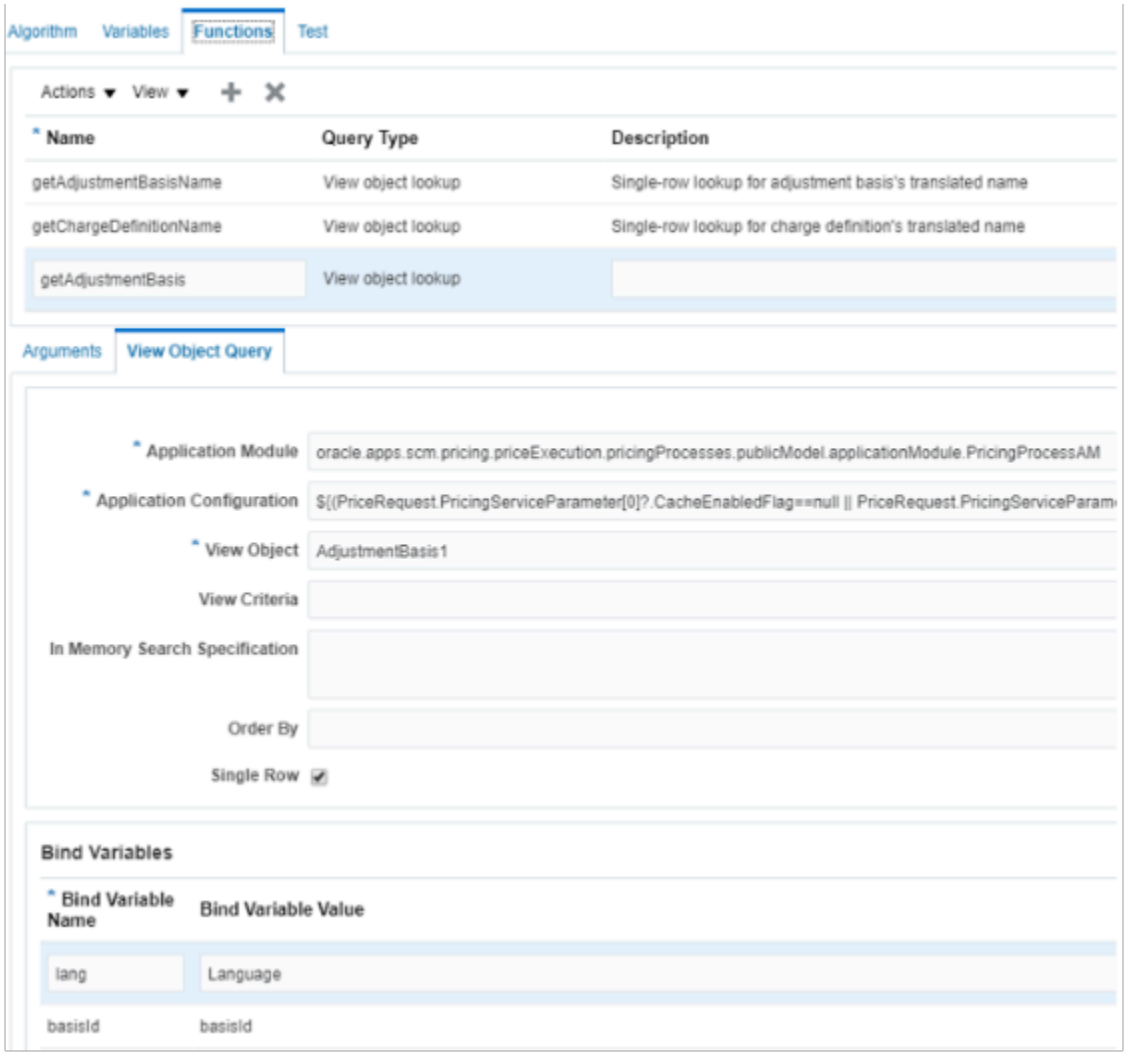
    - Click **OK > Save**.

3. Modify Pricing Algorithms for Simple Discounts

Do this section only if you're applying a simple discount:

1. Add a function:
    - Click **Functions**.

      Here's the function that you will add:



      You add a view object that the pricing algorithm can use in a function that gets the adjustment basis.
    - Click **Actions > Add Row**, then set the value:

Name	Query Type
getAdjustmentBasis	View Object Lookup

Name	Query Type

- In the Arguments area, add the arguments:

Name	Comments
BasisId	Value that identifies the adjustment basis.
Language	Abbreviation that identifies the language.

- Click **View Object Query > Add Row**, then set the values:

Attribute	Value
Application Module	`oracle.apps.scm.pricing.priceExecution.pricingProcesses.publicModel.applicat`
Application Configuration	`${(PriceRequest.PricingServiceParameter[0]?.CacheEnabledFlag==null || PriceRequest.PricingServiceParameter[0].CacheEnabledFlag) ? 'PricingProcessAMShared' : 'PricingProcessAMLocal'}`
View Object	AdjustmentBasis1
Single Row	Contains a check mark.

- In the Bind Variables area, add the bind variables, then click **Save**:

Bind Variable Name	Bind Variable Value
basisId	BasisId
lang	Language

2. Modify the step that gets values for the adjustment basis:
   - Click **Algorithm**.
   - In the Steps area, click the **step** that includes Retrieve Basis Values in the Name column.
   - In the Step Details area, in the Default Action area, scroll all the way to the end, and then locate this code:

```
else {
```

```
finer('\tFound '+component .PriceElementCode+' charge component '+component .ChargeComponentId+'
with unit price '+component .UnitPrice.Value)
TermQ.AdjustmentBasisValue = component .UnitPrice.Value
}
```

- Add this line immediately before the closing curly bracket:

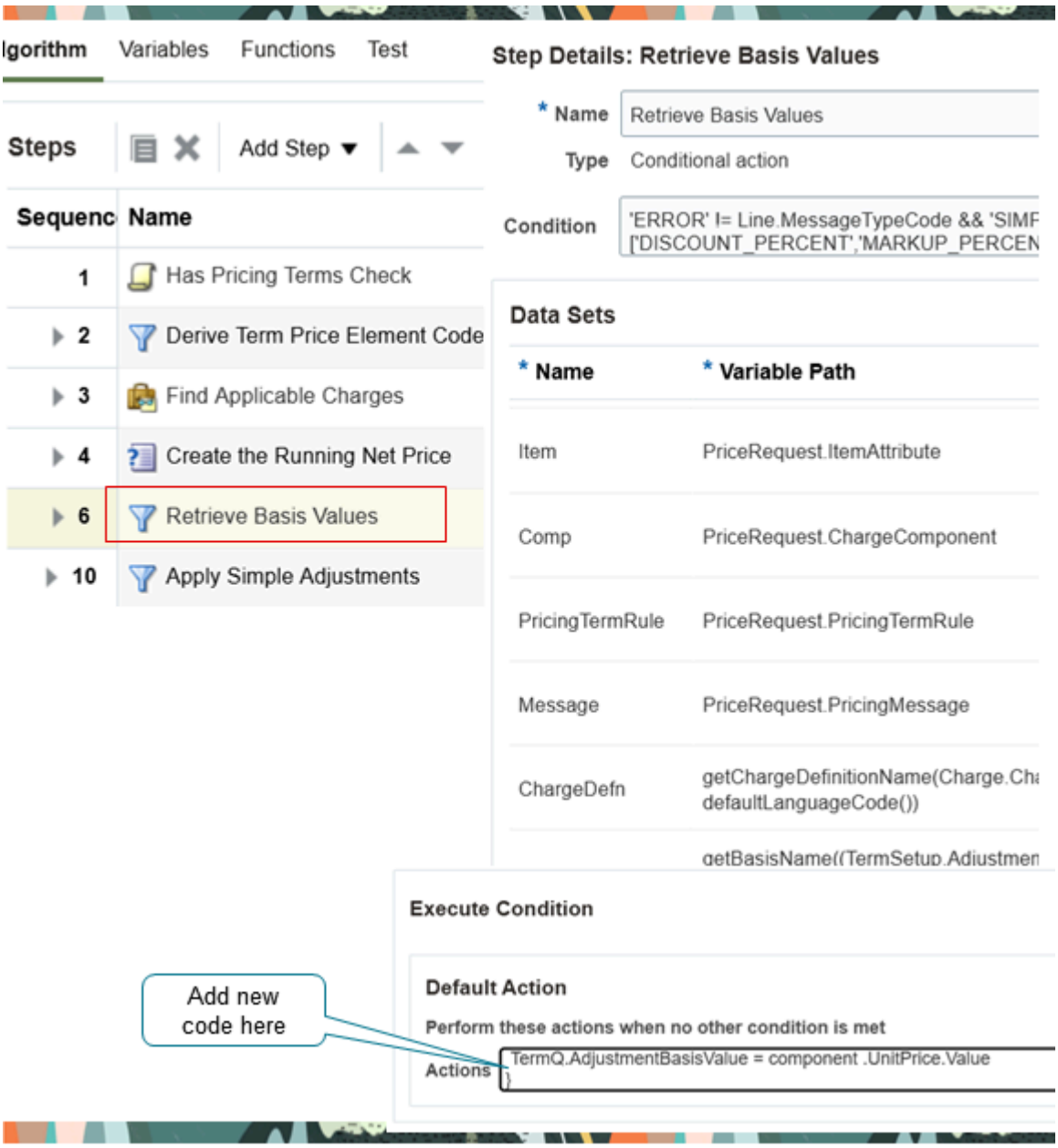
```
//We're adding this line for cascading discounts:
TermQ.TermName = TermSetup.Name
```

For example:

```
else {
 finer('\tFound '+component .PriceElementCode+' charge component '+component .ChargeComponentId+'
 with unit price '+component .UnitPrice.Value)
 TermQ.AdjustmentBasisValue = component .UnitPrice.Value
 //We're adding this line for cascading discounts:
 TermQ.TermName = TermSetup.Name
```

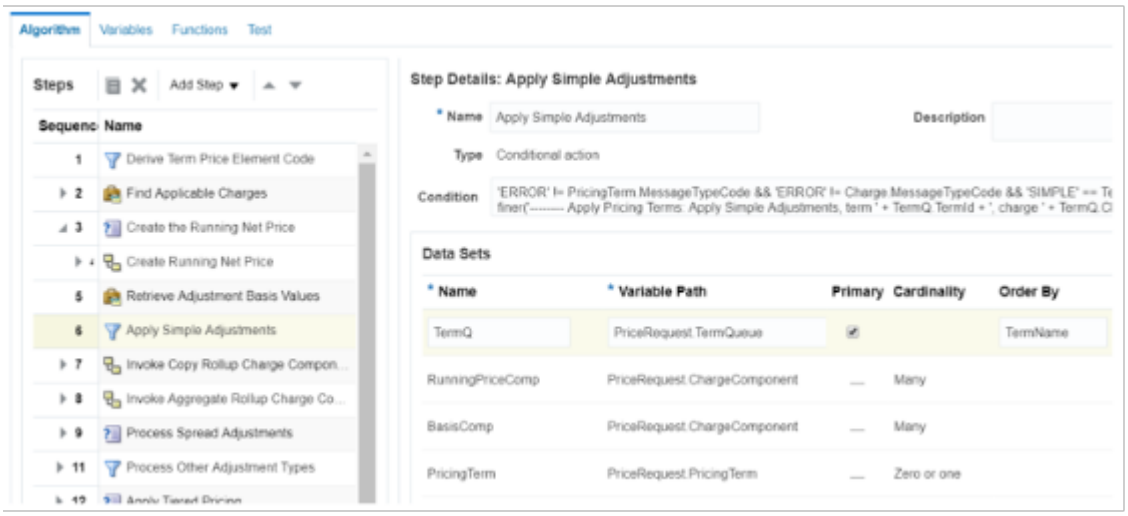
}

You add it here:



- Click **Save**.

3. Modify the step that applies the simple adjustment:



- In the Steps area, click the **step** that includes Apply Simple Adjustments in the Name column.
- In the Data Sets area, add the data sets:

Name	Variable Path	Cardinality
RunningPriceComp	PriceRequest.ChargeComponent	Many This value specifies to use many charge components for the running price to one term in the term queue.
BasisComp	PriceRequest.ChargeComponent	Many This value specifies to use many pricing bases for the charge components to one term in the term queue.

- In the Data Sets area, in the row that includes TermQ in the Name column, set the value:

Attribute	Value
Order By	TermName

- In the Execute Condition area, in the Local Variables area, add the local variables:

Variable Name	Default
BasisElementCode	Leave empty.

Variable Name	Default
RunningPrice	Leave empty.

- In the Default Action area, locate this code.

```
AdjustmentValue = AdjustmentValue * (-1)
 }
 }
 else
```

- Here's the code that cascades the discount according to the adjustment basis. Add it immediately after the code that you just located.

```
// cascade your discount modifications according to the adjustment basis
{
 BasisElementCode = getBasis(TermSetup.AdjustmentBasisId)?.PriceElementCode
 if (BasisElementCode != null) {
 TermQ.AdjustmentBasisValue = BasisComp.locate([ChargeId: TermQ.ChargeId, PriceElementCode:
 BasisElementCode])?.UnitPrice?.Value
}
```

- Locate this code.

```
finer(' Unit price adjustment = '+AdjustmentValue)
```

- Add a closing curly bracket on a new line immediately before the code you just located:

```
}
```

- Here's the entire modified code. The modification is in **bold** font:

```
def runningUnitPriceCurAdj = Charge.RunningUnitPrice
def termInDifferentCurrency = (TermSetup.PricingCurrencyCode != null
 && TermSetup.PricingCurrencyCode != Line.AppliedCurrencyCode && !
TermSetup.AdjustmentTypeCode.contains('PERCENT'))
if (termInDifferentCurrency) {
 convRateRow = locateConversionRate(ConvRate, Header, Line, Charge, PricingTermRule, TermSetup,
 Message, Line.AppliedCurrencyCode, TermSetup.PricingCurrencyCode)
 if (convRateRow) {
 runningUnitPriceCurAdj *= convRateRow.ConversionRate
 }
}
if ('ROOT' == Line.ItemType && 'Y' == TermSetup.ApplyToRollupFlag) {
 finer(' Adjustment type = '+TermSetup.AdjustmentTypeCode)
 finer(' Unit price = '+Charge.RunningUnitPrice)
 finer(' Adjustment amount = '+TermSetup.AdjustmentAmount)
 finer(' Basis value = '+TermQ.AdjustmentBasisValue?:0)
 AdjustmentValue = pricingUtil.computeUnitAdjustment(TermSetup.AdjustmentTypeCode,
 runningUnitPriceCurAdj, TermSetup.AdjustmentAmount, TermQ.AdjustmentBasisValue?:0)
 if (termInDifferentCurrency) {
 convRateRow = locateConversionRate(ConvRate, Header, Line, Charge, PricingTermRule, TermSetup,
 Message, TermSetup.PricingCurrencyCode, Line.AppliedCurrencyCode)
 if (convRateRow) {
 AdjustmentValue *= convRateRow.ConversionRate
 }
 }
 if ('ERROR' != PricingTermRule.MessageTypeCode && AdjustmentValue!=null) {
 spread = Spread.insert([ParentChargeId: Charge.ChargeId])
 spread.UnitAdjustment = AdjustmentValue
 //spread.createDataObject('ExtendedAdjustmentAmount')
```

```
//spread.ExtendedAdjustmentAmount.Value = AdjustmentValue
//spread.ExtendedAdjustmentAmount.CurrencyCode = Charge.CurrencyCode
spread.PriceElementCode = TermElementCode
spread.PriceElementUsageCode = priceElementUsageCode
spread.SourceId = TermSetup.TermId
spread.SourceTypeCode = 'PRICING_TERM'
spread.SpreadId = ++spreadId
// Code Changes for Bug#20265319
spread.ExplanationMessageName = TermExplanationMsg
}
} else {
finer(' Adjustment type = '+TermSetup.AdjustmentTypeCode)
finer(' Unit price = '+Charge.RunningUnitPrice)
finer(' Adjustment amount = '+TermSetup.AdjustmentAmount)
finer(' Basis value = '+TermQ.AdjustmentBasisValue?:0)
if ( TermSetup.AdjustmentTypeCode in
['ORA_QP_ACCRUAL_AMOUNT','ORA_QP_ACCRUAL_DIFFERENCE','ORA_QP_ACCRUAL_PERCENT'] )
{
AdjustmentValue = pricingUtil.computeAccrualAdjustment(TermSetup.AdjustmentTypeCode,
TermSetup.AccrualTypeCode, runningUnitPriceCurAdj,
TermSetup.AdjustmentAmount,TermQ.AdjustmentBasisValue?:0)
TermElementCode = 'QP_ACCRUAL'
priceElementUsageCode = null
//For Un-Reference Return lines, reverse the polarity of adjustment value
if ( Line.UnreferencedReturnLineFlag ) {
AdjustmentValue = AdjustmentValue * (-1)
}
}
else {
BasisElementCode = getBasis(TermSetup.AdjustmentBasisId)?.PriceElementCode
if (BasisElementCode != null) {
TermQ.AdjustmentBasisValue = BasisComp.locate([ChargeId: TermQ.ChargeId, PriceElementCode:
BasisElementCode])?.UnitPrice?.Value
}
AdjustmentValue = pricingUtil.computeUnitAdjustment(TermSetup.AdjustmentTypeCode,
runningUnitPriceCurAdj, TermSetup.AdjustmentAmount, TermQ.AdjustmentBasisValue?:0)
}
finer(' Unit price adjustment = '+AdjustmentValue)
Comp = ChargeComponent.insert([ChargeComponentId:++ServiceParam.ChargeComponentIdCntr])
Comp.ChargeId = TermQ.ChargeId
Comp.createDataObject('UnitPrice')
Comp.UnitPrice.Value = AdjustmentValue
Comp.UnitPrice.CurrencyCode = Charge.CurrencyCode
Comp.CurrencyCode = Charge.CurrencyCode

if (termInDifferentCurrency) {
convRateRow = locateConversionRate(ConvRate, Header, Line, Charge, PricingTermRule, TermSetup,
Message, TermSetup.PricingCurrencyCode, Line.AppliedCurrencyCode)
if (convRateRow) {
Comp.UnitPrice.Value *= convRateRow.ConversionRate
}
}

if ( Charge.PricedQuantity!=null ) {
Comp.createDataObject('ExtendedAmount')
Comp.ExtendedAmount.Value = Comp.UnitPrice.Value*Charge.PricedQuantity.Value
Comp.ExtendedAmount.CurrencyCode = Comp.UnitPrice.CurrencyCode
if (Line.ItemType in ['STANDARD', 'COMPONENT', 'ROOT'] && null != Line.ServiceDuration?.Value &&
null != Line.ServiceDurationPeriodCode ) {
Comp.createDataObject('CoverageExtendedAmount')
if ('ONE_TIME' == Charge.PriceTypeCode) {
Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount.Value
}
else if ('RECURRING' == Charge.PriceTypeCode) {
if (Charge.PricePeriodicityCode != Line.ServiceDurationPeriodCode) {
// Partial Price Period with Conversion Rate from OKC tables
```

```
Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount?.Value *
Charge.PartialPeriodDurationConversionRate?:0
}
else {
//no partial Period pricing based on ServiceDuration
Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount.Value * Line.ServiceDuration?.Value
}
}
Comp.CoverageExtendedAmount.CurrencyCode = Comp.ExtendedAmount.CurrencyCode
}
}
assert Charge.CompSeqCntr != null
Comp.SequenceNumber = (Long) Charge.CompSeqCntr++
Comp.PriceElementCode = TermElementCode
Comp.PriceElementUsageCode = priceElementUsageCode
Comp.ExplanationMessageName = TermExplanationMsg
Comp.PriceValidFrom = Line.PricingDate
Comp.PriceValidUntil = TermSetup.EndDate
Comp.SourceId = TermQ.TermId
Comp.SourceTypeCode = 'PRICING_TERM'
finest('\tCreated ' + Comp.PriceElementCode + ' charge component ' + Comp.ChargeComponentId + '
with unit adjustment ' + Comp.UnitPrice.Value + ' ' + Comp.UnitPrice.CurrencyCode)
// Adjust running unit price
Charge.RunningUnitPrice += Comp.UnitPrice.Value
if (Line.MessageTypeCode != 'ERROR' && Comp.PriceElementCode == 'QP_ACCRUAL')
ChannelProgram.ApplicationStatusCode = 'ORA_APPLIED'
finest('\tAdjusted running unit price by ' + Comp.UnitPrice.Value)
pricingUtil.computeHeaderCurrencyAmounts(Comp, Charge, Line, Header, ConvRate, Messages)
}
```

Make these replacements:

Variable in Code	Attribute from SQL Query You Can Use to Replace the Variable
customer_id	CustomerId
selling_business_unit_id	SellingBusinessUnitId
selling_legal_entity_id	SellingLegalEntityId
inventory_item_id	InventoryItemId
inventory_organization_id	InventoryOrganizationId
unit_code	LineQuantityUOMCode
line_quantity_uom_code	LineQuantityUOMCode

4. Test

1. Sign into Order Management and create a sales order.
2. Set the Customer to a value, such as PennyPack Systems.
3. Click **Actions > View Pricing Segment and Strategy**, then verify that the Segment is the default segment, and that the Strategy is correct for PennyPack Systems.

4. Search for item AS54888 on the order line, wait for the results, tab out of the search attribute, then verify that Order Management gets the price from the price list for PennyPack Systems.
5. Add the AS54888 item to an order line, click **Amount** on the order line, then verify that Pricing correctly calculates the Net Price and applied the discounts.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Assign Pricing Strategy
- Manage Price Lists
- Pricing Guidelines

Examples of Creating Price Lists for Each Customer

Assume your implementation services about 4,000 customers, each customer requires a different percent discount according to the value of one or more attributes, and you use a single pricing strategy for each customer. How will you do it?

Review some examples where you create a price list for each customer:

Example	Description
Set up pricing according to customer and line of business.	• You create an extensible flexfield on the order header that implements your own Line of Business attribute, and you set up pricing to apply a percent discount according to customer and line of business. • The Order Entry Specialist, creates a new sales order, sets the Customer attribute and Line of Business attribute on the order header, then adds an order line. • Pricing applies a percent discount for each item according to customer and line of business.
Set up pricing according to customer, sales channel, and business unit.	• You create an extensible flexfield on the order header that implements your own Sales Channel attribute, and you set up pricing to apply a percent discount according to customer, sales channel, and business unit. • The Order Entry Specialist, creates a new sales order, sets the Customer attribute, Sales Channel attribute, and the Business Unit attribute on the order header, then adds an order line. • Pricing applies a percent discount for each item according to customer, sales channel, and business unit.
Set up pricing according to customer and ship-to address.	• You set up pricing to apply a percent discount according to customer and the ship-to address. • The Order Entry Specialist, creates a new sales order, sets the Customer attribute and Ship-To Address attribute on the order header, then adds an order line. • Pricing applies a percent discount for each item according to customer and ship-to address.

Line of Business Example

Assume customer Computer Service and Rentals includes four lines of business, and each line of business requires a different percent discount. The sales channels are Retail, Corporations, Governments, and Universities:

Set Up	Description
Extensible flexfield	Create an extensible flexfield named Line of Business on the order header.
Customer pricing profile	Create a customer pricing profile that references Attribute_Number1 for the extensible flexfield.
Price list	Here are the values you use for each price list that you create: • Price Type equals One Time • Charge Type equals Sale • Charge Subtype equals Price • Adjustment Type equals Discount Amount • Condition references the Line of Business extensible flexfield

Here are the price lists you set up:

- One for all items that Computer Service and Rentals sells.
- One for item AS54888 Desktop Computer with a sale price of $100.
- One for item AS54999 Desktop Computer with a sale price of $200.

Here are pricing matrixes you create for each price list:

Line of Business	Adjustment Amount for All Items	Adjustment Amount for AS54888	Adjustment Amount for AS54999
Retail	10	20	30
Corporations	20	30	40
Governments	30	40	50
Universities	40	50	60

Sales Channel and Business Unit Example

Assume customer Spruce Hospitals places sales orders through one of four different sales channels and through one of two business units. You must apply a different price and a different discount according to each sales channel and each business unit. The lines of business are Retail, Corporations, Governments, and Universities:

Set Up	Description
Extensible flexfield	Create an extensible flexfield named Sales Channel on the order header.
Customer pricing profile	Create a customer pricing profile that references Attribute_Number1 for the extensible flexfield.

You set up a price list for the Forest Medical Trillium 3000 item with a sale price of $130, Access Set equals Common, and Business Unit equals Care Operations and Care Sales. Here's the pricing matrix you create:

Business Unit in Condition Column	Sales Channel in Condition Column	Adjustment Amount
Care Operations	Hospitals	10
Care Sales	Outpatient Clinics	15
Care Operations	Hospitals	12
Care Sales	Outpatient Clinics	11

Note

- `Header.SellingBusinessUnitId` defines the business unit.
- Sales Channel references the Sales Channel extensible flexfield.
- You set the Adjustment Type to Discount Amount.

You set up another price list for all items that Spruce Hospitals sells. You set Price Type to One Time, Charge Type to Sale, and Charge Subtype to Price. Here's the pricing matrix you create:

Business Unit in Condition Column	Adjustment Amount
Care Operations	10
Care Sales	20

Note

- `Header.SellingBusinessUnitId` defines the business unit.
- You set the Adjustment Type to Discount Percent
- You set the Adjustment Basis to List Price.

Ship-To Example

Assume customer Musical Scores sells full sets of orchestral notation to orchestras throughout the world. They provide different discounts according to ship-to country. You set up a price list for item Orchestral Favorites with a sale price of $100. Here's the pricing matrix you set up:

Ship-To Address in Condition Column	Adjustment Amount
United States	10
Japan	12

Ship-To Address in Condition Column	Adjustment Amount
Korea	14
China	16

You set the Adjustment Type to Discount Amount.

Related Topics

- Pricing Algorithms
- Manage Pricing Algorithms
- Assign Pricing Operations to Pricing Algorithms
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Service Mapping

Create Price Lists for Each Customer

Set up Oracle Pricing so you can apply a different discount for each customer.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements. You must also do additional set up in your environment that this procedure doesn't describe, such as declaring the variables you need to support your code.

Assume you have these requirements:

- **Channel**. You sell through Internet, Call Center, and Broker. You incur the least expense through internet sales and more expense through call center. You incur the most expense through broker, except when selling into China because you have established a favorable relationship with a broker in China.
- **Line of business**. Lines of business include 50% desktop computers, 30% laptop computers, and 20% peripherals. Your desktop computer line is your original line of business. It is long established but in slow decline because your customers increasingly demand portability. Your business plan is to achieve 70% sales through the laptop line within five years. As an incentive, you provide the deepest discounts for laptop sales.
- **Ship-to country**. Your company is most interested in gaining a foothold in emerging and younger markets for the items that it sells. The United States market is already saturated. Japan provides a mature market but maintains some upside potential. Korea is a newer market, and China is emerging.
- **Business unit**. To meet accounting and tax requirements, your accounting department requested that you associate each discounted sale only with the Vision Operations business unit.

Here is the functionality that your technical analysis indicates you must implement:

- Add flexfields that allow the Order Entry Specialist to specify the channel, line of business, ship-to country, and business unit.
- Map flexfields to Pricing and use them as part of the pricing calculation.

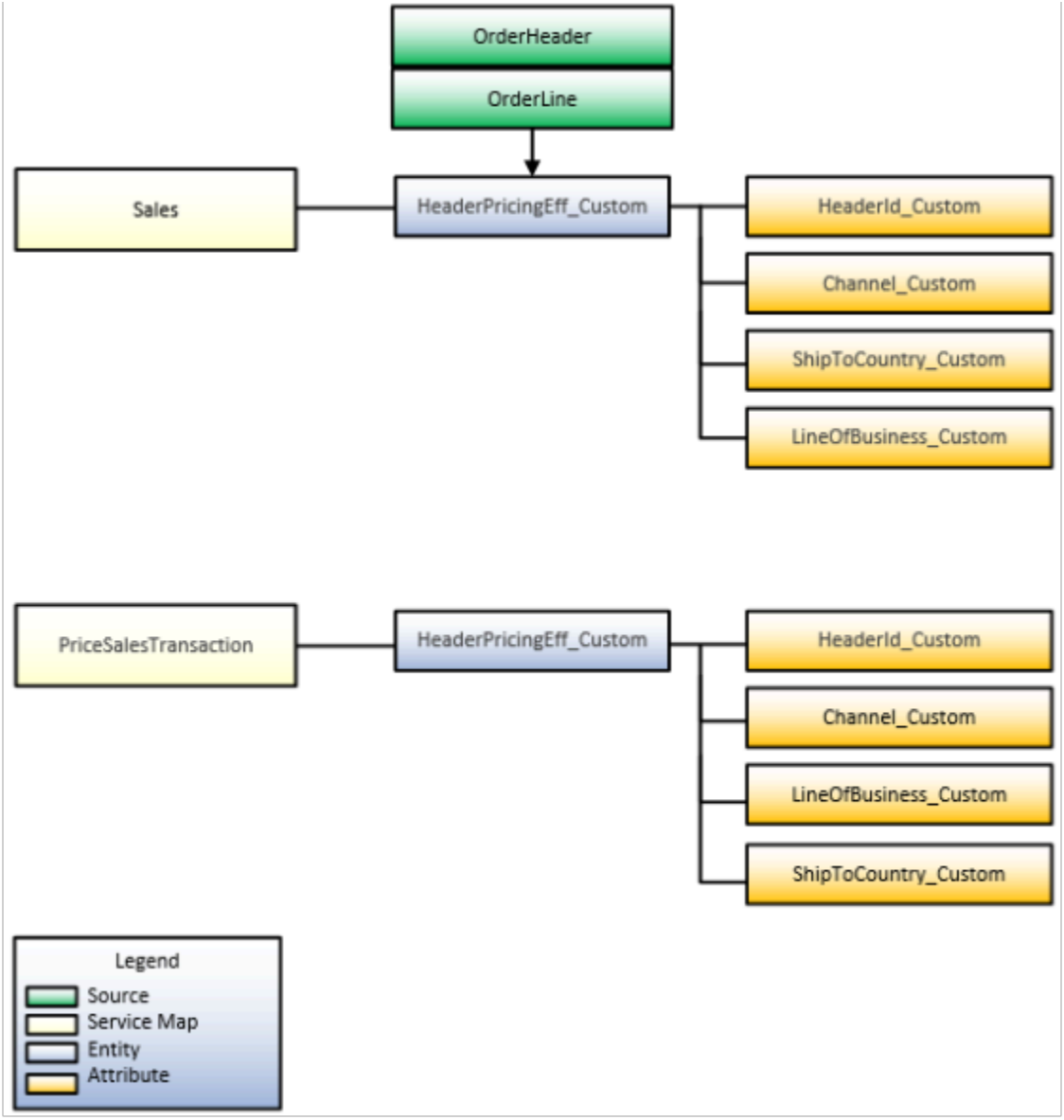
- Create a separate price list that you can use to specify different discounts for each of your major customers, depending on channel, line of business, and ship-to country. Use flexfield values as input to these price lists. Your major customers include:
    - **Fantastic Laptops**. A wholesaler who sells mainly in large quantities to public universities in the United States. You sell through internet and call center. You agreed on specific discounts during contract negotiations after a successful competitive bidding process to provide computing services to public universities.
    - **Computer Service and Rentals**. A retailer who operates in each of the countries where you sell. You sell through internet and call center in all countries except China, where you sell only through a broker. Also, Computer Service and Rentals is a long-term customer. You have negotiated a deeper discount with them over the years.
    - **PennyPack Systems**. Operates only in China and Korea. In China, you sell only through a broker. In Korea, you sell through internet and call centers. Due to the competitive landscape, you provide the deepest discounts of all customers for PennyPack Systems.

Here are the discounts that you identified in consultation with your teams:

Attribute	Percent Discount
Channel	• Sell through broker, 0% • Sell through broker in China, 5% • Sell through call center, 5% • Sell through internet, 10%
Line of Business	• Sell through peripheral line-of-business, 0% • Sell through desktop line-of-business, 0% • Sell through laptop line-of-business, 10%
Ship to Country	• Sell into United States, 0% • Sell into Japan, 5% • Sell into Korea, 10% • Sell into China, 20%
Customer	• Sell to Fantastic Laptops, 5% • Sell to Computer Service and Rentals, 10% • Sell to PennyPack Systems, 20%

The discounts accumulate. For example, if you sell a laptop computer in the United States to Fantastic Laptops through the internet, then the total discount is 25%: 10% for channel, plus 10% for line of business, plus 5% for customer.

Here's the service data object that you will set up. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*:



Note

- The customer pricing profile for each customer assigns the pricing strategy for each customer. For example, if you use Pricing Strategy for Computer Service and Rentals as the pricing strategy for customer Computer Service and Rentals, then set the Pricing Strategy attribute in the global segment of the descriptive flexfield to Pricing Strategy for Computer Service and Rentals.
- At run time, the pricing algorithm gets values from the customer pricing profile and the global segment of the descriptive flexfield. It then uses these values to set the pricing strategy for the sales order, such as Pricing Strategy for Computer Service and Rentals.
- Include the customer name in the pricing strategy name. This technique helps to make sure the strategy name is unique, helps the Order Entry Specialist to understand how the sales order uses the strategy, and helps you to maintain your implementation.
- More than one customer can use the same pricing strategy.

- This topic assumes all business units and customers use the same currency. If you must convert a currency, then add a currency conversion list to the pricing strategy. For details, see *Manage Currency Conversion Lists*.

Summary of the Set Up

1. Create extensible flexfields.
2. Create pricing lookups.
3. Create matrix classes.
4. Create price lists and discount lists.
5. Create pricing profiles and pricing strategies.
6. Modify pricing algorithms.
7. Test your set up.

CAUTION: You use Groovy script to modify a pricing algorithm. Proceed only if you have significant technical experience with writing computer code that interacts with a relational database in an order fulfillment solution.

Create Extensible Flexfields

1. Sign into Order Management with the privileges that you need to administer Order Management.

   Don't sign in with pricing privileges. You can't use them to set up an extensible flexfield.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Order Extensible Flexfields**.

   This example assumes you're implementing the Order Management offering. If you're using a different offering, then click it instead of Order Management.
3. Modify the extensible flexfield:
   - On the Manage Order Extensible Flexfields page, enter the value, then click **Search**:

Attribute	Value
Name	Header Information

   - Click **Actions > Edit**.
   - Click **Manage Contexts**.
   - On the Manage Contexts page, click **Actions > Create**.
   - On the Create Context page, set values:

Attribute	Value
Display Name	HeaderPricingEff
Code	HeaderPricingEff
API Name	`Headerpricingeff`

Attribute	Value
Enabled	Contains a check mark
Behavior	Single Row

- In the Context Usages area, click **Actions > Create**, set the values, then click **Save**:

Attribute	Value
Name	Additional Header Information
View Privileges	None
Edit Privileges	None

- 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	SalesChannel
Code	SalesChannel
API Name	`saleschannel`
Enabled	Contains a check mark.
Data Type	Character
Table Column	Select any column that's available.
Value Set	10 Characters
Prompt	Sales Channel

- In the Context Sensitive Segments area, click **Actions > Create**.

- On the Create Segment page, set values, then click **Save and Close**:

Attribute	Value
Name	LineofBusiness
Code	LineofBusiness
API Name	`lineofbusiness`
Enabled	Contains a check mark.
Data Type	Character
Table Column	Select any column that's available.
Value Set	10 Characters
Prompt	Line of Business

- On the Edit Context page, click **Save and Close**.
- On the Manage Contexts page, click **Save and Close**.
- On the Edit Extensible Flexfield page, click **Save and Close**.

4. Deploy the extensible flexfield:
   - On the Manage Order Extensible Flexfields page, click **Actions> Deploy Flexfield**.
   - Wait for the deployment dialog to indicate that deployment successfully finished, then click **OK**.
   - Click **Actions > Download Flexfield Archive**, wait for the dialog to indicate that the archive successfully finished, then click **Download**.
   - Save the 10008_DOO_HEADERS_ADD_INFO.zip file to your local hard drive, extract it, then navigate to the folder.

     ```
     . . .10008_DOO_HEADERS_ADD_INFO\oracle\apps\scm\doo\processOrder\flex\headerContextsB\view
     ```

   - Open HeaderEffBHeaderPricingEffprivateVO.xml.
   - Verify that HeaderEffBHeaderPricingEffprivateVO.xml contains these values:

Attribute	Value
EntityAttrName for the header	HeaderId

Attribute	Value
EntityAttrName for the channel segment	Channel
EntityAttrName for the line of business segment	`lineofbusiness`
EntityAttrName for the ship to country segment	`shiptocountry`

Add Extensible Flexfields to Service Mapping

Create a service mapping that maps values from the extensible flexfield into Pricing:

1. Sign out of Order Management, and then sign into Oracle Pricing with administrative privileges.
2. Go to **Home > Order Management > Pricing Administration > Manage Service Mappings** under Pricing Configuration.
3. On page Manage Service Mappings, in the Name column, click **Sales**.
4. Add the entities:
    - On the Edit Service Mappings page, on the Entities tab, click **Action > Add Row**, then set the values:

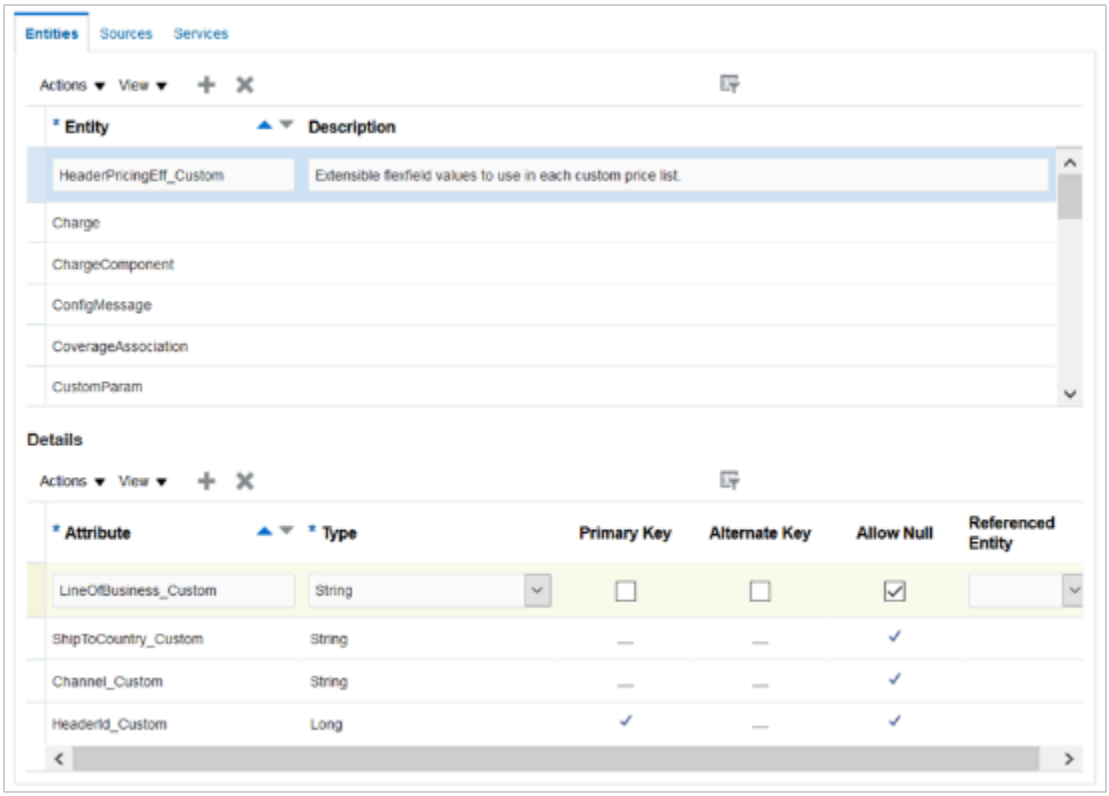
Attribute	Value
Entity	`HeaderPricingEff_Custom`
Description	Get the extensible flexfield values to use in each `custom` price list.

    - In the Details area, add the attributes, then click **Save**:

Attribute	Type	Primary Key
HeaderId_Custom	Long	Contains a check mark.
Channel_Custom	String	Does not contain a check mark.
ShipToCountry_Custom	String	Does not contain a check mark.
LineOfBusiness_Custom	String	Does not contain a check mark.

Attribute	Type	Primary Key

For example:



- Click **Services**, then click the **row** that contains the value:

Attribute	Value
Service	PriceSalesTransaction

- In the Details area, click **Actions**, click **Add Row**, then set the values:

Entity	Read	Write
`HeaderPricingEff_Custom`	Contains a check mark.	Does not contain a check mark.

This step associate the entity you use to send values from the extensible flexfield to Pricing.

- In the HeaderPricingEff_Custom Entities area, add attributes, and then click **Save**:

Attribute	Read	Write
HeaderId_Custom	Contains a check mark.	Does not contain a check mark.
Channel_Custom	Contains a check mark.	Does not contain a check mark.
LineOfBusiness_Custom	Contains a check mark.	Does not contain a check mark.
ShipToCountry_Custom	Contains a check mark.	Does not contain a check mark.

5. Create the sources:
   - Click **Sources**, then click the **row** that includes this value:

Attribute	Value
Source	OrderHeader

   - On the Entity Mapping tab, click **Actions**, click **Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	`HeaderPricingEff_Custom`
Type	View Object
View Object	`HeaderEffBHeaderPricingEffprivateVO` This value must match the name of the public view object that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
Query Type	Unique Identifier
Query Attribute	HeaderId

- In the **HeaderPricingEff_Custom** Details area, add attributes, then click **Save**:

Attribute	View Object Attribute
HeaderId_Custom	HeaderId
Channel_Custom	Channel This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
LineOfBusiness_Custom	**lineofbusiness** This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
ShipToCountry_Custom	**shiptocountry** This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.

This step sets up Pricing to read extensible flexfield values from the public view object that Order Management sends to Pricing when the Order Entry Specialist uses the Price Order action or saves the sales order.

- In the Sources area, click the **row** that includes the value:

Attribute	Value
Source	OrderLine

- In the OrderLine Details area, click **View**, click **Columns**, then add a check mark to Joined Entity and Joined Entity Attribute.
- On the Entity Mapping tab, click **Actions**, click **Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	**HeaderPricingEff_Custom**
Type	View Object
View Object	**HeaderEffBHeaderPricingEffprivateVO**

Attribute	Value
	This value must match the name of the public view object that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
Query Type	Join
Query Attribute	HeaderId
Joined Entity	Header It might be necessary to refresh the row before you set the Joined Entity. To refresh, step out of the row. You can also save your work, cancel the dialog, and then set the Joined Entity.
Joined Entity Attribute	HeaderId

This step maps the extensible flexfield attributes in the public view object. It configures Pricing to read extensible flexfield values from the public view object that Order Management sends to Pricing when Order Management updates the line quantity or shipping attributes on the order line.

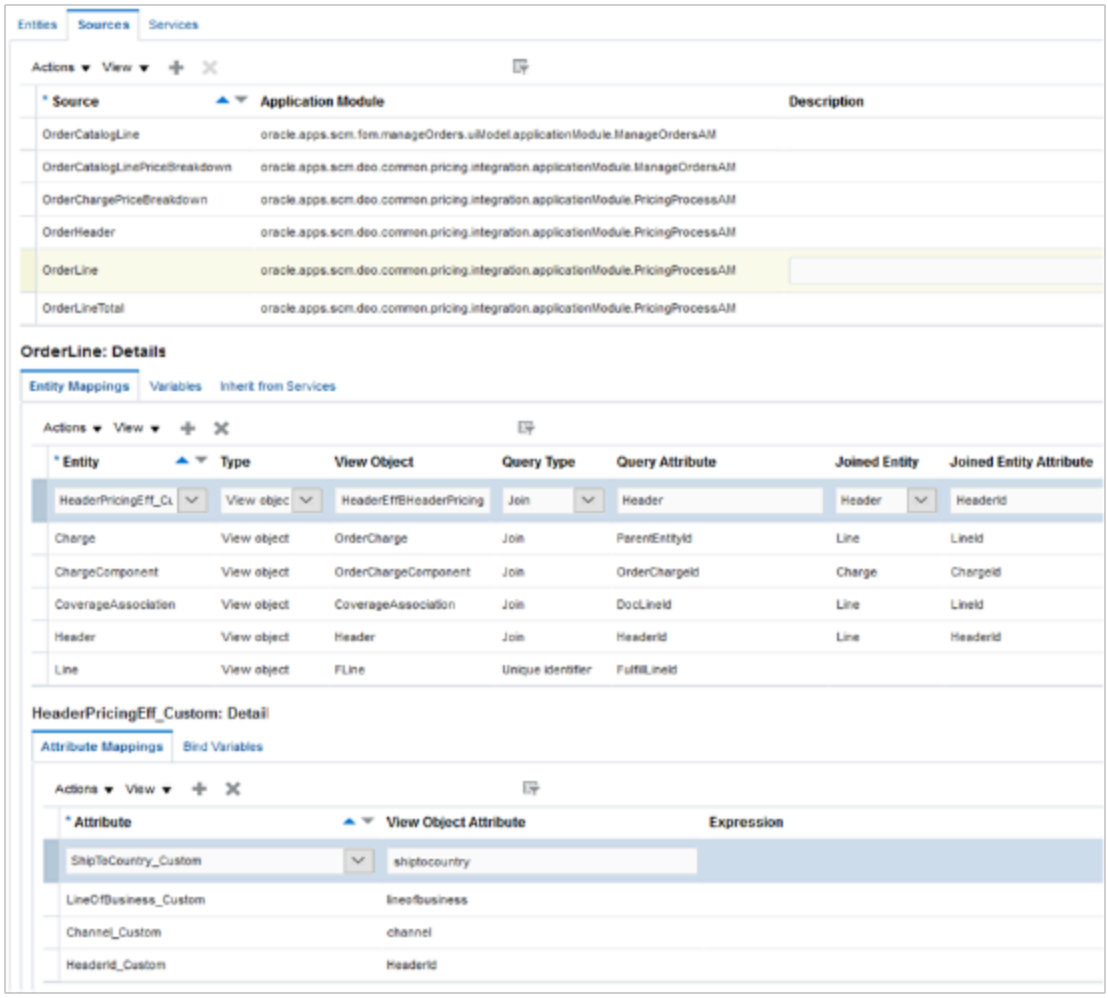
- In the **`HeaderPricingEff_Custom`** Details area, add these attributes:

Attribute	View Object Attribute
HeaderId_Custom	HeaderId This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
Channel_Custom	channel This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
LineOfBusiness_Custom	**`lineofbusiness`** This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.
ShipToCountry_Custom	**`shiptocountry`** This value must match the value of EntityAttrName that you noted earlier in this topic when you examined the contents of the 10008_DOO_HEADERS_ADD_INFO.zip file.

Attribute	View Object Attribute

This step sets up Pricing to read the extensible flexfield values from the public view object that Order Management sends to Pricing when Order Management updates the quantity or shipping fields on the order line.

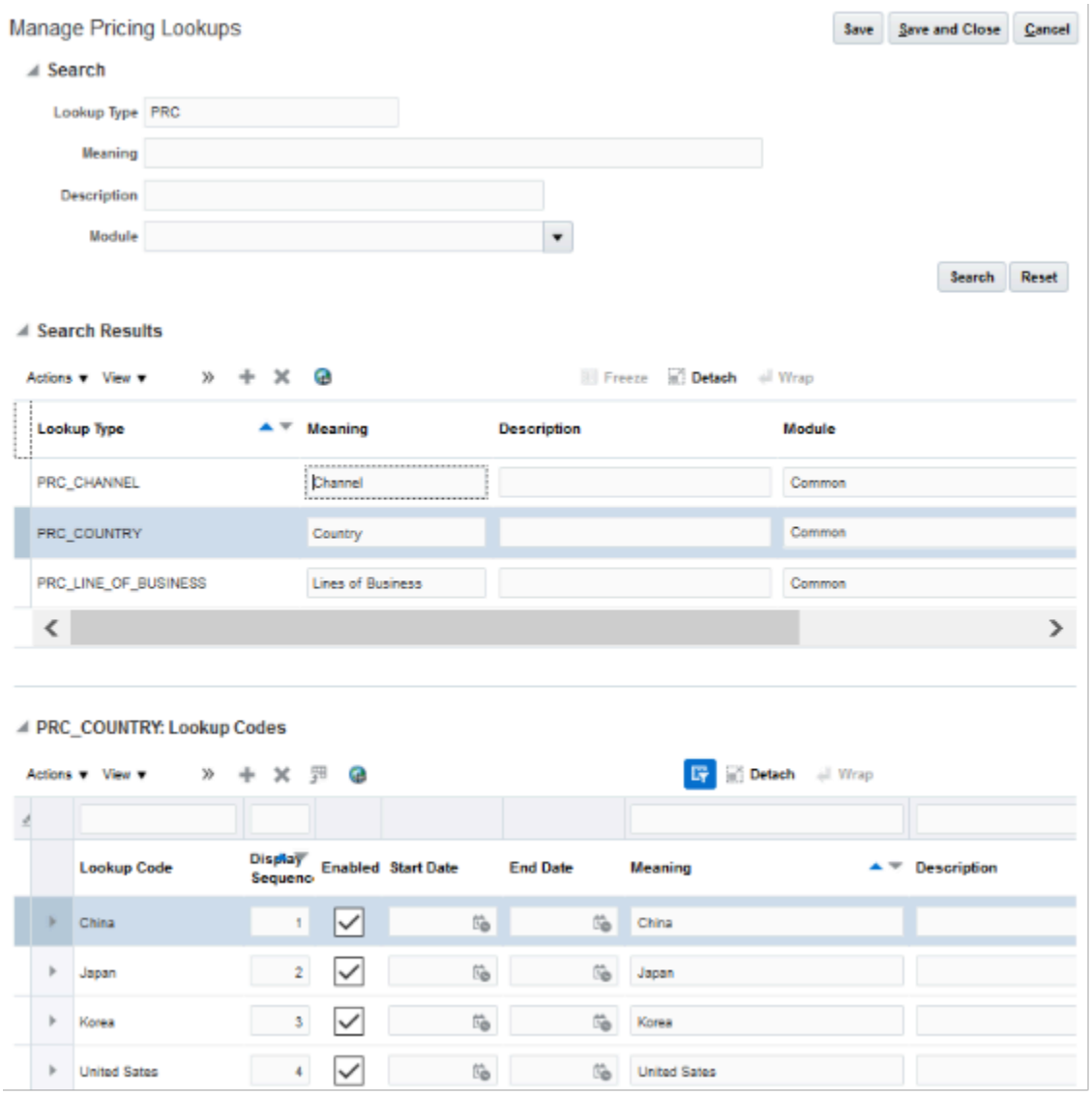
For example:



6. Click **Save and Close > Done**.

Create Pricing Lookups

Here are the lookups you create. The Order Entry Specialist uses them to set attribute values:



Create the lookups:

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Lookups**.
2. On the Manage Pricing Lookups page, in the search results, click **Actions > New**, create the lookup, then click **Save**:

Attribute	Meaning
Lookup Type	PRC_COUNTRY
Meaning	Country
Module	Common

3. In the Lookup Codes area, click **Actions > New**, create lookup codes for countries, then click **Save**:

Lookup Code	Display Sequence	Meaning
China	1	China
Japan	2	Japan
Korea	3	Korea
United States	4	United States

Display Sequence determines the sequence that Order Management uses to display countries when the Order Entry Specialist sets the Ship-to Country attribute. You can display countries in any sequence. In this example, you use Display Sequence to display countries in alphabetic order.

4. Repeat steps 3 and 4 for the Channels lookup. Use these values:

Attribute	Value
Lookup Type	PRC_CHANNELS
Meaning	Channels
Module	Common

Add these lookup codes:

- Internet
- Call Center
- Broker

5. Repeat steps 3 and 4 for the Lines of Business lookup. Use these values:

Attribute	Value
Lookup Type	PRC_LINE_OF_BUSINESS
Meaning	Lines of Business
Module	Common

Attribute	Value

Add these lookup codes:

- Laptops
- Desktops
- Peripherals

6. Click **Save and Close**.

Create Matrix Classes

You set up the same attributes in the matrix class and on the discount list so the discount rule can reference them. For example, you add channels and lines of business in the matrix class and on the discount list:

1. Go to **Home > Order Management > Pricing Administration > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, in the Name column, click **Price List Charge Adjustment**.

   You use this matrix class to get values for your custom attributes, the business unit, and the customer.
3. On the Edit Matrix Class page, in the Condition Columns area, click **Actions > Add Row**, then add the condition:

Attribute	Value
Name	Channel
Source Code Name	Channel
Comparison	Equals
Compare to Attribute	HeaderPricingEff_Custom.Channel_Custom
Allow Null	Contains a check mark
Null is Wildcard	Contains a check mark
Domain	Lookup: PRC_CHANNEL

Here's the condition that this step sets up:

- Set the Channel_Custom attribute on extensible flexfield HeaderPricingEff_Custom to the value that the Order Entry Specialist set in the PRC_CHANNEL lookup.

4. In the Condition Columns area, click **Actions > Add Row**, then add the condition:

Attribute	Value
Name	Line of Business
Source Code Name	LineofBusiness
Comparison	Equals
Compare to Attribute	HeaderPricingEff_Custom.LineofBusiness_Custom
Allow Null	Contains a check mark
Null is Wildcard	Contains a check mark
Domain	Lookup: PRC_LINE_OF_BUSINESS

Here's the condition that this step sets up:

- Set the LineofBusiness_Custom attribute on extensible flexfield HeaderPricingEff_Custom to the value that the Order Entry Specialist set in the PRC_LINE_OF_BUSINESS lookup.

5. In the Condition Columns area, click **Actions > Add Row**, then add the condition:

Attribute	Value
Name	Ship to Country
Source Code Name	ShiptoCountry
Comparison	Equals
Compare to Attribute	HeaderPricingEff_Custom.ShiptoCountry_Custom
Allow Null	Does not contain a check mark
Null is Wildcard	Does not contain a check mark
Domain	Lookup: PRC_COUNTRY

Attribute	Value

Here's the condition that this step sets up:

- Set the ShiptoCountry_Custom attribute on extensible flexfield HeaderPricingEff_Custom to the value that the Order Entry Specialist set in the PRC_COUNTRY lookup.

6. In the Condition Columns area, click **Actions > Add Row**, then add the condition:

Attribute	Value
Name	Business Unit
Source Code Name	BusinessUnit
Comparison	Equals
Compare to Attribute	Header.SellingBusinessUnitId
Allow Null	Does not contain a check mark
Null is Wildcard	Does not contain a check mark

Use these values when you set the Domain:

Attribute	Value
Name	Business Unit
Domain Type	View Object Query
Application Module	MatrixDomainAM (oracle.apps.scm.pricing.common.publicModel.applicationModule.MatrixDomainAM)
Configuration	MatrixDomainAMLocal
View Object	BusinessUnitPVO
Key Attribute	BusinessUnitId

Attribute	Value
Display Attribute	Name
View Criteria	Leave empty
Data Type	Number
View Object Bind Variables	Leave empty

Here's the condition that this step sets up:

- Set the Business Unit attribute to the value that the Order Entry Specialist set in the SellingBusinessUnitId attribute on the order header.

The Business Unit attribute is predefined. The domain specifies how to get the value through a predefined view object.

7. In the Condition Columns area, click **Actions > Add Row**, then add the condition:

Attribute	Value
Name	Customer
Source Code Name	Customer
Comparison	Equals
Compare to Attribute	Header.CustomerId
Allow Null	Does not contain a check mark
Null is Wildcard	Does not contain a check mark

Use these values when you set the Domain:

Attribute	Value
Name	Customer

Attribute	Value
Domain Type	View Object Query
Application Module	ManageOrdersAM (oracle.apps.scm.fom.common.ManageOrders.uiModel.applicationModule.ManageOrdersAM)
Configuration	ManageOrdersAMLocal
View Object	Customer
Key Attribute	PartyId
Display Attribute	PartyName
View Criteria	Leave empty
Data Type	Number
View Object Bind Variables	Leave empty

Here's the condition that this step sets up:

- Set the Customer attribute to the value that the Order Entry Specialist set in the CustomerId attribute on the order header.

The Customer attribute is predefined. The domain specifies how to get this value through a predefined view object.

8. Click **Save**.
9. Create the matrix class that specifies the pricing term adjustment:
    - On the Manage Matrix Classes page, in the Name column, click **Pricing Term Adjustment**.
    - On the Edit Matrix Class page, add all the same conditions that you added to the Price List Charge Adjustment matrix class earlier in this procedure.

      You use the Price List Charge Adjustment matrix class to create the conditions for the Price List Charge Adjustment Matrix. For details, see *Manage Pricing Matrix Types*.

Create Price Lists and Discount Lists

Create Price Lists

Create a separate price list for each of your major customers for the AS54888 laptop computer:

1. Go to **Home > Order Management > Pricing Administration > Price Lists**.
2. Create a new price list:

Attribute	Value
Name	Price List for Fantastic Laptops
Business Unit	Vision Operations
Currency	USD

3. Add an item:

Attribute	Value
Item	AS54888

4. Create a charge:

Attribute	Value
Charge	Sale Price
Based On	Price
Base Price	499
Allow Price Override	Yes

5. Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
6. Add these columns:
    - Channel
    - Line of Business
    - Business Unit
    - Customer

7. Click **Add Row**, then add them:

Channel	Line of Business	Business Unit	Customer	Adjustment Type	Adjustment Amount
Call Center	Laptops	Vision Operations	Fantastic Laptops	Discount Percent	20
Internet	Laptops	Vision Operations	Fantastic Laptops	Discount Percent	25

Note

- 20% discount equals 5% for call center, 10% for laptop, plus 5% for customer
- 25% discount equals 10% for internet, 10% for laptop, plus 5% for customer

8. Click **Additional Information** , then add an access set:

Attribute	Value
Set Code	Common

9. Click **Save > Activate**.
10. Repeat these steps for the Computer Service and Rentals customer. Use the same values but with this difference:

Attribute	Value
Name	Price List for Computer Service and Rentals

Add these rows:

Row	Channel	Line of Business	Ship to Country	Business Unit	Customer	Adjustment Type	Adjustment Amount
1	Call Center	Laptops	United States	Vision Operations	Computer Service and Rentals	Discount Percent	25
2	Internet	Laptops	United States	Vision Operations	Computer Service and Rentals	Discount Percent	25

Row	Channel	Line of Business	Ship to Country	Business Unit	Customer	Adjustment Type	Adjustment Amount
3	Call Center	Laptops	Japan	Vision Operations	Computer Service and Rentals	Discount Percent	30
4	Internet	Laptops	Japan	Vision Operations	Computer Service and Rentals	Discount Percent	35
5	Call Center	Laptops	Korea	Vision Operations	Computer Service and Rentals	Discount Percent	35
6	Internet	Laptops	Korea	Vision Operations	Computer Service and Rentals	Discount Percent	40
7	Broker	Laptops	China	Vision Operations	Computer Service and Rentals	Discount Percent	45

Note these calculations for each row:

Total Discount	Calculation
Row 1, 25%	5% channel, 10% line of business, 0% country, 10% customer
Row 2, 30%	10% channel, 10% line of business, 0% country, 10% customer
Row 3, 30%	5% channel, 10% line of business, 5% country, 10% customer
Row 4, 35%	10% channel, 10% line of business, 5% country, 10% customer
Row 5, 35%	5% channel, 10% line of business, 10% country, 10% customer
Row 6, 40%	10% channel, 10% line of business, 10% country, 10% customer
Row 7, 45%	5% channel, 10% line of business, 20% country, 10% customer

Total Discount	Calculation

11. Repeat these steps for the PennyPack Systems customer. Use the same values but with this difference:

Attribute	Value
Name	Price List for PennyPack Systems

Add these rows:

Row	Channel	Line of Business	Ship to Country	Business Unit	Customer	Adjustment Type	Adjustment Amount
1	Call Center	Laptops	Korea	Vision Operations	PennyPack Systems	Discount Percent	45
2	Internet	Laptops	Korea	Vision Operations	PennyPack Systems	Discount Percent	50
3	Broker	Laptops	China	Vision Operations	PennyPack Systems	Discount Percent	55

Note these calculations for each row:

Total Discount	Calculation
Row 1, 45%	5% channel, 10% line of business, 10% country, 20% customer
Row 2, 50%	10% channel, 10% line of business, 10% country, 20% customer
Row 3, 55%	5% channel, 10% line of business, 20% country, 20% customer

Create Discount Lists

1. Create a discount list:

Attribute	Value
Name	Discount List for Fantastic Laptops

Attribute	Value
Business Unit	Vision Operations
Currency	USD
Price Type	All
Charge Type	Sale
Charge Subtype	Price
Line Type	Buy

2. In the search results, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Item Level	All Items
Pricing UOM	Each
Line Type	Buy

This step specifies to apply the discounts that this discount list creates to any sale item that meets the discount rules.

3. Create the customer discount:
    - In the Discount Rules area, click **Actions > Create**, then click **Attribute Based Rule**.
    - In the Create Discount Rule dialog, set the values, then click **OK**:

Attribute	Value
Price Type	All
Charge Type	Sale
Charge Subtype	Price

Attribute	Value
Name	Customer Discount

- In the Details area, click **Actions > Edit Rules Table Columns**.
- In the Edit Rules Table Columns dialog, in the Condition Columns area, click **Actions > New**, set the values, then click **OK**:

Attribute	Value
Name	Customer
Source Code Name	Customer
Comparison	Equals
Compare to Attribute	Header.CustomerId
Allow Null	Does not contain a check mark
Null is Wildcard	Does not contain a check mark

Use these values when you set the Domain:

Attribute	Value
Name	Customer
Domain Type	View Object Query
Application Module	ManageOrdersAM (oracle.apps.scm.fom.common.ManageOrders.uiModel.applicationModule.ManageOrdersAM)
Configuration	ManageOrdersAMLocal
View Object	Customer

Attribute	Value
Key Attribute	PartyName
Display Attribute	PartyName
View Criteria	Leave empty
Data Type	Text
View Object Bind Variables	Leave empty

The condition in this step sets the Customer attribute to the value that the Order Entry Specialist set in the CustomerId attribute on the order header.

The Customer attribute is predefined. The domain specifies how to get the value through a predefined view object.

- In the Details area, add rows, then click **Save**:

Customer	Adjustment Type	Adjustment Amount
Fantastic Laptops	Discount Percent	5
Computer Service and Rentals	Discount Percent	10
PennyPack Systems	Discount Percent	20

4. Create the line-of-business discount:
   - In the Discount Rules area, click **Actions > Create**, then click **Attribute Based Rule**.
   - In the Create Discount Rule dialog, set the values, then click **OK**:

Attribute	Value
Price Type	All
Charge Type	Sale
Charge Subtype	Price

Attribute	Value
Name	Line of Business Discount

- In the Details area, click **Actions > Edit Rules Table Columns**.
- In the Edit Rules Table Columns dialog, in the Condition Columns area, click **Actions > New**, set the values, then click **OK**:

Attribute	Value
Name	Line of Business
Source Code Name	LineofBusiness
Comparison	Equals
Compare to Attribute	HeaderPricingEff_Custom.LineofBusiness_Custom The service mapping provides this value. If this attribute doesn't display, then examine the service mapping you set up earlier in this topic.
Allow Null	Contains a check mark
Null is Wildcard	Contains a check mark
Domain	Lookup: PRC_LINE_OF_BUSINESS

The condition in this step sets the LineofBusiness_Custom attribute on extensible flexfield HeaderPricingEff_Custom to the value that the Order Entry Specialist set in the PRC_LINE_OF_BUSINESS lookup:

- In the Details area, add the rows, then click **Save**:

Channel	Adjustment Type	Adjustment Amount
Call Center	Discount Percent	5
Internet	Discount Percent	10

5. Create the country discount:
    - In the Discount Rules area, click **Actions > Create**, then click **Attribute Based Rule**.
    - In the Create Discount Rule dialog, set the values, then click **OK**:

Attribute	Value
Price Type	All
Charge Type	Sale
Charge Subtype	Price
Name	Ship to Country Discount

    - In the Details area, click **Actions > Edit Rules Table Columns**.
    - In the Edit Rules Table Columns dialog, in the Condition Columns area, click **Actions > New**, set the values, then click **OK**:

Attribute	Value
Name	Ship to Country
Source Code Name	ShiptoCountry
Comparison	Equals
Compare to Attribute	HeaderPricingEff_Custom.ShiptoCountry_Custom
Allow Null	Does not contain a check mark
Null is Wildcard	Does not contain a check mark
Domain	Lookup: PRC_COUNTRY Tip: To filter items in a list, enter the first few characters, such as PRC, then press the Enter key on your keyboard.

    The condition in this step sets the ShiptoCountry_Custom attribute on extensible flexfield HeaderPricingEff_Custom to the value that the Order Entry Specialist set in the PRC_COUNTRY lookup

- In the Details area, add the rows, then click **Save**:

Customer	Adjustment Type	Adjustment Amount
Japan	Discount Percent	5
Korea	Discount Percent	10
China	Discount Percent	20

6. Create the business unit discount.
7. At the top of the page, click **Approve**.

Create Pricing Profiles and Pricing Strategies

Create Descriptive Flexfields for Pricing Profiles

For details, see *Use Extensible Flexfields with Pricing*:

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Pricing Descriptive Flexfields**.
2. On the Manage Pricing Descriptive Flexfields page, search for the value:

Attribute	Value
Name	Customer Pricing Profile DFF

3. Click **Actions > Edit**.
4. On the Edit Descriptive Flexfield page, click **Actions > Create**.
5. On the Create Segment page, set values, then click **Create Value Set**:

Attribute	Value
Name	Strategy
Code	Strategy
API Name	strategy
Data Type	Number
Table Column	ATTRIBUTE_NUMBER1

Attribute	Value
	Select the lowest attribute number that's available. For example, if ATTRIBUTE_NUMBER1 through ATTRIBUTE_NUMBER5 aren't available, and if ATTRIBUTE_NUMBER6 is available, then select ATTRIBUTE_NUMBER6.

6. On the Create Value Set page, set values, then click **Save and Close**:

Attribute	Value
Value Set Code	pricing_strategies_set
Module	Pricing
Validation Type	Table
Value Data Type	Character
From Clause	qp_pricing_strategies_vl
Value Column Name	Name
ID Column Name	Pricing_Strategy_Id
Where Clause	status_code = 'Approved'
Order By Clause	Name

7. On the Create Segment page, set values, click **Save and Close**, then click **Save and Close** again:

Attribute	Value
Value Set	pricing_strategies_set
Prompt	Strategy
Display Type	List of Values

8. On the Manage Pricing Descriptive Flexfields field page, click **Deploy Flexfield**.

9. Wait for the deployment to finish, then examine the result dialog to confirm that the deployment succeeded without error.

Create Pricing Strategies

Create a pricing strategy for each of your customers. For details, see *Manage Pricing Strategies*:

1. Create a pricing strategy for Fantastic Laptops:
    - Use these values for the pricing strategy:

Attribute	Value
Name	Pricing Strategy for Fantastic Laptops
Business Unit	Vision Operations
Currency	USD
Allow Price List Override	Contains a check mark

    - Select and add a segment price list. Use this value:

Attribute	Value
Name	Price List for Fantastic Laptops

    - Select and add a discount list. Use this value:

Attribute	Value
Name	Discount List for Fantastic Laptops

    - Click **Approve**.
2. Create a pricing strategy for Computer Service and Rentals:
    - Use these values for the pricing strategy:

Attribute	Value
Name	Pricing Strategy for Computer Service and Rentals
Business Unit	Vision Operations

Attribute	Value
Currency	USD
Allow Price List Override	Contains a check mark

- Select and add a segment price list. Use this value:

Attribute	Value
Name	Price List for Computer Service and Rentals

- Select and add a discount list. Use this value:

Attribute	Value
Name	Discount List for Computer Service and Rentals

- Click **Approve**.

3. Define a pricing strategy for PennyPack Systems:
   - Use these values for the pricing strategy:

Attribute	Value
Name	Pricing Strategy for PennyPack Systems
Business Unit	Vision Operations
Currency	USD
Allow Price List Override	Contains a check mark

   - Select and add a segment price list. Use this value:

Attribute	Value
Name	Price List for PennyPack Systems

Attribute	Value

- Select and add a discount list. Use this value:

Attribute	Value
Name	Discount List for PennyPack Systems

- Click **Approve**.

Create Customer Pricing Profiles

For details, see *Manage Pricing Profiles*:

1. Create a customer pricing profile for Fantastic Laptops. Use these values:

Attribute	Value
Customer Name	Fantastic Laptops
Strategy	Pricing Strategy for Fantastic Laptops

2. Create a customer pricing profile for Computer Service and Rentals. Use these values:

Attribute	Value
Customer Name	Computer Service and Rentals
Strategy	Pricing Strategy for Computer Service and Rentals

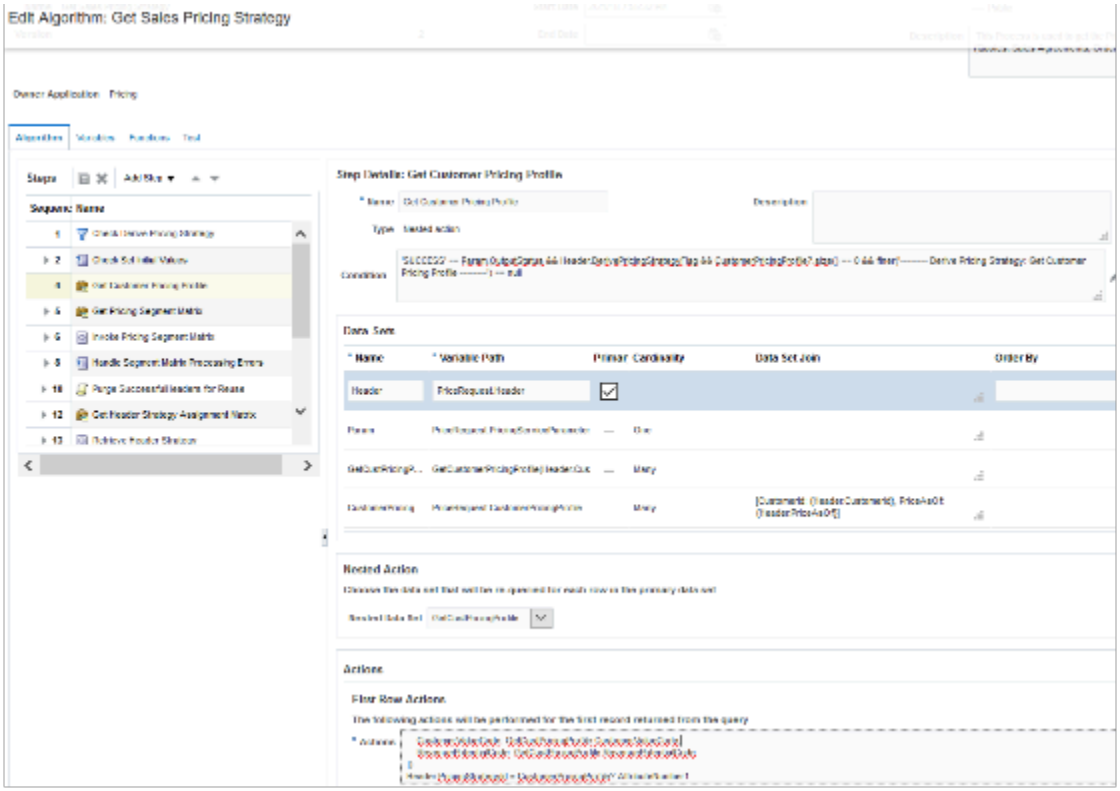
3. Create a customer pricing profile for PennyPack Systems. Use these values:

Attribute	Value
Customer Name	PennyPack Systems
Strategy	Pricing Strategy for PennyPack Systems

Modify Pricing Algorithms

Modify the Get Sales Pricing Strategy Pricing Algorithm

Here's the modification you will make in the Get Sales Pricing Strategy pricing algorithm:



This modification gets the pricing strategy for each customer that you defined in the customer pricing profile, and then assigns this strategy to the order header:

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, select a published version of the pricing algorithm, such as Version 1:

Attribute	Value
Name	Get Sales Pricing Strategy

3. Click **Actions > Create Version**, then wait for the page to display your In Progress version, such as Version 2.
4. Open your In Progress version.
5. On the Edit Algorithm page, in the Algorithm tab, click step 4, **Get Customer Pricing Profile**.
6. In the Step Details area, delete all the code in the Condition window.
7. Add this code in the Condition window.

```
'SUCCESS' == Param.OutputStatus && Header.DerivePricingStrategyFlag && Header.PricingStrategyId == null
 && Header.PricingSegmentCode == null && CustomerPricingProfile?.size() == 0 && finer('-------- Derive
 Pricing Strategy: Get Customer Pricing Profile --------') == null
```

This code adds the values of the descriptive flexfield attributes to the service data object.

8. In the First Row Actions area, add this line immediately after `]).`

```
Header.PricingStrategyId = CustomerPricingProfile?.AttributeNumber1
```

Make sure your revised code matches this code.

```
finest('test customer Pricing profile - on First rows')
CustomerPricingProfile.insert([
 CostToServeCode: GetCustPricingProfile.CostToServeCode,
 CustomerPricingProfileId: getNextId(),
 CustomerRatingCode: GetCustPricingProfile.CustomerRatingCode,
 CustomerSizeCode : GetCustPricingProfile.CustomerSizeCode,
 CustomerValueCode: GetCustPricingProfile.CustomerValueCode,
 RevenuePotentialCode: GetCustPricingProfile.RevenuePotentialCode
])
Header.PricingStrategyId = CustomerPricingProfile?.AttributeNumber1
```

9. If you defined a pricing strategy assignment, then skip this step.

In this example, assume you didn't set up a pricing strategy assignment, but instead you will use the pricing strategy that you referenced earlier in this topic when you defined the descriptive flexfield in the customer pricing profile, and that you will use this profile for each of your customers.

So, delete these steps from the pricing algorithm:

- Get Pricing Segment Matrix
- `Invoke Pricing Segment Matrix`
- Handle Segment Matrix Processing Errors
- Purge SuccessfulHeaders for Reuse
- Get Header Strategy Assignment Matrix
- Retrieve Header Strategy
- Purge Successful Headers

If you prefer not to delete these steps, then do this:

- Don't delete the steps. Instead, add a new step immediately before step Get Pricing Segment Matrix. Use these values for the new step:

Attribute	Value
Type	Composite, If
Name	Disabled Steps
Condition	`false`
Description	This step disables the steps that it contains. It sets up the pricing algorithm to get the pricing segment and pricing strategy from the pricing profile instead of from the disabled steps.

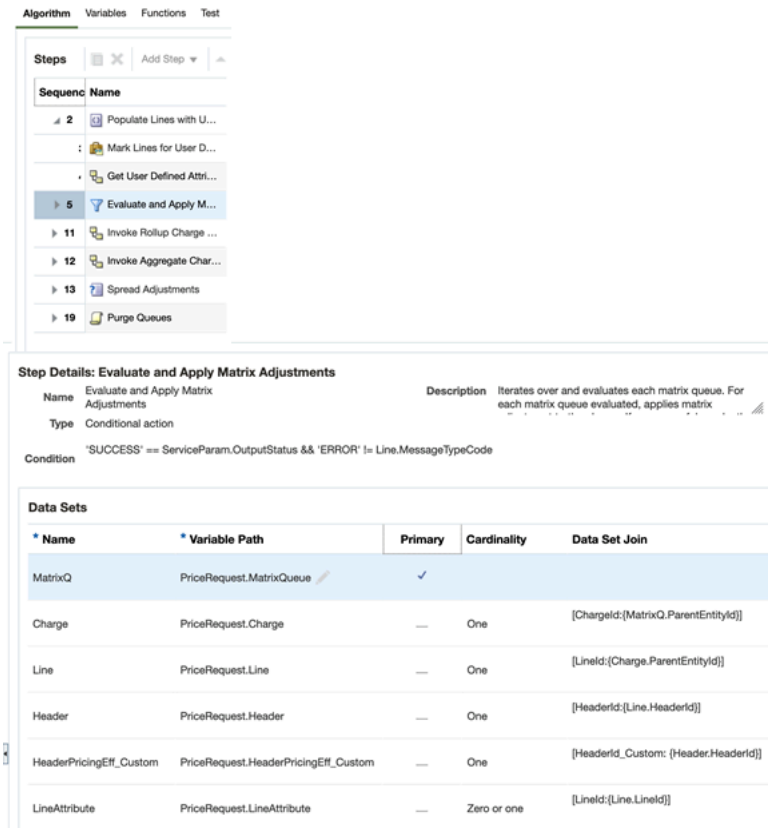
Attribute	Value

- Expand the Disabled Steps step, then use **Move Up** to move each of the steps that you didn't delete into Disabled Steps.

10. Click **Save and Close**.
11. On the Manage Algorithms page, click **Actions > Publish**.

Modify the Pricing Algorithm

Here's the modification that you will make.



This modification adds a data set that references the attribute values for the extensible flexfields you specified earlier in this topic when you defined the adjustment matrix in your price lists, and the adjustment matrix in your discount lists:

1. On the Manage Algorithms page, select a published version of this pricing algorithm, such as Version 1:

Attribute	Value
Name	`Apply Matrices`

2. Click **Actions > Create Version**, then wait for the page to display your In Progress version, such as Version 2.
3. Open your In Progress version.

4. On the Edit Algorithm page, in the Algorithm tab, click the Evaluate and Apply Matrix Adjustments step.

   It might be necessary to expand several steps.
5. In the Step Details area, in the Data Sets area, click the **row** that contains this value:

Attribute	Value
Name	Header

6. In the Data Sets area, click **Add Row**, then set the values:

Attribute	Value
Name	HeaderPricingEff_Custom
Variable Path	PriceRequest.HeaderPricingEff_Custom
Cardinality	Zero or one
Data Set Join	[HeaderId_Custom:{Line.HeaderId}]

7. Click **Save and Close**.
8. On the Manage Algorithms page, click **Actions > Publish**.

Test Your Set Up

Test the Price Sales Transaction Pricing Algorithm

You must publish your algorithms before you can test because they affect Price Sales Transaction:

1. On the Manage Algorithms page, open Price Sales Transaction for editing.
2. On the Edit Algorithm page, click **Test**.
3. Click **Actions > Add Row**, then set the value:

Attribute	Value
Test Case Name	Override Test for Customer Price List

4. In the Test Input area, in the row that contains PriceRequest in the Variable Name column, click the **pencil** in the Variable Value column.
5. In the Edit Variable dialog, delete all the code.
6. Add this code.

```
<?xml version="1.0" encoding="UTF-8"?>
<PriceRequestInternal:PriceRequestInternalType
xmlns:ns0="http://xmlns.oracle.com/adf/svc/types/"
```

```
xmlns:PriceRequestInternal="http://xmlns.oracle.com/apps/scm/pricing/priceExecution/
pricingProcesses/PriceRequestInternal" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="PriceRequestInternal:PriceRequestInternalType">
<PriceRequestInternal:Header>
 <PriceRequestInternal:CustomerId>1028144</PriceRequestInternal:CustomerId>
 <PriceRequestInternal:HeaderId>1</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:CalculatePricingChargesFlag>true</
PriceRequestInternal:CalculatePricingChargesFlag>
<PriceRequestInternal:CalculateShippingChargesFlag>false</
PriceRequestInternal:CalculateShippingChargesFlag>
<PriceRequestInternal:CalculateTaxFlag>false</PriceRequestInternal:CalculateTaxFlag>
<PriceRequestInternal:SellingBusinessUnitId>103</PriceRequestInternal:SellingBusinessUnitId>
<PriceRequestInternal:SellingLegalEntityId>204</PriceRequestInternal:SellingLegalEntityId>
 <PriceRequestInternal:TransactionTypeCode>ORA_SALES_ORDER</PriceRequestInternal:TransactionTypeCode>
 </PriceRequestInternal:Header>
 <PriceRequestInternal:PricingServiceParameter>
 <PriceRequestInternal:PricingContext>SALES</PriceRequestInternal:PricingContext>
 </PriceRequestInternal:PricingServiceParameter>
 <PriceRequestInternal:Line>
 <PriceRequestInternal:HeaderId>101</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:InventoryItemId>100000017351102</PriceRequestInternal:InventoryItemId>
 <PriceRequestInternal:InventoryOrganizationId>204</PriceRequestInternal:InventoryOrganizationId>
 <PriceRequestInternal:LineId>1001</PriceRequestInternal:LineId>
 <PriceRequestInternal:LineCategoryCode>ORDER</PriceRequestInternal:LineCategoryCode>
 <PriceRequestInternal:LineQuantity unitCode="Ea" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">2</PriceRequestInternal:LineQuantity>
 <PriceRequestInternal:LineQuantityUOMCode>"Ea"</PriceRequestInternal:LineQuantityUOMCode>
 <PriceRequestInternal:LineTypeCode>ORA_BUY</PriceRequestInternal:LineTypeCode>
 </PriceRequestInternal:Line>
 <PriceRequestInternal:ChangeSummary logging="false" xmlns:sdo="commonj.sdo"/>
</PriceRequestInternal:PriceRequestInternalType>
```

where

Value	Is An Example For
1028144	CustomerId
103	SellingBusinessUnitId
204	SellingLegalEntityId
100000017351102	InventoryItemId
204	InventoryOrganizationId
Ea	unitCode
Ea	LineQuantityUOMCode

Value	Is An Example For

To get the values that you must use in your environment, see the Getting Identifiers for Test Payloads section, below in this topic.

7. Click **Run Test > Test Output**, then make sure the output payload includes these details:

Attribute	Example Value
StrategyName	Pricing Strategy for Fantastic Laptops
OverridePriceListId	A numeric value, such as 300100231040800. This value identifies the price list that you set up in this example.
Explanation	Base List Price Applied from Price List for Fantastic Laptops
PriceElementCode	QP_BASE_LIST_PRICE
currencyCode	USD
`xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/">399</PriceRequestInternal:UnitPrice`	399 Verify that this value represents the price of the item that Pricing calculates from the price list, which is 399 in this example. Note that the default price list on the pricing strategy might contain a different value, such as 499.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<PriceRequestInternal:PriceRequestInternalType xmlns:ns0="http://xmlns.oracle.com/adf/svc/
types/" xmlns:PriceRequestInternal="http://xmlns.oracle.com/apps/scm/pricing/priceExecution/
pricingProcesses/PriceRequestInternal" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="PriceRequestInternal:PriceRequestInternalType">
 <PriceRequestInternal:Header>
 <PriceRequestInternal:DerivePricingStrategyFlag>true</PriceRequestInternal:DerivePricingStrategyFlag>
 <PriceRequestInternal:PricingDate>2016-10-13T18:58:22.99Z</PriceRequestInternal:PricingDate>
 <PriceRequestInternal:StrategyName>HDO Productions Pricing Strategy</PriceRequestInternal:StrategyName>
 <PriceRequestInternal:AllowCurrencyOverrideFlag>true</PriceRequestInternal:AllowCurrencyOverrideFlag>
 <PriceRequestInternal:AppliedCurrencyCode>USD</PriceRequestInternal:AppliedCurrencyCode>
 <PriceRequestInternal:CalculatePricingChargesFlag>true</
PriceRequestInternal:CalculatePricingChargesFlag>
 <PriceRequestInternal:CalculateShippingChargesFlag>false</
PriceRequestInternal:CalculateShippingChargesFlag>
 <PriceRequestInternal:CalculateTaxFlag>true</PriceRequestInternal:CalculateTaxFlag>
 <PriceRequestInternal:CustomerId>1028144</PriceRequestInternal:CustomerId>
 <PriceRequestInternal:DefaultCurrencyCode>USD</PriceRequestInternal:DefaultCurrencyCode>
```

```
 <PriceRequestInternal:HeaderId>1</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:PriceAsOf>2016-10-13T18:58:22.99Z</PriceRequestInternal:PriceAsOf>
 <PriceRequestInternal:PriceValidFrom>2016-10-13T18:58:22.99Z</PriceRequestInternal:PriceValidFrom>
 <PriceRequestInternal:PricedOn>2016-10-13T18:58:22.99Z</PriceRequestInternal:PricedOn>
 <PriceRequestInternal:PricingStrategyId>300100071304851</PriceRequestInternal:PricingStrategyId>
 <PriceRequestInternal:SellingBusinessUnitId>103</PriceRequestInternal:SellingBusinessUnitId>
 <PriceRequestInternal:SellingLegalEntityId>204</PriceRequestInternal:SellingLegalEntityId>
 <PriceRequestInternal:TransactionTypeCode>ORA_SALES_ORDER</PriceRequestInternal:TransactionTypeCode>
 </PriceRequestInternal:Header>
 <PriceRequestInternal:PricingServiceParameter>
 <PriceRequestInternal:CacheEnabledFlag>false</PriceRequestInternal:CacheEnabledFlag>
 <PriceRequestInternal:ChargeComponentIdCntr>4</PriceRequestInternal:ChargeComponentIdCntr>
 <PriceRequestInternal:ChargeIdCntr>1</PriceRequestInternal:ChargeIdCntr>
 <PriceRequestInternal:CurrentPricingDate>2016-10-13T18:58:22.99Z</
PriceRequestInternal:CurrentPricingDate>
 <PriceRequestInternal:OutputStatus>SUCCESS</PriceRequestInternal:OutputStatus>
 <PriceRequestInternal:PricingContext>SALES</PriceRequestInternal:PricingContext>
 </PriceRequestInternal:PricingServiceParameter>
 <PriceRequestInternal:Line>
 <PriceRequestInternal:AppliedCurrencyCode>USD</PriceRequestInternal:AppliedCurrencyCode>
 <PriceRequestInternal:DefaultPriceListPrecedence>2</PriceRequestInternal:DefaultPriceListPrecedence>
 <PriceRequestInternal:FromCurrencyCode>USD</PriceRequestInternal:FromCurrencyCode>
 <PriceRequestInternal:GlConversionTypeCode>Corporate</PriceRequestInternal:GlConversionTypeCode>
 <PriceRequestInternal:HasAlternatePriceList>true</PriceRequestInternal:HasAlternatePriceList>
 <PriceRequestInternal:ItemType>STANDARD</PriceRequestInternal:ItemType>
 <PriceRequestInternal:PricingDate>2016-10-13T18:58:22.99Z</PriceRequestInternal:PricingDate>
 <PriceRequestInternal:StrategyName>Corporate Pricing Strategy</PriceRequestInternal:StrategyName>
 <PriceRequestInternal:AllowCurrencyOverrideFlag>true</PriceRequestInternal:AllowCurrencyOverrideFlag>
 <PriceRequestInternal:AllowPriceListUpdateFlag>true</PriceRequestInternal:AllowPriceListUpdateFlag>
 <PriceRequestInternal:AppliedPriceListId>300100231040800</PriceRequestInternal:AppliedPriceListId>
 <PriceRequestInternal:DefaultCurrencyCode>USD</PriceRequestInternal:DefaultCurrencyCode>
 <PriceRequestInternal:DefaultPriceListId>300100231040790</PriceRequestInternal:DefaultPriceListId>
 <PriceRequestInternal:HeaderId>1</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:InventoryItemId>100000017351102</PriceRequestInternal:InventoryItemId>
 <PriceRequestInternal:InventoryOrganizationId>204</PriceRequestInternal:InventoryOrganizationId>
 <PriceRequestInternal:LineCategoryCode>ORDER</PriceRequestInternal:LineCategoryCode>
 <PriceRequestInternal:LineId>1</PriceRequestInternal:LineId>
 <PriceRequestInternal:LineQuantity unitCode="Ea" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">2</PriceRequestInternal:LineQuantity>
 <PriceRequestInternal:LineQuantityUOMCode>Ea</PriceRequestInternal:LineQuantityUOMCode>
 <PriceRequestInternal:LineTypeCode>ORA_BUY</PriceRequestInternal:LineTypeCode>
 <PriceRequestInternal:OverridePriceListId>300100231040800</PriceRequestInternal:OverridePriceListId>
 <PriceRequestInternal:PriceAsOf>2016-10-13T18:58:22.99Z</PriceRequestInternal:PriceAsOf>
 <PriceRequestInternal:PriceValidFrom>2016-10-13T18:58:22.99Z</PriceRequestInternal:PriceValidFrom>
 <PriceRequestInternal:PricedOn>2016-10-13T18:58:22.99Z</PriceRequestInternal:PricedOn>
 <PriceRequestInternal:PricingStrategyId>300100071304851</PriceRequestInternal:PricingStrategyId>
 <PriceRequestInternal:ShipToLocationId>929</PriceRequestInternal:ShipToLocationId>
 </PriceRequestInternal:Line>
 <PriceRequestInternal:Charge>
 <PriceRequestInternal:CompSeqCntr>1004</PriceRequestInternal:CompSeqCntr>
 <PriceRequestInternal:NeedsCostPlus>false</PriceRequestInternal:NeedsCostPlus>
 <PriceRequestInternal:NeedsMargin>true</PriceRequestInternal:NeedsMargin>
 <PriceRequestInternal:RunningUnitPrice>399</PriceRequestInternal:RunningUnitPrice>
 <PriceRequestInternal:CanAdjustFlag>false</PriceRequestInternal:CanAdjustFlag>
 <PriceRequestInternal:ChargeAppliesTo>PRICE</PriceRequestInternal:ChargeAppliesTo>
 <PriceRequestInternal:ChargeDefinitionCode>QP_SALE_PRICE</PriceRequestInternal:ChargeDefinitionCode>
 <PriceRequestInternal:ChargeDefinitionId>300100070852423</PriceRequestInternal:ChargeDefinitionId>
 <PriceRequestInternal:ChargeId>1</PriceRequestInternal:ChargeId>
 <PriceRequestInternal:ChargeSubtypeCode>ORA_PRICE</PriceRequestInternal:ChargeSubtypeCode>
 <PriceRequestInternal:ChargeTypeCode>ORA_SALE</PriceRequestInternal:ChargeTypeCode>
 <PriceRequestInternal:CurrencyCode>USD</PriceRequestInternal:CurrencyCode>
 <PriceRequestInternal:EstimatedPricedQuantityFlag>false</
PriceRequestInternal:EstimatedPricedQuantityFlag>
 <PriceRequestInternal:EstimatedUnitPriceFlag>false</PriceRequestInternal:EstimatedUnitPriceFlag>
 <PriceRequestInternal:ParentEntityCode>LINE</PriceRequestInternal:ParentEntityCode>
 <PriceRequestInternal:ParentEntityId>1</PriceRequestInternal:ParentEntityId>
```

```
 <PriceRequestInternal:PriceTypeCode>ONE_TIME</PriceRequestInternal:PriceTypeCode>
 <PriceRequestInternal:PricedQuantity unitCode="Ea" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">2</PriceRequestInternal:PricedQuantity>
 <PriceRequestInternal:PricedQuantityUOMCode>Ea</PriceRequestInternal:PricedQuantityUOMCode>
 <PriceRequestInternal:PrimaryFlag>true</PriceRequestInternal:PrimaryFlag>
 <PriceRequestInternal:RollupFlag>false</PriceRequestInternal:RollupFlag>
 <PriceRequestInternal:SequenceNumber>1</PriceRequestInternal:SequenceNumber>
 <PriceRequestInternal:TaxIncludedFlag>false</PriceRequestInternal:TaxIncludedFlag>
 </PriceRequestInternal:Charge>
 <PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:AbsoluteExtendedAmount>798</PriceRequestInternal:AbsoluteExtendedAmount>
 <PriceRequestInternal:PriceValidFrom>2016-10-13T18:58:22.99Z</PriceRequestInternal:PriceValidFrom>
 <PriceRequestInternal:ChargeComponentId>1</PriceRequestInternal:ChargeComponentId>
 <PriceRequestInternal:ChargeId>1</PriceRequestInternal:ChargeId>
 <PriceRequestInternal:CurrencyCode>USD</PriceRequestInternal:CurrencyCode>
 <PriceRequestInternal:Explanation>Base List Price Applied from HDO Productions PL</
PriceRequestInternal:Explanation>
 <PriceRequestInternal:ExplanationMessageName>QP_BASE_LIST_PRICE_CHARGE</
PriceRequestInternal:ExplanationMessageName>
 <PriceRequestInternal:ExtendedAmount currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">798</PriceRequestInternal:ExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyCode>USD</PriceRequestInternal:HeaderCurrencyCode>
 <PriceRequestInternal:HeaderCurrencyExtendedAmount currencyCode="USD" xmlns:tns="http://
xmlns.oracle.com/adf/svc/errors/">798</PriceRequestInternal:HeaderCurrencyExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyUnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/
adf/svc/errors/">399</PriceRequestInternal:HeaderCurrencyUnitPrice>
 <PriceRequestInternal:PercentOfComparisonElement>1</PriceRequestInternal:PercentOfComparisonElement>
 <PriceRequestInternal:PriceElementCode>QP_BASE_LIST_PRICE</PriceRequestInternal:PriceElementCode>
 <PriceRequestInternal:RollupFlag>false</PriceRequestInternal:RollupFlag>
 <PriceRequestInternal:SequenceNumber>1000</PriceRequestInternal:SequenceNumber>
 <PriceRequestInternal:SourceId>300100231040804</PriceRequestInternal:SourceId>
 <PriceRequestInternal:SourceTypeCode>PRICE_LIST_CHARGE</PriceRequestInternal:SourceTypeCode>
 <PriceRequestInternal:UnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">399</PriceRequestInternal:UnitPrice>
 </PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:AbsoluteExtendedAmount>798</PriceRequestInternal:AbsoluteExtendedAmount>
 <PriceRequestInternal:SkipRunningPrice>true</PriceRequestInternal:SkipRunningPrice>
 <PriceRequestInternal:ChargeComponentId>2</PriceRequestInternal:ChargeComponentId>
 <PriceRequestInternal:ChargeId>1</PriceRequestInternal:ChargeId>
 <PriceRequestInternal:CurrencyCode>USD</PriceRequestInternal:CurrencyCode>
 <PriceRequestInternal:ExtendedAmount currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">798</PriceRequestInternal:ExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyCode>USD</PriceRequestInternal:HeaderCurrencyCode>
 <PriceRequestInternal:HeaderCurrencyExtendedAmount currencyCode="USD" xmlns:tns="http://
xmlns.oracle.com/adf/svc/errors/">798</PriceRequestInternal:HeaderCurrencyExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyUnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/
adf/svc/errors/">399</PriceRequestInternal:HeaderCurrencyUnitPrice>
 <PriceRequestInternal:PercentOfComparisonElement>1</PriceRequestInternal:PercentOfComparisonElement>
 <PriceRequestInternal:PriceElementCode>QP_LIST_PRICE</PriceRequestInternal:PriceElementCode>
 <PriceRequestInternal:PriceElementUsageCode>LIST_PRICE</PriceRequestInternal:PriceElementUsageCode>
 <PriceRequestInternal:RollupFlag>false</PriceRequestInternal:RollupFlag>
 <PriceRequestInternal:SequenceNumber>1001</PriceRequestInternal:SequenceNumber>
 <PriceRequestInternal:TaxIncludedFlag>false</PriceRequestInternal:TaxIncludedFlag>
 <PriceRequestInternal:UnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">399</PriceRequestInternal:UnitPrice>
 </PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:ChargeComponent> <PriceRequestInternal:AbsoluteExtendedAmount>798</
PriceRequestInternal:AbsoluteExtendedAmount>
 <PriceRequestInternal:SkipRunningPrice>true</PriceRequestInternal:SkipRunningPrice>
 <PriceRequestInternal:ChargeComponentId>3</PriceRequestInternal:ChargeComponentId>
 <PriceRequestInternal:ChargeId>1</PriceRequestInternal:ChargeId>
 <PriceRequestInternal:CurrencyCode>USD</PriceRequestInternal:CurrencyCode>
 <PriceRequestInternal:ExtendedAmount currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">798</PriceRequestInternal:ExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyCode>USD</PriceRequestInternal:HeaderCurrencyCode>
```

```
 <PriceRequestInternal:HeaderCurrencyExtendedAmount currencyCode="USD" xmlns:tns="http://
xmlns.oracle.com/adf/svc/errors/">798</PriceRequestInternal:HeaderCurrencyExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyUnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/
adf/svc/errors/">399</PriceRequestInternal:HeaderCurrencyUnitPrice>
 <PriceRequestInternal:PercentOfComparisonElement>1</PriceRequestInternal:PercentOfComparisonElement>
 <PriceRequestInternal:PriceElementCode>QP_NET_PRICE</PriceRequestInternal:PriceElementCode>
 <PriceRequestInternal:PriceElementUsageCode>NET_PRICE</PriceRequestInternal:PriceElementUsageCode>
 <PriceRequestInternal:RollupFlag>false</PriceRequestInternal:RollupFlag>
 <PriceRequestInternal:SequenceNumber>1002</PriceRequestInternal:SequenceNumber>
 <PriceRequestInternal:TaxIncludedFlag>false</PriceRequestInternal:TaxIncludedFlag>
 <PriceRequestInternal:UnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">399</PriceRequestInternal:UnitPrice>
 </PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:AbsoluteExtendedAmount>798</PriceRequestInternal:AbsoluteExtendedAmount>
 <PriceRequestInternal:SkipRunningPrice>true</PriceRequestInternal:SkipRunningPrice>
 <PriceRequestInternal:ChargeComponentId>4</PriceRequestInternal:ChargeComponentId>
 <PriceRequestInternal:ChargeId>1</PriceRequestInternal:ChargeId>
 <PriceRequestInternal:CurrencyCode>USD</PriceRequestInternal:CurrencyCode>
 <PriceRequestInternal:ExtendedAmount currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">798</PriceRequestInternal:ExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyCode>USD</PriceRequestInternal:HeaderCurrencyCode>
 <PriceRequestInternal:HeaderCurrencyExtendedAmount currencyCode="USD" xmlns:tns="http://
xmlns.oracle.com/adf/svc/errors/">798</PriceRequestInternal:HeaderCurrencyExtendedAmount>
 <PriceRequestInternal:HeaderCurrencyUnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/
adf/svc/errors/">399</PriceRequestInternal:HeaderCurrencyUnitPrice>
 <PriceRequestInternal:PercentOfComparisonElement>1</PriceRequestInternal:PercentOfComparisonElement>
 <PriceRequestInternal:PriceElementCode>QP_MARGIN</PriceRequestInternal:PriceElementCode>
 <PriceRequestInternal:PriceElementUsageCode xsi:nil="true"/>
 <PriceRequestInternal:RollupFlag>false</PriceRequestInternal:RollupFlag>
 <PriceRequestInternal:SequenceNumber>1003</PriceRequestInternal:SequenceNumber>
 <PriceRequestInternal:TaxIncludedFlag>false</PriceRequestInternal:TaxIncludedFlag>
 <PriceRequestInternal:UnitPrice currencyCode="USD" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">399</PriceRequestInternal:UnitPrice>
 </PriceRequestInternal:ChargeComponent>
 <PriceRequestInternal:ChangeSummary logging="false" xmlns:sdo="commonj.sdo"/>
</PriceRequestInternal:PriceRequestInternalType>
```

8. Verify that Last Execution Status displays Succeeded, or a similar success indication.

   If it displays Failure, then click **Show Exception Details** to examine and troubleshoot.

   If you receive this error, then use an XML editor to validate that your code uses the correct XML format.

   ```
   Error: Unable to parse the variable[PriceRequest] using the service
   definition [Sales.PriceRequestInternal]. Please check the variable value or service schema
   ```

9. Click **Save and Close**.
10. On the Manage Algorithms page, click **Actions > Publish**.

Create a Test Sales Order

1. Sign out of Oracle Pricing, then sign into Order Management with the privileges that you need to manage sales orders.
2. Go to the Order Management work area.
3. Enter the value, then step out of the Customer attribute:

Attribute	Value
Customer	Fantastic Laptops

4. Click **Additional Order Details > Pricing and Currency**, then verify the values:

Attribute	Value
Pricing Segment	Corporate Segment
Strategy Name	Pricing Strategy for Fantastic Laptops

5. Add item AS54888, set various attributes, such as Channel, Line of Business, and Ship-to Country, then verify that Pricing applies the discounts that you specified in the price lists and discount lists.
6. Repeat steps 3 through 5 for Computer Service and Rentals.
7. Repeat steps 3 through 5 for PennyPack Systems.

Get Identifiers for Test Payloads

1. Sign into the Order Management work area, then create a sales order:
    - Set the Customer attribute and the Business Unit attribute to values you must test. For this example, set these values:

Attribute	Value
Customer	Fantastic Laptops
Business Unit	Vision Operations

    - Add an order line that includes the item you're testing. For this example, add AS54888.
    - Submit, then notice the sales order number that Order Management creates, such as 12345.
2. Get values for the order header. Run SQL query on the database that your implementation uses to store the sales order that you created in step 1.

```
select header_id, sold_to_party_id as CustomerId, org_id as SellingBusinessUnitId, legal_entity_id as
 SellingLegalEntityId
from doo_headers_all where Order_Number = $12345;
```

   where
    - 12345 is the order number you noted in step 1.
    - The query returns values that you can use in your payload for these header attributes:

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
sold_to_party_id	Header.CustomerId	Header_id equals 100413 and CustomerId equals 1006.

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
org_id	Header.SellingBusinessUnitId	SellingBusinessUnitId equals 204.
legal_entity_id	Header.SellingLegalEntityId	SellinglegalEntityId equals 204.

3. Get values for the order line. Run an SQL query.

```
select inventory_item_id as InventoryItemId, inventory_organization_id as InventoryOrganizationId,
 ordered_uom as LineQuantityUOMCode from doo_fulfill_lines_all where header_id = $56789;
```

where

- 56789 is the header Id in the query result from step 2.
- The query returns values that you can use in your payload for these order line attributes:

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
inventory_item_id	Line.InventoryItemId	Header_id equals 100413 and InventoryItemId equals 149.
inventory_organization_id	Line.InventoryOrganizationId	InventoryOrganizationId equals 204.
ordered_uom	Line.LineQuantity.UOMCode Note that you must replace the UOMCode in two different locations in the payload. In this example, the value is `Ea`.	LineQuantityUOMCode equals USD.

Related Topics

- Pricing Algorithms
- Manage Pricing Algorithms
- Assign Pricing Operations to Pricing Algorithms
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Service Mapping

Extensibles

Use Extensible Flexfields with Pricing

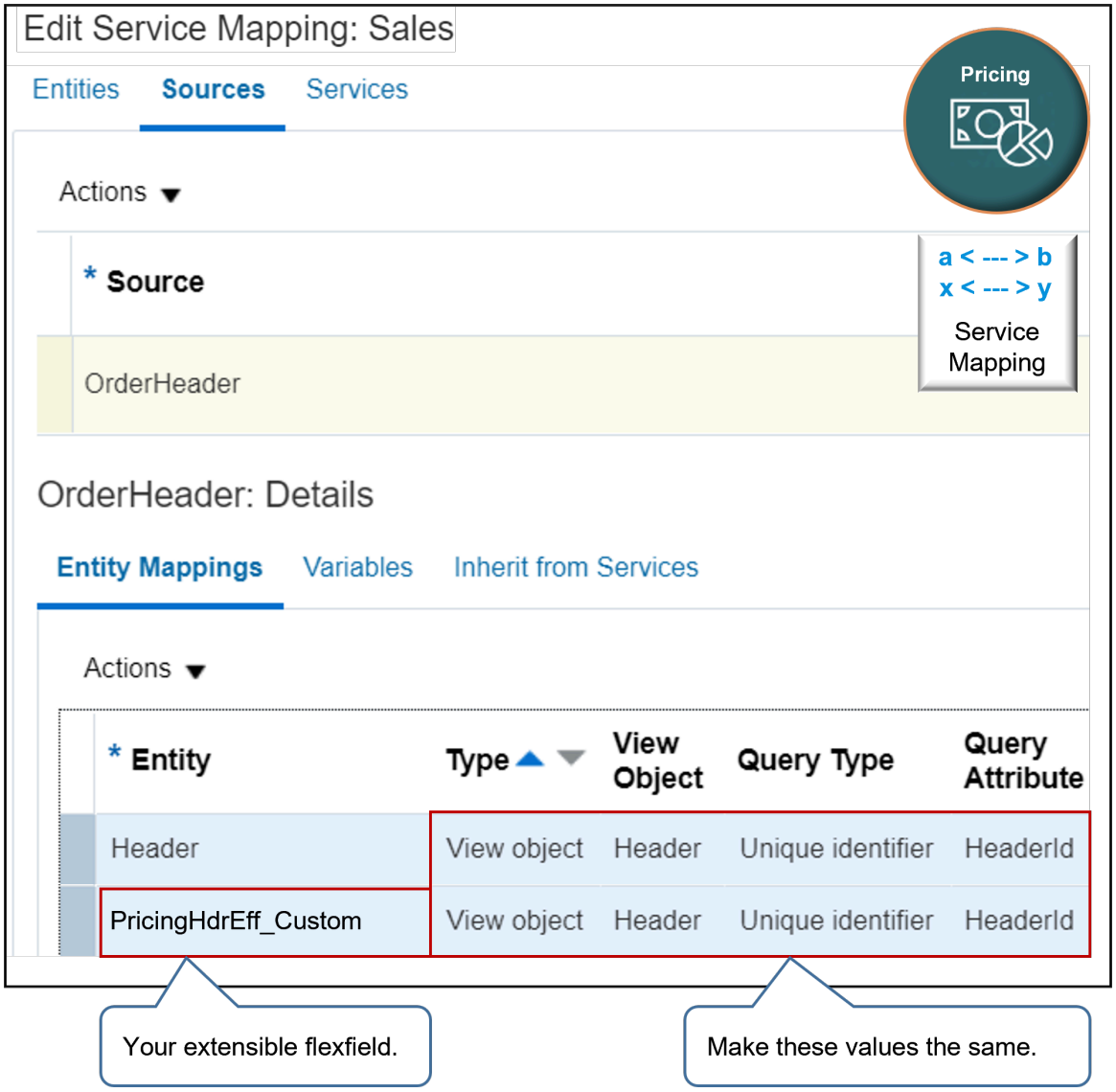
Use an extensible flexfield to modify how Oracle Pricing calculates pricing.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

To use an extensible flexfield for a sales order in a pricing rule, you add an entity to the source in a service mapping to store your own data:

- Select the entity on the source. For example, if you're adding an extensible flexfield to the order header, then select the OrderHeader source.
- Copy the name of the view object from the flexfield archive.
- On the Entity Mappings tab, click **Actions > Add Row**.

- If you add an extensible flexfield on the order header, then make sure all the values you set on the entity that you add for the flexfield match the values that Pricing uses for the predefined Header entity:



- If you add an extensible flexfield on the source for the:
  - Order header, then you must also add it to source for the order line.
  - Order line, then you must also add it to source for the order header.

You will modify Order Management and Pricing so they use the sales channel attribute of an extensible flexfield to determine the pricing strategy.

Here are the tasks that you do:

Task	Description
Manage Service Mappings	Define an extensible flexfield attribute as part of a service mapping so Pricing can use it to identify the pricing strategy it will use to determine the price according to the sales channel.
Manage Matrix Classes	Add an extensible flexfield attribute named Sales Channel as part of a condition in the predefined matrix class that the pricing strategy assignment references.
Manage Algorithms	Add a data set for an extensible flexfield entity to the Get Sales Pricing Strategy pricing algorithm.

Summary of the Set Up

1. Create the value set.
2. Define the extensible flexfield.
3. Administer the service mapping.
4. Modify the matrix class.
5. Modify the pricing algorithm.
6. Create the pricing strategy.
7. Test your set up.

For details, see *Developing Applications with Flexfields*.

Create the Value Set

The Order Entry Specialist enters a value into a flexfield segment while using an Oracle Application. The flexfield validates each segment against the value set that you set up in this topic. To validate a segment means that the flexfield compares the value that you enter into the segment against the values in the value set.

Create the value set:

1. Go to the Setup and Maintenance work area.
2. Click **Tasks > Search**, then open Manage Application Core Value Sets.
3. On the Manage Application Core Value Sets page, click **Actions > Create**, set the values, then click **Save and Close**:

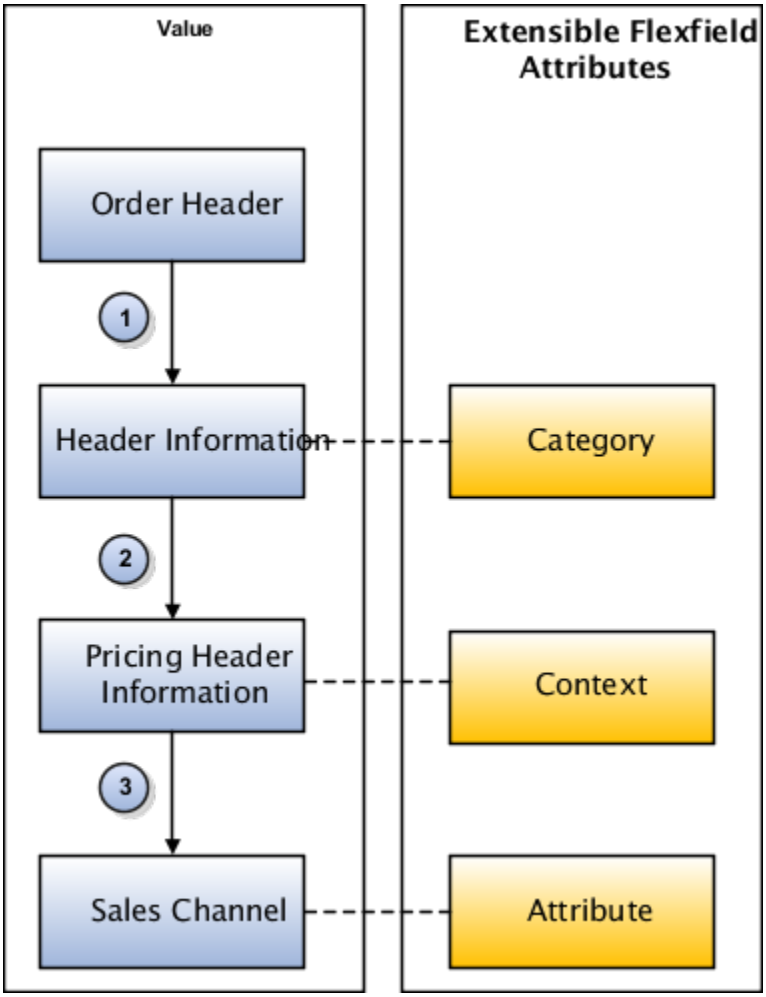
Attribute	Value
Value Set Code	SALES_ORDER_CHANNEL
Description	Set pricing strategy according to sales order channel
Module	Application Flexfields
Validation Type	Table
Value Data Type	Character
From Clause	FND_LOOKUPS

Attribute	Value
Value Column Name	LOOKUP_CODE
Description Column Name	MEANING
ID Column Name	LOOKUP_CODE
WHERE Clause	LOOKUP_TYPE='ORA_QP_CHANNEL_METHODS'
ORDER BY Clause	DISPLAY_SEQUENCE

4. On the Manage Applications Core Value Sets page, Click **Save and Close**.

Define the Extensible Flexfield

Here's the hierarchy you set up for your extensible flexfield:



Note

Reference	Description
1. The Order Header references the Header Information flexfield category.	Not applicable
2. The Header Information flexfield category references the Pricing Header Information flexfield context.	A category is an organizing structure that you use to dynamically display different sets of pages and contexts at run time. In this example, you create a single category for a flexfield so Order Management displays the same extensible flexfield context for each sales channel.
3. The Pricing Header Information flexfield context references the Sales Channel attribute.	An extensible flexfield context is a group of attributes that you use to arrange segments into meaningful groups. For example, the Order Management work area displays each context in the same area on a page at run time.

Create the context for the extensible flexfield:

1. On the Search page, search for, then open Manage Extensible Flexfields.
2. On the Manage Extensible Flexfields page, in the Search area, enter the value, then click **Search**:

Attribute	Value
Name	Header Information

3. In the search results, click **Actions > Edit**.
4. On the Edit Extensible Flexfield page, click **Manage Contexts**.
5. On the Manage Contexts page, click **Actions > Create**.
6. On the Create Contexts page, click **Actions > Create**, then set the values:

Attribute	Value
Display Name	Pricing Header Information
Code	PricingHeaderInformation
API Name	`Pricingheaderinformation`
Enabled	Contains a check mark
Behavior	Single Row

7. On the Context Usages tab, click **Actions > Create**, set the values, then click **Save**:

Attribute	Value
Name	Additional Header Information
View Privileges	None
Edit Privileges	None

8. In the Context Sensitive Segments area, click **Actions > Create**.

9. On the Create Segment page, set values, then click **Save and Close**:

Attribute	Value
Name	SalesChannel
Code	SalesChannel
API Name	`saleschannel`
Enabled	Contains a check mark
Data Type	Character
Table Column	ATTRIBUTE_CHAR1
Value Set	SALES_ORDER_CHANNEL
Prompt	Sales Channel
Display Type	List of Values

10. On the Edit Context page, click **Save and Close**.
11. On the Manage Contexts page, click **Save and Close**.

Associate the context with the category:

1. On the Edit Extensible Flexfield page, in the Category area, click the **row** that contains the value:

Attribute	Value
Display Name	Additional Header Information

2. On the Associated Contexts tab, click **Actions > Select and Add**.
3. In the Select and Add dialog, search for the value:

Attribute	Value
Name	Pricing Header Information

4. In the search results, click the **row**, click **Apply**, then click **OK**.

5. In the Associated Contexts tab, notice that the Associated Contexts tab now displays a row for Pricing Header Information, which means you successfully assigned the Pricing Header Information context to the Additional Header Information category.
6. Click **Save**.

Specify where to display the extensible flexfield:

1. In the Category Details area, click **Pages**, then click **Actions > Create**.
2. In the Create Page dialog, set the values, then click **OK**:

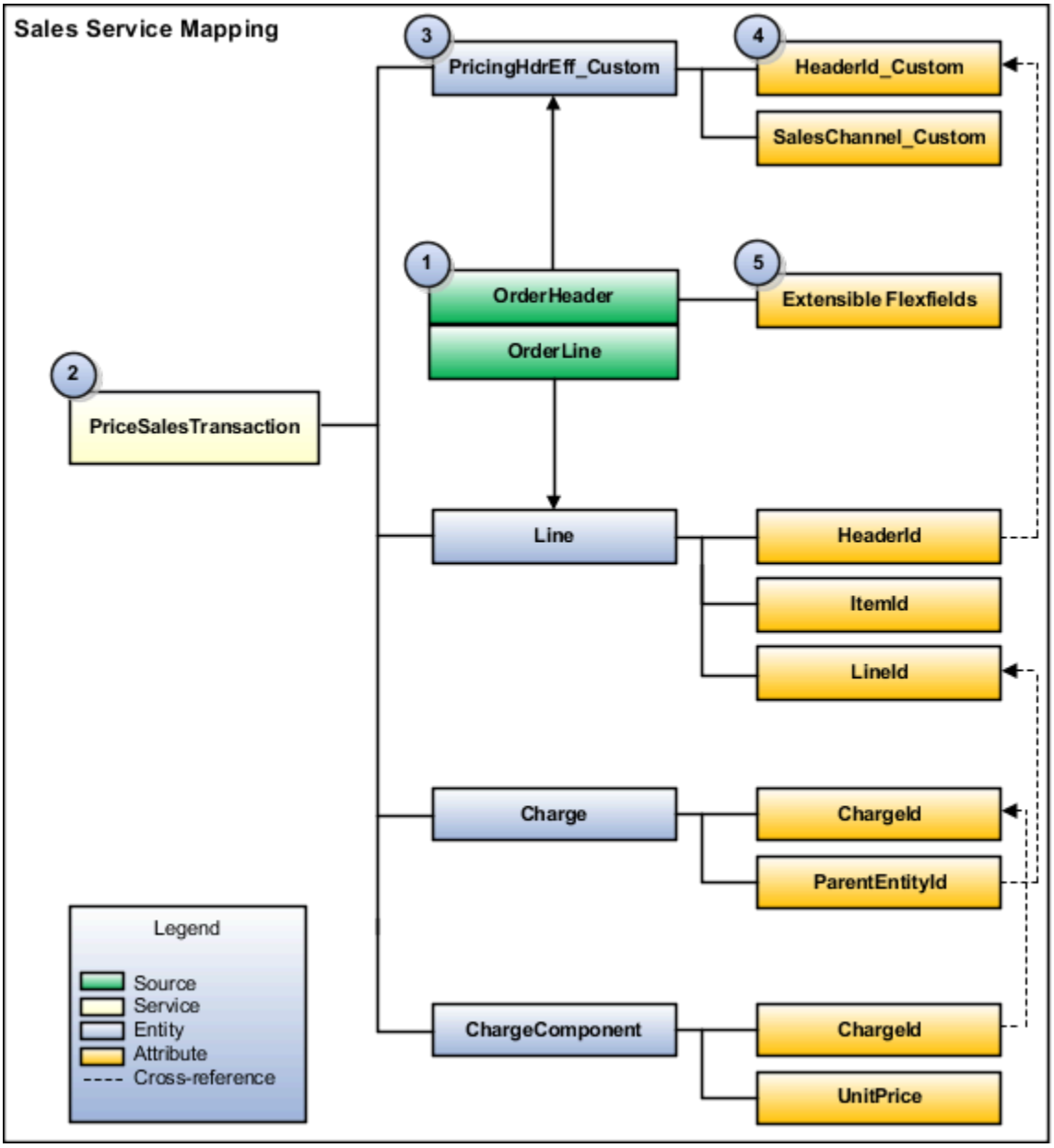
Attribute	Value
Display Name	Pricing Header Information
Code	PricingHeaderInformation
Usage	Additional Header Information

   This step assign the contexts to the page in the Order Management work area that will display the extensible flexfield.
3. On the Edit Extensible Flexfield page, in the Category Details area, click the **row** that contains Pricing Header Information in the Display Name column.
4. In the Pricing Header Information - Associated Contexts Details area, click **Actions > Select and Add**.
5. In the Select and Add dialog, search for Pricing Header Information, wait for the search to finish, click the result **row**, click **Apply**, then click **OK**.
6. Click **Save and Close**.
7. On the Manage Extensible Flexfields page, click **Deploy Flexfield**, and then use the Processing dialog that displays to monitor the deployment.

   Deploy makes the extensible flexfield available in the Order Management work area.
8. Click **Done**, then click **Done** again.

Administer the Service Mapping

Administer the service mapping that implements this SDO:



Here are the objects that this SDO contains:

Object	Your Work in This Example
1. Source. Provides a structure so Pricing can model data in the input SDO.	You specify the OrderHeader source on the predefined Sales service mapping. You map this source to the PricingHdrEff_Custom entity, then map the SalesChannel_Custom and HeaderId_Custom attributes to this entity.
2. Service. Requests the service mapping and receives the output SDO.	The PriceSalesTransaction service references the entities and attributes that the pricing algorithm uses to calculate the item price of each sales order.

Object	Your Work in This Example
3. Entity. References the entities that the service mapping requires to structure the output SDO.	You add the PricingHdrEff_Custom entity to the Sales service mapping.
4. Attribute. References the attributes that the service mapping requires to structure the output SDO.	You add the HeaderId_Custom attribute and SalesChannel_Custom attribute.
5. Extensible flexfields. Contains the data that this example uses.	You map the extensible flexfields that you created earlier in this topic. You map them as attributes of the PricingHdrEff_Custom entity on the OrderHeader source.

Learn about SDOs. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

Add the header entity to the service mapping:

1. Make sure you have the privileges that you need to administer pricing.
2. Go to **Home > Order Management > Pricing Administration > Manage Service Mappings** under Pricing Configuration.
3. On the Manage Service Mappings page, in the Name column, click **Sales**.
4. On the Entities tab, click **Action > Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	PricingHdrEff_Custom
Description	Entity for the flexfield on the order header. This modification supports the business flow that uses a sales channel.

5. In the Details area, add the attributes, then click **Save**:

Attribute	Type	Allow Null
HeaderId_Custom	Long	Contains a check mark.
SalesChannel_Custom	String	Contains a check mark.

Specify the service:

1. Click **Services**, then click the **row** that contains this value:

Attribute	Value
Service	PriceSalesTransaction

Attribute	Value

2. In the Details area, on the Entities tab, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	PricingHdrEff_Custom
Read	Contains a check mark.
Write	Does not contain a check mark.

3. In the PricingHdrEff_Custom Entities area, add these attributes:

Attribute	Read	Write
HeaderId_Custom	Contains a check mark.	Doesn't contain a check mark.
SalesChannel_Custom	Contains a check mark.	Doesn't contain a check mark.

Specify the sources:

1. Click **Sources**, then click the **row** that contains this value:

Attribute	Value
Source	OrderHeader

2. In the OrderHeader Details area, click **Actions > Add Row**, set the \values, then click **Save**:

Attribute	Value
Entity	PricingHdrEff_Custom
Type	View Object
View Object	PricingHeaderInformation

Attribute	Value
	Make sure the value you enter for View Object matches the value from the Code attribute on the Create Context page you used when you created the extensible flexfield earlier in this topic.
Query Type	Unique Identifier
Query Attribute	HeaderId Make sure the value you enter for Query Attribute matches the name you used earlier in this topic when you defined the entity for the extensible flexfield.

3. In the PricingHdrEff_Custom Details area, add the attributes:

Attribute	View Object Value
HeaderId_Custom	HeaderId Make sure the value you enter matches the name you used earlier in this topic when you defined the entity for the extensible flexfield.
SalesChannel_Custom	`Saleschannel` Make sure the value you enter matches the value of the API Name attribute on the Create Segment page that you used when you defined the extensible flexfield earlier in this topic.

Modify the Matrix Class

Modify the matrix class so it uses the SalesChannel_Custom attribute of the PricingHdrEff_Custom entity when it determines whether the condition is true. For details, see *Pricing Matrix Class*:

1. On the Pricing Administration page, click **Tasks > Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click the **link** that contains this value:

Attribute	Value
Name	Sales Pricing Strategy Assignment

3. On the Edit Matrix Class page, in the Condition Columns area, in the Channel Method row, set the values:

Attribute	Value
Name	Channel Method

Attribute	Value
Source Code Name	SaleschannelCode
Comparison	=
Compare to Attribute	PricingHdrEff_Custom.SalesChannel_Custom
Required	Does not contain a check mark.
Allow null	Contains a check mark.
Null is Wild card	Contains a check mark.
Domain	Lookup: ORA_QP_CHANNEL_METHODS

Modify the Pricing Algorithm

Modify the predefined pricing algorithm so it constrains the set of records that it processes to the PricingHdrEff_Custom entity. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*:

1. On the Pricing Administration page, click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, click the **row** that contains the value, then click **Actions > Create Version**:

Attribute	Value
Name	Get Sales Pricing Strategy

3. In the Name column, click **Get Sales Pricing Strategy**.
4. On the Edit Algorithm page, on the Algorithm tab, expand **step 13, Retrieve Header Strategy**, then click **step 14, Get Header Strategy**.
5. In the Data Sets area, click **Add Row**, set the values, then click **Save and Close**:

Attribute	Value
Name	PricingHdrEff_Custom
Variable Path	PriceRequest.PricingHdrEff_Custom
Cardinality	Zero or one

Attribute	Value
Primary	Doesn't contain a check mark.
Data Set Join	`[HeaderId_Custom:{Header.HeaderId}]`

6. On the Manage Algorithms page, click **Actions > Publish**.

Create the Pricing Strategy

1. On the Pricing Administration page, click **Tasks > Manage Pricing Strategies**.
2. On the Manage Pricing Strategies page, click **Actions > Create**.
3. In the Create Pricing Strategy dialog, set the values, then click **Save and Edit**:

Attribute	Value
Name	Corporate Pricing Strategy Sales Channel
Description	Corporate strategy for sales channels.
Business Unit	Vision Operations
Default Currency	USD US Dollar
Default GL Conversion Type	Corporate
Allow Price List Override	Contains a check mark.
Allow Currency Override	Contains a check mark.

4. Add a price list.
   For details, see *Manage Pricing Strategies*.
5. Click **Approve > Save and Close**.
6. Click **Tasks > Manage Pricing Strategy Assignments**.
7. On the Manage Pricing Strategy Assignments page, immediately below the page title, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Assignment Level	Header

Attribute	Value
Pricing Context	Sales
Transaction Type	All

8. Click **Create Assignment Matrix**.
9. In the Create Assignment Matrix dialog, add a check mark to each of these condition columns:
    - Channel Method
    - Pricing Segment
    - Transaction Type
10. Click **OK**.
11. In the Pricing Strategy Assignment Rules area, click **Actions > Add Row**, set the values, then click **Save and Close**. Leave Pricing Segment and Transaction Type empty:

Attribute	Value
Channel Method	Inside Sales
Pricing Strategy	Corporate Pricing Strategy, Sales Channel
Precedence	100

Test Your Set Up

1. Make sure you have the privileges that you need to manage sales orders.

   Go to the Order Management work area and create a new sales order.
2. Verify that you can adjust the sales channel in the order header.
3. Adjust the sales channel, and then verify that the pricing changes depending on channel.

Related Topics

- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Pricing Matrices
- Pricing Algorithms
- Manage Pricing Strategies

Use Your Own Attributes in Pricing

Add your own attribute to an item, then reference it in a pricing rule that determines the price or discount.

An item attribute describes some aspect of item. For example, assume you have been selling an item named My Favorite Shirt for five years, one of your best sellers. My Favorite Shirt already exists in the Oracle database, and it includes attributes Size, Color, and Texture. Size is an example of an item attribute.

Through market research, you learn your customers would love a shirt that allows more or less airflow through the fabric according to the ambient air temperature. So you develop a fabric that increases permeability when the air is above 50 degrees, and decreases permeability when the air is below 50 degrees. You need a new item attribute. You can create an **item extensible attribute** named Permeability to meet this need.

Oracle Applications uses the term **extensible** to describe a capability that enables you to modify Oracle Applications in a way that meets your specific requirement. An item extensible attribute is an attribute that you specify to store details about an item when the details are unique to your business requirement.

You might need to calculate or adjust price according to the attributes of the item, but it might be too cumbersome to set up these calculations for each item. For example, to discount price according to item color or size, or to modify the base price according to the material, such as silver, copper, gold, diamond, Californium, and so on.

Use the Setup and Maintenance work area to set up an item extensible attribute, then use a matrix class that references details about the item instead of defining the attribute for each item:

- Use the Price Sales Transaction pricing algorithm to add this logic.
- Set up and run a pricing rule on a price list or discount list according to the item attributes in a pricing matrix.
- Use an item extensible attribute in the condition column of a matrix class:
    - Use the Price List Charge Adjustment matrix class for a price list.
    - Use the Pricing Term Adjustment matrix class for a discount list.

Related Topics

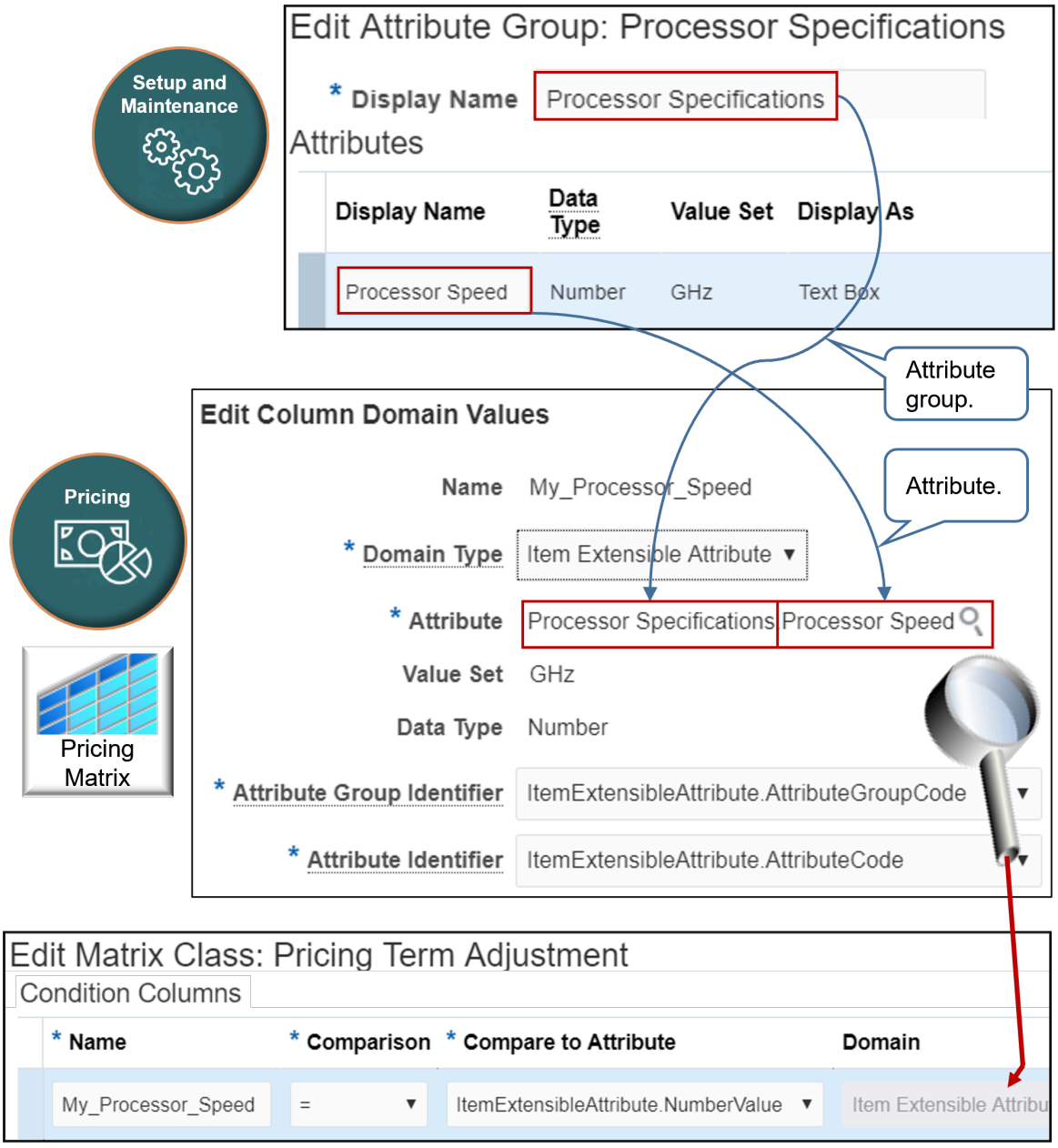
- How Profiles, Segments, and Strategies Work Together
- How Service Mappings, Pricing Algorithms, and Matrixes Work Together
- Roadmap to Manage Oracle Pricing
- Add Your Own Attributes to Items in Pricing

Add Your Own Attributes to Items in Pricing

Set up item extensible attributes on an item class to represent details about the item, then reference them in a pricing rule that determines the price or discount.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Vision Corporation sells a microprocessor item and provides a discount according to processor speed. You can create an item extensible attribute named Processor Speed, and set up a discount for the microprocessors that run at 1.1 GHz:



Note

- Use the Setup and Maintenance work area to create an item extensible attribute named Processor Speed, and to add it to the Processor Specifications attribute group.
- Use the Pricing Administration work area to create a pricing term adjustment that references the Processor Speed attribute.
- Use the Domain to reference Processor Speed.

Here's the format that the Domain column uses:

- `attribute_group_identifier.attribute_identifier:attribute_group.attribute_name. = default_value`

where

- `attribute_group_identifier` specifies the attribute of the service data object to use when comparing against the attribute group code.
- `attribute_identifier` specifies the attribute of the service data object to use when comparing against the attribute code.

Note

- AttributeGroupCode specifies to use the Processor Specifications attribute group when comparing against the attribute group code.
- AttributeCode specifies to use the Processor Speed attribute when comparing against the attribute code.

Summary of the Set Up

1. Create the item extensible attribute.
2. Add your attribute group to an item class.
3. Edit the matrix class.
4. Create the discount list.

Create the Item Extensible Attribute

1. Sign into Oracle Pricing with administrative privileges.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Item Attribute Groups and Attributes**.
3. Create the attribute group:
    - On the Manage Attribute Groups page, click **Actions > Create**.

      For details, see *Item Attribute Groups and Attributes*.
    - On the Create Attribute Group page, create an attribute group:

Attribute	Value
Display Name	Processor Specifications
Internal Name	Processor Specifications
API Name	ProcessorSpecifications
Behavior	Single Row
Enabled	Contains a check mark
Flexfield Code	EGO_ITEM_EFF

- On the Context Usages tab, click **Actions > Create**, then set the value. Other values are optional:

Attribute	Value
Name	Item

- Click **Save**

4. Create the attribute:
   - In the Attributes area, click **Actions > Create**.
   - On the Create Attribute page, set the values:

Attribute	Value
Name	Processor Speed
Internal Name	Processor Speed
API Name	ProcessorSpeed
Flexfield Code	EGO_ITEM_EFF
Data Type	Number

   - Click **Create Value Set**.
   - On the Create Value Set page, set the values:

Attribute	Value
Value Set Code	GHz
Description	Gigahertz
Module	Product Model
Validation Type	Independent You must select Independent or Format Only for each item extensible attribute that you plan to use with Pricing.

Attribute	Value
Value Data Type	Number

- Click **Save and Close**.
- In the Display Properties area, set the value, then click **Save and Close**:

Attribute	Value
Display Type	Text Box You must select Text Box or List of Values for each item extensible attribute that you plan to use with Pricing.

5. On the Edit Attribute Groups page, click **Save and Close**.

Add Your Attribute Group to An Item Class

For details, see *Item Classes*:

1. Go to the Setup and Maintenance work area, then go to the Manage Item Classes task.
2. On the Manage Item Classes page, scan the Name column for an appropriate item class, then click the row that includes the class.
   For this example, click **Processors**. You can also use Root Item Class.
3. Click **Actions > Edit**.
4. On the Edit Item Class page, click **Pages and Attribute Groups**.
5. Add the attribute group to the class:
   - Click **Actions > Select and Add**.
   - In the Select and Add dialog, search for Processor Specifications, which is the attribute group you created earlier in this procedure, wait a moment, then click the line in the search results that includes Processor Specifications in the Name column.
   - Click **Apply**, wait until the application moves your result from the dialog to the page, then click **OK**.
6. Add the page to the class.
   Do this step only if Pricing Item Attributes doesn't already display in the Display Name Column:
   - Click **Pages**, click **Actions > Create**.
   - In the Create Page dialog, set values, then click **OK**

Attribute	Value
Display Name	Pricing Item Attributes
Internal Name	Pricing Item Attributes

Attribute	Value
Data Level	Item

7. Add the attribute to the page:
    - Click **Pages**, then click the **row** that contains Pricing Item Attributes in the Name column.
    - In the Attribute Groups area, click **Actions > Select and Add**.
    - In the Select and Add dialog, search for Processor Specifications, wait a moment, then click the line in the search results that includes Processor Specifications in the Name column.
    - Click **Apply**, wait until the application moves your result from the dialog to the page, then click **OK**.
    - Select the row that includes the page you just added. It includes Pricing Item Attributes in the Name Column.
    - Click **Save**.
8. In the Attribute Groups area, add Processor Specifications.
9. Click **Functional Item Pages**, verify values for the Pricing functional area, then click **Save and Close**:

Attribute	Value
Functional Area	Pricing
Usage	Pricing
Item Page	Pricing Item Attributes

Make sure your set up assigns only the attributes that are associated with the attribute group for the Pricing Item Attributes page to the Pricing functional area. Pricing will examine only the attributes from the attribute groups that the Pricing Item Attributes page references.

Edit the Matrix Class

Edit the matrix class so you can reference it from a rule in a discount list. For details, see *Pricing Matrix Class*:

1. Go to the Pricing Administration work area, then click **Manage Matrix Classes**.
2. On the Manage Matrix Classes page, click **Pricing Term Adjustment**.
3. On the Edit Matrix Class page, in the Condition Columns area, click **Actions > Add Row**, then set the values:

Attribute	Value
Name	My_Processor_Speed
Source Code Name	Accept the default value.

Attribute	Value
Comparison	= (equals sign)
Compare to Attribute	Select an attribute. For this example, processor speed is a Number data type, so select **`itemExtensibleAttribute.NumberValue`**. Make sure you select the same data type that you specify when you set up the attribute on the Manage Item Attribute Groups and Attributes page. If these data types don't match, then Pricing can't calculate the price. You can select only one of these values: - itemExtensibleAttribute.AttributeGroupCode - itemExtensibleAttribute.AttributeCode - itemExtensibleAttribute.VarCharValue - itemExtensibleAttribute.NumberValue - itemExtensibleAttribute.DateValue - itemExtensibleAttribute.TimeStampValue

4. Set the domain:
    - In the Domain column, click **Edit Domain**.
    - In the Edit Column Domain Values dialog, set the value:

Attribute	Value
Domain Type	Item Extensible Attribute

    - In the Attribute field, search for:

Attribute	Value
Attribute Name	Processor Pricing uses the attribute you specify in the input SDO (service data object) of the service mapping. For details, see *How Service Mappings, Pricing Algorithms, and Matrixes Work Together*.

Attribute	Value

Note

- The search gets the attributes that it displays from the Manage Item Attribute Groups and Attributes page. You can search on any text as long as it matches the attributes that you specify on this page.

  For example, if the Processor Specifications attribute group in the Manage Item Attribute Groups and Attributes page contains the Processor Speed attribute, then you can search Attribute Group for Processor Specifications.
- Only some of the attributes from the Manage Item Attribute Groups and Attributes page are useful for Pricing. For example, it might contain several thousand attributes for processors for use in different contexts. You can use the attribute group to help filter the search results so they're specific to Pricing.

- In the search results, select the attribute that your pricing matrix must reference, such as Processor Speed, then click **OK**.
- Here's the dot notation that the Attribute field uses to indicate your choice:

  - `Attribute_group_name.attribute_name`

  For example:

  - `Processor Specifications.Processor Speed`
- Set the values:

Attribute	Value
Attribute Group Identifier	ItemExtensibleAttribute.AttributeGroupCode
Attribute Identifier	ItemExtensibleAttribute.AttributeCode

- Set the value, then click **OK**:

Attribute	Value
Default Value	1.1

As an option, if you set Default Value, then Pricing will display it on the Discount List page that you administer later in this topic.

Pricing gets the attribute values from that item extensible attribute that you set up. In this example, Processor Speed is a Number data type, so you can enter a numeric value. You might specify another

attribute differently. For example, you might specify Color as a Varchar value, and with a set of values, such as blue, green, and red. So, you would select from a list instead of entering a number.

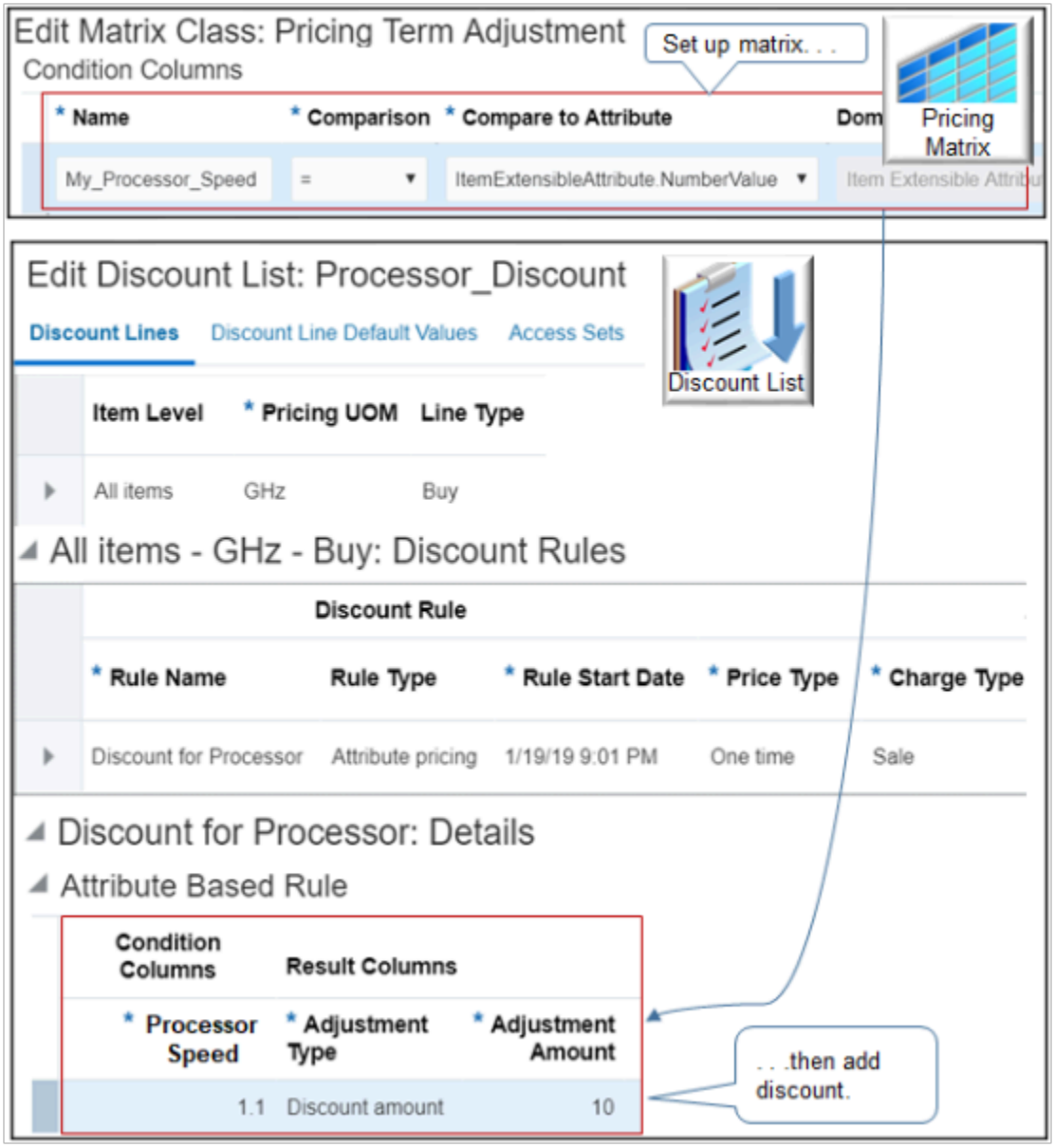
- Set the value, then click **OK**:

Attribute	Value
Default Is Fixed Value	Doesn't contain a check mark. If you enable Default Is Fixed Value, then you can't modify the default value on the Discount List page.

5. On the Edit Matrix Class page, in the Condition Columns area, set the values, then click **OK**:

Attribute	Value
Allow Null	Doesn't contain a check mark. You usually don't enable Allow Null. If you enable it, then the condition column can remain empty when you create the matrix rule. Pricing will evaluate the rule even if the attribute doesn't contain a value.
Null is Wildcard	Doesn't contain a check mark. You usually don't enable Null is Wildcard. If you enable it, then Pricing will accept an empty value when it processes the matrix rule.

Create the Discount List



Set up a discount list so it provides a $10 discount for all 1.1 Ghz processors, to apply only on the sales charge:

1. Create a new discount list:

Attribute	Value
Name	Processor_Discount
Currency	USD
Business Unit	Vision Operations

Attribute	Value

2. In the search results, click **Actions > Add Row**, then set the values:

Attribute	Value
Item Level	All Items
Pricing UOM	GHz
Line Type	Buy

3. Expand **Discount Rules**, then click **Actions > Create > Attribute Based Rule**.
4. In the Create Attribute Based Rule dialog, add a check mark to **Processor Speed**, then click **OK** in the Create Discount Rule dialog, set the values, then click **OK**.
5.

Attribute	Value
Price Type	One Time
Charge Type	Sale
Charge Subtype	Price
Name	Discount for Processor

6. In the Attribute Based rule area, notice that the dialog displays the Condition Column you set up earlier in this topic, then set the values:

Attribute	Value
Processor Speed	1.1 Notice that the attribute defaults to a value of 1.1, which is the value you set up earlier in this topic.
Adjustment Type	Discount Amount
Adjustment Amount	10

7. Click **OK**.

Test Your Setup

1. Sign into Order Management, then create a sales order:

Attribute	Value
Customer	Computer Service and Rentals

2. Add an order line:

Attribute	Value
Item	AS54888 Standard Desktop
Quantity	1

3. On the order line, in the Amount column, click the **value**.
4. Verify that the charge breakdown includes the discount:

Price Component	Amount
Base List Price Applied from Corporate Segments Price List	2,500.00
List Price	2,500.00
Attribute-based discount rule applied because Customer = Computer Service and Rentals	-10.00
Your Price	2490.00
Exclusive Tax (20%)	498.00
Net Price Plus Tax	2988.00

Related Topics

- Pricing Rules
- How Profiles, Segments, and Strategies Work Together
- Use Your Own Attributes in Pricing
- Item Classes

Adjustments

Adjust Price According to Predefined Attributes

Set up Oracle Pricing so it adjusts price according to modifications you make to a predefined attribute.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Free On Board specifies the point during transit where the obligation, cost, and risk involved in shipping transfers from seller to buyer. Assume you maintain a sales agreement with customer Computer Service and Rentals to ship through your preferred distribution depot, and to offer a $100 discount when using this depot as free-on-board point.

You will add a Preferred Depot value to the list of values for the predefined FOB (Free On Board) attribute. The Order Management work area displays this attribute in the sales order's shipment details. You will set up Pricing so it discounts $100 from the base list price when the Order Entry Specialist sets FOB to Preferred Depot.

1. Set up the lookup.
2. Set up the service mapping.
3. Create the pricing basis.
4. Set up a unique lookup.
5. Modify the price list.
6. Test your set up.

Set Up the Lookup

1. Sign into Order Management with administrative privileges.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage FOB Lookup**.
3. On the Manage FOB Lookup page, in the Standard Lookup Type area, click the **row** that includes the value:

Attribute	Value
Lookup Type	FOB

4. In the Lookup Codes area, click **Actions,**, click **New**, set the values, click **Save and Close**, and then click **Done**:

Attribute	Value
Lookup Code	Preferred Depot
Meaning	Preferred Depot
Description	Transfer ownership at the distribution depot

Set Up the Service Mapping

Set up the service mapping:

1. Sign out of Order Management, and then sign back in as the pricing manager.
2. Go to the Pricing Administration work area, then click **Manage Service Mappings** under Pricing Configuration.
3. On the Manage Service Mappings page, click the **row** that includes the value:

Attribute	Value
Name	Sales

4. Map the attribute to the order header:
    - On the Edit Service Mapping page, click **Sources**, then click the **row** that includes the value:

Attribute	Value
Source	OrderHeader

    - In the OrderHeader Details area, on the Entity Mappings tab, click the **row** that includes the value:

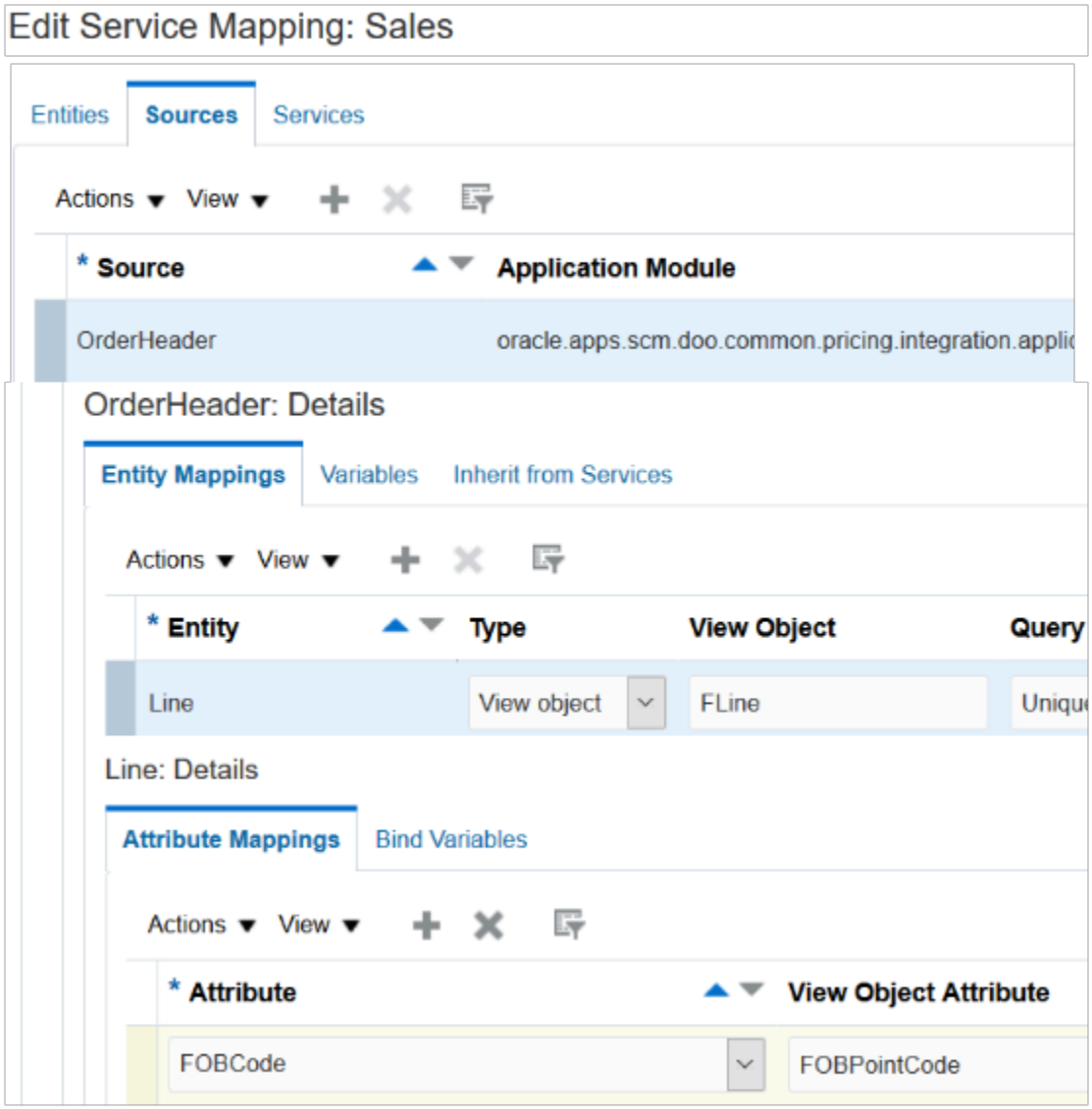
Attribute	Value
Entity	Line

    - In the Line Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FOBCode

Attribute	Value
View Object Attribute	FOBPointCode

For example:



5. Map the attribute to the order line:
    - On the Sources tab, click the **row** that includes the value:

Attribute	Value
Source	OrderLine

Attribute	Value

- In the OrderLine Details area, on the Entity Mappings tab, click the **row** that includes the value:

Attribute	Value
Entity	Line

- In the Line Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FOBCode
View Object Attribute	FOBPointCode

- Click **Save and Close**.

Create the Pricing Basis

1. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search >** Manage Pricing Bases.
2. On the Manage Pricing Bases page, click **Actions > Create**.
3. On the Create Pricing Basis page, set the values, then click **Save and Close**:

Attribute	Value
Name	Adjustment Basis for Base List Price
Usage	Adjustment Basis
Price Element	Base List Price
Description	Adjustment basis for base list price
Active	Contains a check mark

Set Up a Unique Lookup

1. Sign out of Order Management, and then sign back in with order administrator privileges.
2. Navigate to the Order Management work area, then create a sales order:
    - Set the FOB attribute in the sales order's shipment details to Depot.
    - Submit.
3. Run SQL (Structured Query Language) on your Oracle database.

```
SELECT DHA.source_order_number,
 DFLA.FOB_Point_Code
FROM doo_headers_all DHA,
 doo_lines_all DLA ,
 doo_fulfill_lines_all DFLA
WHERE DHA.header_id = DLA.header_id
 AND DLA.Header_id = DFLA.header_ID
 AND DLA.Line_id = DFLA.line_ID
 AND DHA.Source_order_number = '&Source_Order_Number';
```

   This query gets the numeric value stored in the FOB_Point_Code attribute of the doo_fulfill_lines_all table. You will use this value when you modify the price adjustment matrix.

   To get details about how to run this query, see *Use SQL to Query Order Management Data*.
4. Go to the Setup and Maintenance work area, then go to the Manage Order Lookups task.
5. On the Manage Order Lookups page, search for the value:

Attribute	Value
Lookup Type	FOB%

6. In the Lookup Codes area, click **Actions > New**, set the values, click **Save and Close**, then click **Done**:

Attribute	Value
Lookup Code	Enter the value that your SQL query returned, such as 11251.
Meaning	Depot
Description	Transfer ownership at the distribution depot

Modify the Price List

1. Open another browser window, then sign in as the pricing administrator.
2. Go to **Home > Order Management > Pricing Administration**.
3. Click **Price Lists**.

4. On the Price Lists page, search for the value. This topic assumes you already created this price list. For details, see *Manage Price Lists*:

Attribute	Value
Name	Price List for Computer Service and Rentals

5. Search for the value:

Attribute	Value
Item	AS54888

6. Verify the values on the price list's line:

Attribute	Value
Base Price	2500
Allow Manual Adjustments	Contains a check mark

7. Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
8. Add the FOB Preferred Depot column.
9. Click **Add Row**, then add one:

Attribute	Value
Adjustment Type	Discount Amount
Adjustment Amount	100
Adjustment Basis	Adjustment Basis for Base Price List
FOB Depot	FOB Preferred Depot

10. Click **Save > Activate**.

Test Your Set Up

1. Sign out of Pricing, then sign into Order Management with the privileges that you need to manage sales orders.
2. Create a sales order. Use these values:

Attribute	Value
Customer	Computer Service and Rentals
Item	AS54888
Quantity	1

3. Click **Save**, then verify that the order header displays this value:

Attribute	Value
Total	$2,500

4. Set this value:

Attribute	Value
FOB	FOB Depot To access this attribute, click Shipment Details, and then click Shipping.

5. Click **Save**, then verify that the order header displays these values:

Attribute	Value
Total on the order header	$2,400 Notice that the value is $100 less than the value you examined before you set the FOB attribute.
Amount on the order line	$2,400

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Assign Pricing Strategy
- Manage Price Lists
- Use SQL to Query Order Management Data

Adjust Price According to FOB Attribute

Set up pricing so it adjusts price according to the FOB attribute on the order line.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

FOB (free on board) indicates you won't charge your customer for shipping costs you incur to get them item from your warehouse or factory to the FOB location.

Assume:

- The customer is Computer Service and Rentals.
- The item is the AS54888.
- Your factory is in `Culver City`, a suburb of Los Angeles, but you ship from the Port of Los Angeles.
- You don't charge for shipping you incur to get the AS54888 from `Culver City` to the Port of Los Angeles.
- You do charge for the cost of shipping from the Port of Los Angeles to the ship-to site.
- You offer a $100 discount when you ship the AS54888 out of the Port of Los Angeles.
- You already added the AS54888 to the Corporate Segment Price List, and set the base price to 2500.

Summary of the Set Up

1. Examine the current behavior.
2. Modify the lookup.
3. Set up the service mapping.
4. Create the pricing basis.
5. Create lookup type.
6. Modify the matrix class.
7. Modify the price list.
8. Test your set up.

Examine the Current Behavior

1. Sign into Order Management with administrative privileges.
2. Go to the Order Management work area, then create a sales order:

Attribute	Value
Customer	Computer Service and Rentals
Business Unit	Vision Operations

3. Add the AS54888 item to an order line, then notice the value of Amount on the order line is 2500.

4. Click **Shipment Details > Shipping**, change the value in the FOB attribute, then save your work.

   Order Management reprices the order when you save your work.
5. Click **Lines**, then notice the value of Amount on the order line is still 2500.
6. Notice the sales order number. For this example, assume its 508670.

Modify the Lookup

1. In the Setup and Maintenance work area, click **Search**, search for, then open Manage FOB Lookup.
2. On the Manage FOB Lookup page, in the Standard Lookup Type area, select the row:

Attribute	Value
Lookup Type	FOB
Meaning	Valid FOB Codes

3. In the Lookup Codes area, Click **Actions > New**, set the values, then click **Save and Close**:

Attribute	Value
Lookup Code	FOB_LA
Enabled	Contains a check mark.
Meaning	Port of Los Angeles
Description	Shipping charges start at the Port of Los Angeles.

4. Propagate your lookup across Oracle applications:
   - Go to the Setup and Maintenance work area, then go to the Collect Planning Data task.
   - On the page that displays, verify that Process Name is Collection Job Set, then set the values:

Attribute	Value
Source System	GPR
Collection Type	Net Change

   - On the Reference Data tab, move Order Orchestration Reference Objects from the Reference Entities list to the Selected Entities list.

You use Order Orchestration Reference Objects for any data entity you need for Order Management, such as free on board, return reason, invoicing and accounting rules, payment terms, document categories, sales credit types, receipt methods, shipment priority, payment method, freight charge terms, and so on.

Note that the Supply Planning Data tab includes the Sales Orders supply entity. You use this entity for each sales order you create or import, and scheduling details for these sales orders.

- Submit.
- Wait for one hour for the collection process to finish.

For details, see *Overview of Data Collections for Supply Chain Planning*.

Set Up the Service Mapping

1. Sign out, then sign into Oracle Pricing with administrative privileges.
2. Go to the Pricing Administration work area, then click **Manage Service Mappings**.
3. On the Manage Service Mappings page, open the Sales service mapping.
4. Create the source for the order header:
   - On the Edit Service Mappings page, click Sources, then click the row:

Attribute	Value
Source	OrderHeader

   - In the Details area, on the Entity Mappings tab, click the row:

Attribute	Value
Entity	Line
Type	View Object
View Object	Fline
Query Type	Unique Identifier
Query Attribute	HeaderId

   - In the Line Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FOB Code

Attribute	Value
View Object Attribute	FobPointCode

- In the Line Details area, click **Sources**, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FOB Code
View Object Attribute	FobPointCode

5. Create the source for the order line:
   - In the Sources list, click the **row**:

Attribute	Value
Source	OrderLine

   - In the Details area, on the Entity Mappings tab, click the **row**:

Attribute	Value
Entity	Line
Type	View Object
View Object	Fline
Query Type	Unique Identifier
Query Attribute	HeaderId

   - In the Line Details area, on the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	FOB Code

Attribute	Value
View Object Attribute	FobPointCode

- Click **Save and Close**.

Create the Pricing Basis

1. Sign out of Pricing, then sign into Order Management with administrative privileges.
2. Go to the Setup and Maintenance work area, then go to the Manage Pricing Bases task.
3. On the Manage Pricing Bases page, click **Actions > Create**, set the values, then click **Save and Close**:

Attribute	Value
Name	Adjustment Basis for Base Price List
Usage	Adjustment Basis
Price Element	Base List Price
Active	Contains a check mark.

Create the Lookup Type

1. Go to the Order Management work area, then open the sales order you created earlier. In this example, open order 508670.
2. On the order line, set the FOB attribute to Port of Los Angeles, then save your work.
3. Use SQL to query the Oracle database.

```
SELECT DHA.source_order_number,
 DFLA.FOB_Point_Code
FROM doo_headers_all DHA,
 doo_lines_all DLA ,
 doo_fulfill_lines_all DFLA
WHERE DHA.header_id = DLA.header_id
 AND DLA.Header_id = DFLA.header_ID
 AND DLA.Line_id = DFLA.line_ID
 AND DHA.Source_order_number = '&Source_Order_Number';
```

For details, see *Use SQL to Query Order Management Data*.

Replace Source_Order_Number with your order number. For example:

```
AND DHA.Source_order_number = 508670;
```

The query gets the numeric value that the database stores in the FOB_Point_Code attribute of table doo_fulfill_lines_all. You need this value because the price adjustment matrix uses the internal ID for the FOB attribute.

Assume the query returns the value 11251.

4. In the Setup and Maintenance work area, click **Tasks > Search**, then search for and open Manage Standard Lookups.
5. On the Manage Standard Lookups page, click **Actions > New**, set values, then click **Save**:

Attribute	Adjustment Basis for Base Price List
Name	FOB_MSC_CODE
Meaning	FOB MSC Codes
Description	FOB MSC Codes for price adjustment
Module	Application Common

6. In the Lookup Codes area, click **Actions > New**, set the values, then click **Save and Close**:

Attribute	Adjustment Basis for Base Price List
Lookup Code	11251
Enabled	Contains a check mark.
Meaning	Port of Los Angeles

Modify the Matrix Class

1. Sign out, then sign into Pricing.
2. Click **Tasks > Manage Matrix Classes**.
3. On the Manage Matrix Classes page, open the Price List Charge Adjustment class.
4. In the Condition Columns area, click **Actions > Add Row**, set values, then click **Save and Close**:

Attribute	Value
Name	FOB
Source Code Name	FOB

Attribute	Value
Comparison	=
Compare to Attribute	Line.FOBCode
Domain	Set Domain Type to Lookup Set Lookup to FOB_MSC_CODE

Modify the Price List

1. Click **Price Lists**.
2. On the Price Lists page, search for, then open price list Corporate Segment Price List.
3. On the Price List page, search for the AS54888 item.
4. In the Charge area, verify values:

Attribute	Adjustment Basis for Base Price List
Base Price	2500
Allow Manual Adjustment	Contains a check mark.

5. Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
6. Add the Ordered Quantity column.
7. Click **Add Row**, then add one:

Attribute	Value
FOB	Port of Los Angeles
Adjustment Type	Discount Amount
Adjustment Amount	100
Adjustment Basis	Adjustment Basis for Base Price List Recall that this is the basis you created earlier in this topic. It instructs Pricing to apply the adjustment to the Base List Price.

8. Click **Save**.

Test Your Set Up

1. Open another browser window, then sign into Order Management.
2. Go to the Order Management work area, then open the sales order you created earlier.

   In this example, open order 508670.
3. On the order line, set the FOB attribute to any value other than Port of Los Angeles, then click **Save**.
4. Notice the value of Amount on the order line is 2500.
5. Set the FOB attribute to Port of Los Angeles, then click **Save**.
6. Verify that the value of Amount on the order line is 2400, which is Base Price of 2500 minus discount of 100.

Related Topics

- Overview of Data Collections for Supply Chain Planning
- Use SQL to Query Order Management Data

Override Base List Price

Set up Oracle Pricing so it uses the value from an extensible flexfield to override the base list price.

> **CAUTION:** The example in this topic describes one way to set up pricing. It is intended only as a general outline that you can use to learn about different ways to set up Pricing. You will need to use different procedures, different steps, different values, and different objects for your implementation, depending on your business requirements.

Use this procedure only with a standard item. You can't use it with a coverage item or subscription item.

In this example, you set up an extensible flexfield that displays in the Order Management work area. You then use the value that the Order Entry Specialist enters in the flexfield to override the Sale Price charge.

Summary of the Set Up

1. Set up the extensible flexfield.
2. Modify the pricing algorithm.
3. Test your set up.

Set Up the Extensible Flexfield

1. Make sure you have the privileges that you need to administer Order Management.

   Don't sign in with pricing privileges. You can't use them to set up an extensible flexfield.
2. Go to **Home > My Enterprise > Setup and Maintenance > Tasks > Search > Manage Order Extensible Flexfields**.

   This example assumes you're implementing the Order Management offering. If you're using a different offering, then click it instead of Order Management.

3. Modify the extensible flexfield:
    - On the Manage Order Extensible Flexfields page, enter the values, then click **Search**:

Attribute	Value
Name	Fulfillment Line Information
Flexfield Code	DOO_FULFILL_LINES_ADD_INFO

    - Click **Actions > Edit**.
    - Click **Manage Contexts**.
    - On the Manage Contexts page, click **Actions > Create**.
    - On the Create Context page, set values:

Attribute	Value
Display Name	LinePrcOverride
Code	LinePrcOverride
API Name	Lineprcoverride
Enabled	Contains a check mark
Behavior	Single Row

    - In the Context Usages area, click **Actions > Create**, set the values, then click **Save**:

Attribute	Value
Name	Additional Fulfillment Line Information
View Privileges	None
Edit Privileges	None

    - 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	SalePrcOverrideVal
Code	SalePrcOverrideVal
API Name	saleprcoverrideval
Enabled	Contains a check mark.
Data Type	Number
Table Column	Select any column that's available.
Value Set	15 Digit Number
Prompt	SalePrcOverrideVal

- On the Edit Context page, click **Save and Close**.
- On the Manage Contexts page, click **Save and Close**.
- On the Edit Extensible Flexfield page, click **Save and Close**.

4. Deploy the extensible flexfield:
   - On the Manage Order Extensible Flexfields page, click **Actions > Deploy Flexfield**.
   - Wait for the deployment dialog to indicate that deployment successfully finished, then click **OK**.
   - Click **Actions > Download Flexfield Archive**, wait for the dialog to indicate that the archive successfully finished, then click **Download**.
   - Save file 10008_DOO_FULFILL_LINES_ADD_INFO.zip to your local hard drive, extract it, then navigate to this folder.

     ```
     oracle\apps\scm\doo\processOrder\flex\fulfillLineContextsB\view\
     ```

   - Open FulfillLineEffBLinePrcOverrideprivateVO.xml.

Add Extensible Flexfield to Service Mapping

You set up a service mapping that maps values from the extensible flexfield into Pricing.

Add the extensible flexfield to the service mapping:

1. Sign out, then sign in with the privileges that you need to administer pricing.
2. Go to **Home > Order Management > Pricing Administration > Manage Service Mappings**..

3. On the Manage Service Mappings page, in the Name column, click **Sales**.
4. Define the entities:
    - On the Edit Service Mappings page, on the Entities tab, click **Action > Add Row**, then set the values:

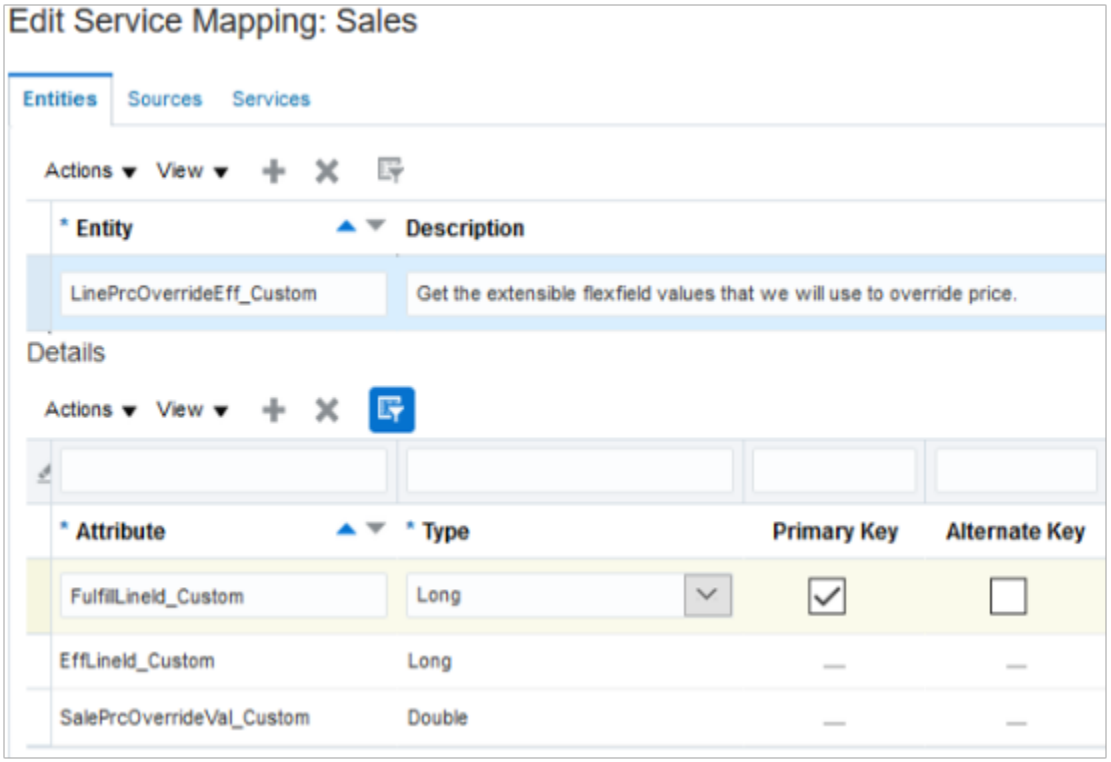
Attribute	Value
Entity	LinePrcOverrideEff_Custom
Description	Get the extensible flexfield values that we will use to override price.

    - In the Details area, add attributes, then click **Save**:

Attribute	Type	Primary Key
FulfillLineId_Custom	Long	Contains a check mark.
EffLineId_Custom	Long	Does not contain a check mark.
SalePrcOverrideVal_Custom	Decimal	Does not contain a check mark.

Attribute	Type	Primary Key

For example:



- Click **Services**, then click the **row** that contains this value:

Attribute	Value
Service	PriceSalesTransaction

- In the Details area, click **Actions > Add Row**, then set these values:

Entity	Read	Write
`LinePrcOverrideEff_Custom`	Contains a check mark.	Does not contain a check mark.

This step creates a relationship between Pricing and the entity that you use to send values from the extensible flexfield.

- In the LinePrcOverrideEff_Custom Entities area, add attributes, then click **Save**:

Attribute	Read	Write
EffLineId_Custom	Contains a check mark.	Does not contain a check mark.
FulfillLineId_Custom	Contains a check mark.	Does not contain a check mark.
SalePrcOverrideVal_Custom	Contains a check mark.	Does not contain a check mark.

5. Set up the source for the order header:
   - Click **Sources**, then click the **row** that includes the value:

Attribute	Value
Source	OrderHeader

   - On the Entity Mapping tab, click **View > Columns**, then add a check mark to Joined Entity and Joined Entity Attribute.
   - Click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	LinePrcOverrideEff_Custom
Type	View Object
View Object	FulfillLineEffBLinePrcOverrideprivateVO Make sure the value you enter matches exactly the value you noticed earlier in this topic when you examined the flexfield archive. Don't include the `.xml` extension.
Query Type	Join
Query Attribute	FulfillLineId
Joined Entity	Line It might be necessary to click another row, and then return to this row to get the joined entity attributes to populate values.

Attribute	Value
Joined Entity Attribute	LineId

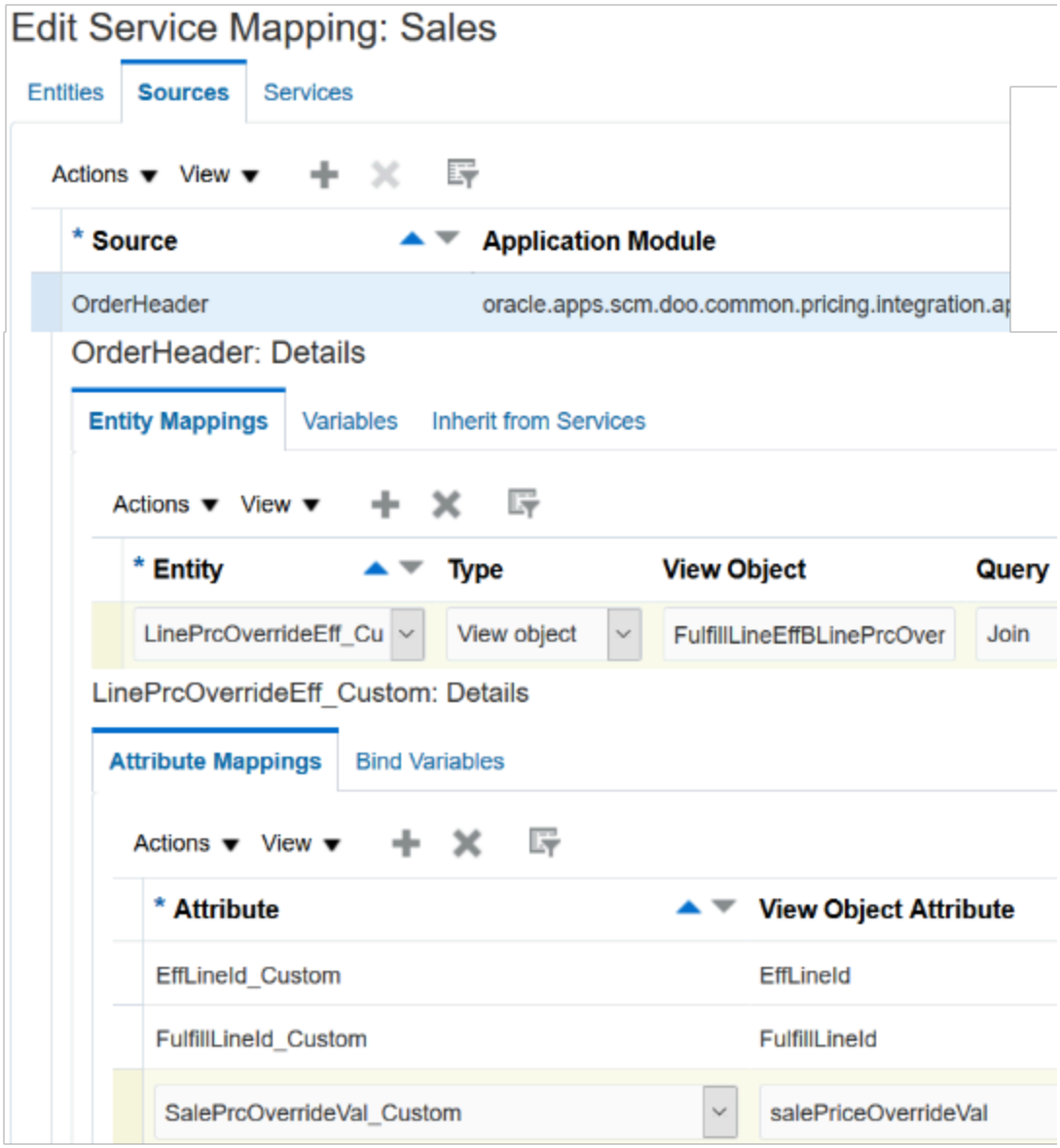
- In Details area LinePrcOverrideEff_Custom, add the attributes, then click **Save**:

Attribute	View Object Attribute
EffLineId_Custom	EffLineId
FulfillLineId_Custom	FulfillLineId
SalePrcOverrideVal_Custom	salePriceOverrideVal Make sure the value you enter matches exactly the value you noticed in tag ViewAttribute Name of the flexfield archive you examined earlier in this topic. For example, `ViewAttribute Name="salePriceOverrideVal"`.

Attribute	View Object Attribute

This step sets up Pricing to read extensible flexfield values from the public view object that Order Management sends to Pricing when the Order Entry Specialist uses the Price Order action or saves the sales order.

For example:



6. Set up the source for the order line:
    - In the Sources area, click the **row** that includes the value:

Attribute	Value
Source	OrderLine

    - On the Entity Mapping tab, click **Actions > Add Row**, set the values, then click **Save**:

Attribute	Value
Entity	LinePrcOverrideEff_Custom
Type	View Object
View Object	FulfillLineEffBLinePrcOverrideprivateVO Make sure the value you enter matches exactly the value you noticed earlier in this topic when you examined the flexfield archive.
Query Type	Join
Query Attribute	FulfillLineId
Joined Entity	Line
Joined Entity Attribute	LineId

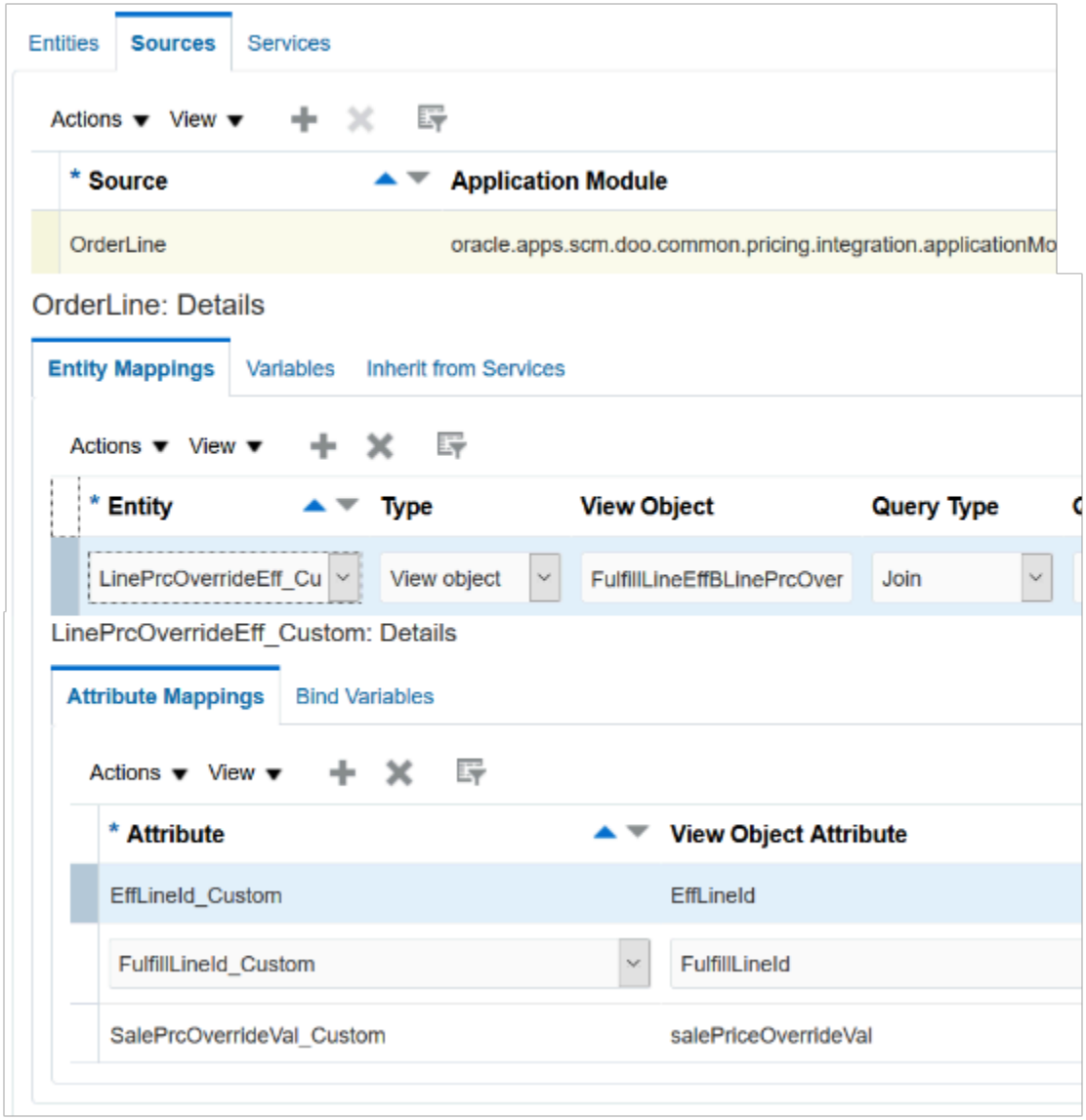
    This step maps the extensible flexfield attributes in the public view object. It sets up Pricing to read extensible flexfield values from the public view object that Order Management sends to Pricing when Order Management updates the line quantity or shipping attributes on the order line.
    - In the Details area for HeaderPricingEff_Custom, add the attributes:

Attribute	View Object Attribute
EffLineId_Custom	EffLineId
FulfillLineId_Custom	FulfillLineId

Attribute	View Object Attribute
SalePrcOverrideVal_Custom	salePriceOverrideVal Make sure the value you enter matches exactly the value you noticed in the ViewAttribute Name tag of the flexfield archive you examined earlier in this topic. For example, `ViewAttribute Name="salePriceOverrideVal"`.

This step sets up Pricing to read extensible flexfield values from the public view object that Order Management sends to Pricing when Order Management updates the quantity attribute or a shipping attribute on the order line.

For example:



7. Click **Save and Close > Done**.

Modify Pricing Algorithm

1. Click **Tasks > Manage Algorithms** under Pricing Configuration.
2. On the Manage Algorithms page, select a published version of this pricing algorithm, such as Version 1:

Attribute	Value
Name	Get Base List Price For Goods And Services

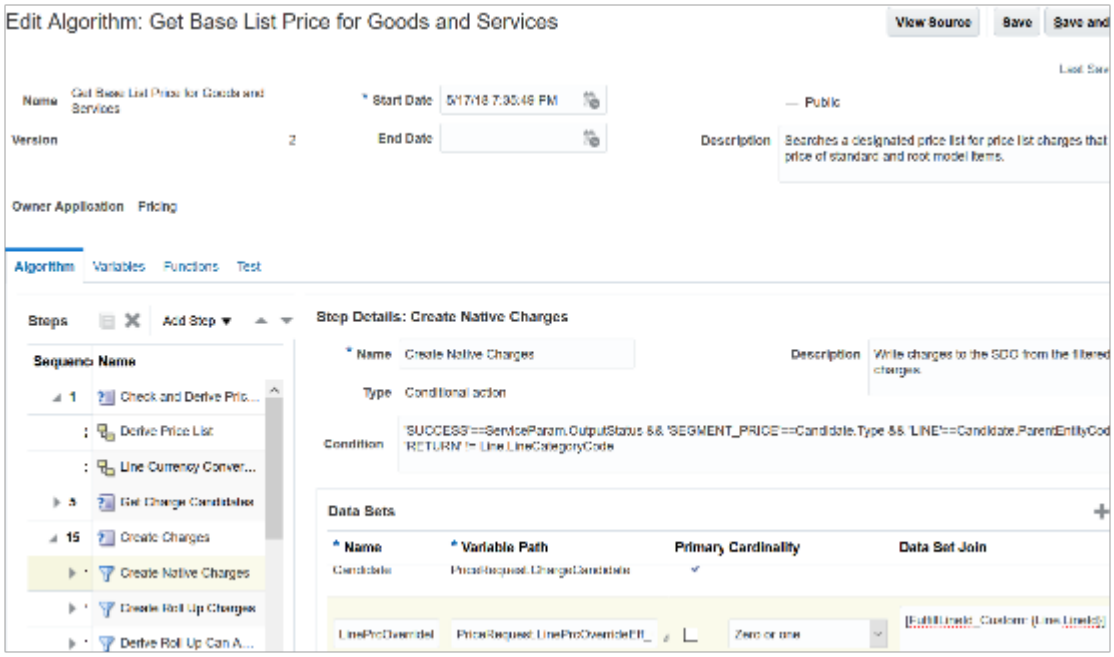
3. Click **Actions > Create Version**, then wait for the page to display your In Progress version, such as Version 2
4. Open your In Progress version for editing.
5. On the Edit Algorithm page, on the Algorithm tab, expand step **Create Charges**, then click step **Create Native Charges**.

   You will modify this step so it references the extensible flexfield that contains the override value.
6. In the Data Sets area, add this data set:

Attribute	Value
Name	LinePrcOverrideEff
Variable Path	PriceRequest.LinePrcOverrideEff_Custom
Cardinality	Zero or one
Data Set Join	[FulfillLineId_Custom: {Line.LineId}]

Attribute	Value

For example:



7. In the Default Action area, in the Actions window, replace the code:

Old Code	New Code

`Ch.CurrencyCode = Line.AppliedCurrencyCode`	`//Ch.CurrencyCode = Line.AppliedCurrencyCode`
	`//Base List Price Override`
	`if (null != LinePrcOverrideEff?.SalePrcOverrideVal_Custom) {`
	`Ch.CurrencyCode = Header.AppliedCurrencyCode`
	`Line.AppliedCurrencyCode = Header.AppliedCurrencyCode`
	`Ch.NeedsCostPlus = false`
	`//clear out the tier and matrix adjustments`
	`Candidate.unset('TieredPricingHeaderId')`
	`Candidate.unset('AttributePricingMatrixId')`
	`}`
	`else {`
	`Ch.CurrencyCode = Line.AppliedCurrencyCode`
	`Ch.NeedsCostPlus = ('COST'==CalculationMethod)`
	`}`

8. Comment this code:

Old Code	New Code
`Ch.NeedsCostPlus = ('COST'==CalculationMethod)`	`//Ch.NeedsCostPlus = ('COST'==CalculationMethod)`

Here's the complete, revised code.

```
Ch = Charge.insert([ChargeId:++ServiceParam.ChargeIdCntr, LineId:Line.LineId,
 ParentEntityId:Candidate.ParentEntityId, ParentEntityCode:'LINE'])
if ( CanAdjustFlag!=null ) Ch.CanAdjustFlag = ('Y'==CanAdjustFlag)
 Ch.ChargeAppliesTo = Candidate.ChargeAppliesTo
 Ch.ChargeDefinitionCode = Candidate.ChargeDefinitionCode
 Ch.ChargeDefinitionId = ChargeDefinitionId
 Ch.ChargeSubtypeCode = ChargeSubtypeCode
 Ch.ChargeTypeCode = ChargeTypeCode
// Set the CompSeqCntr as Long instead of Integer
Ch.CompSeqCntr = 1000L
//Ch.CurrencyCode = Line.AppliedCurrencyCode
//Base List Price Override
if (null != LinePrcOverrideEff?.SalePrcOverrideVal_Custom) {
 Ch.CurrencyCode = Header.AppliedCurrencyCode
 Line.AppliedCurrencyCode = Header.AppliedCurrencyCode
 Ch.NeedsCostPlus = false
 //clear out the tier and matrix adjustments
 Candidate.unset('TieredPricingHeaderId')
 Candidate.unset('AttributePricingMatrixId')
}
else {
 Ch.CurrencyCode = Line.AppliedCurrencyCode
 Ch.NeedsCostPlus = ('COST'==CalculationMethod)
}
Ch.EstimatedPricedQuantityFlag = ('USAGE'==PriceTypeCode || 'RECURRING_USAGE'==PriceTypeCode)
Ch.EstimatedUnitPriceFlag = Ch.EstimatedPricedQuantityFlag
Ch.PrimaryFlag = false
Ch.RollupFlag = false
//Ch.NeedsCostPlus = ('COST'==CalculationMethod)
Ch.NeedsMargin = ('Y'==CalculateMarginFlag)
if (PricePeriodicityCode!=null) Ch.PricePeriodicityCode = PricePeriodicityCode
 Ch.PriceTypeCode = PriceTypeCode
if ( 'USAGE'==PriceTypeCode || 'RECURRING_USAGE'==PriceTypeCode ) {
 // PricedQty should be null if estimated usage quantity is not passed in
 PricedQty = (Ci?.EstimatedUsageQuantity!=null ? (Ci.PerUnitUsageFlag ? (Line.ExtendedQuantity?.Value ?:
 Line.LineQuantity.Value) : 1)*Ci?.EstimatedUsageQuantity?.Value : null)
}
else {
 PricedQty = Line.ExtendedQuantity?.Value ?: Line.LineQuantity.Value
}
if ( PricedQty != null ) {
 Ch.createDataObject('PricedQuantity')
 Ch.PricedQuantity.Value = PricedQty
 Ch.PricedQuantity.UnitCode = PricingUomCode
 Ch.PricedQuantityUOMCode = PricingUomCode
}
Ch.RunningUnitPrice = 0
if (Ci?.UsagePeriodCode!=null) Ch.UsagePeriodCode = Ci.UsagePeriodCode
if (Ch.UsagePeriodCode!=null && Candidate.UsagePeriodCode!=null) Ch.UsagePeriodCode =
 Candidate.UsagePeriodCode
if (UsageUomCode!=null) Ch.UsageUOMCode = UsageUomCode
```

```
 Ch.TaxIncludedFlag = false
 finer(AlgmName+': created native charge '+Ch.ChargeId+' ('+Ch.PriceTypeCode+', '+Ch.ChargeTypeCode+',
 '+Ch.ChargeSubtypeCode+') for line '+Line.LineId)

// Needed to write charge component
Candidate.ChargeId = Ch.ChargeId

// Initialize for tiered pricing
if (Candidate.TieredPricingHeaderId != null) {
 TierQ.insert([TieredPricingHeaderId:Candidate.TieredPricingHeaderId, ChargeId:Ch.ChargeId,
 OriginId:Line.LineId, OriginType:'LINE'])
}

// Initialize for attribute-based pricing
if (Candidate.AttributePricingMatrixId != null) {
 MatrixQ.insert([DynamicMatrixId:Candidate.AttributePricingMatrixId, ParentEntityId:Ch.ChargeId,
 FromCurrencyCode:Ch.CurrencyCode, ParentEntityCode:'CHARGE'])
}

// Coverage Duration being calculated if its a Phantom Line
if ('COVERED_STANDARD' == Line.ItemType && Line.PhantomFlag)
{
 if (CoverageAssociation != null)
 {
 Ch.createDataObject('CoverageDuration')
 Ch.CoverageDuration.Value = CoverageAssociation.CoverageDuration.Value
 Ch.CoverageDuration.UnitCode = CoverageAssociation.CoverageDuration.UnitCode
 Ch.CoverageDurationUOMCode = CoverageAssociation.CoverageDurationUOMCode
 }
 Ch.LineId = CoverageLine.LineId
}
```

9. Click the **Create Charge Components** step.

   You will modify this step so it references the extensible flexfield that contains the override value, then use this value to calculate the unit price.
10. In the Data Sets area, add this data set:

Attribute	Value
Name	LinePrcOverrideEff
Variable Path	PriceRequest.LinePrcOverrideEff_Custom
Cardinality	Zero or one
Data Set Join	[FulfillLineId_Custom: {Line.LineId}]

11. In the Default Action area, locate this code.

```
Comp.createDataObject('UnitPrice')
```

12. Add this code immediately under the code you just located.

```
//Base List Price Override
if(Candidate.CalculationMethod == 'COVERED_ITEM_PRICE_PERCENT') {
```

```
 fine({'UnitPrice Calc - Cand.CalculationAmount: '+Candidate.CalculationAmount+' CoverageBasis:
 '+Candidate.CoverageBasisValue})
 Comp.UnitPrice.Value = Candidate.CalculationAmount * Candidate.CoverageBasisValue/ 100
 Ch.HasPercentPriceFlag = true
}
else {
 //Base List Price Override
 if (LinePrcOverrideEff?.SalePrcOverrideVal_Custom != null)
 Comp.UnitPrice.Value = LinePrcOverrideEff.SalePrcOverrideVal_Custom
 else
 Comp.UnitPrice.Value = Candidate.BasePrice
}
Comp.UnitPrice.CurrencyCode = Line.AppliedCurrencyCode
Comp.CurrencyCode = Line.AppliedCurrencyCode

// convert currency if needed
if ( Line.FromCurrencyCode!=null && Line.FromCurrencyCode!=Line.AppliedCurrencyCode ) {
 if (ConvRate?.MessageTypeCode == 'ERROR') {
 finest('creating line message')
 Line.MessageTypeCode = 'ERROR'
 Ch.MessageTypeCode = 'ERROR'
 def messageMap = [ParentEntityCode:'LINE', ParentEntityId:Line.LineId, HeaderId:Header.HeaderId,
 MessageText:ConvRate.PrcErrorMessage]
 msg = Message.locate(messageMap)
 if ( msg==null ) {
 // create new error message for Line
 msg = Message.insert(messageMap)
 msg.PricingMessageId = getNextId()
 msg.MessageName = ConvRate.PrcMessageName
 msg.MessageTypeCode = Line.MessageTypeCode
 }
 }
 else {
 Ch.CurrencyCode = Line.AppliedCurrencyCode
 Comp.UnitPrice.Value *= ConvRate.ConversionRate?:1
 }
}
//end base price override
```

Here's the complete, revised code.

```
Comp = ChargeComponent.insert([ChargeComponentId:++ServiceParam.ChargeComponentIdCntr, ChargeId:
 Ch.ChargeId])

Comp.PriceElementCode = ElementParam
Comp.PriceValidFrom = Line.PricingDate
if ( EndDate!=null ) Comp.PriceValidUntil = EndDate
Comp.SequenceNumber = (Long) Ch.CompSeqCntr++
Comp.SourceId = Candidate.PriceListChargeId
Comp.SourceTypeCode = 'PRICE_LIST_CHARGE'

fine({'Before UnitPrice Calc - CalcMethod: '+Candidate.CalculationMethod})
Comp.createDataObject('UnitPrice')
//Base List Price Override
if(Candidate.CalculationMethod == 'COVERED_ITEM_PRICE_PERCENT') {
 fine({'UnitPrice Calc - Cand.CalculationAmount: '+Candidate.CalculationAmount+' CoverageBasis:
 '+Candidate.CoverageBasisValue})
 Comp.UnitPrice.Value = Candidate.CalculationAmount * Candidate.CoverageBasisValue/ 100
 Ch.HasPercentPriceFlag = true
}
else {
 //Base List Price Override
 if (LinePrcOverrideEff?.SalePrcOverrideVal_Custom != null)
 Comp.UnitPrice.Value = LinePrcOverrideEff.SalePrcOverrideVal_Custom
 else
```

```
 Comp.UnitPrice.Value = Candidate.BasePrice
}
Comp.UnitPrice.CurrencyCode = Line.AppliedCurrencyCode
Comp.CurrencyCode = Line.AppliedCurrencyCode

// convert currency if needed
if ( Line.FromCurrencyCode!=null && Line.FromCurrencyCode!=Line.AppliedCurrencyCode ) {
 if (ConvRate?.MessageTypeCode == 'ERROR') {
 finest('creating line message')
 Line.MessageTypeCode = 'ERROR'
 Ch.MessageTypeCode = 'ERROR'
 def messageMap = [ParentEntityCode:'LINE', ParentEntityId:Line.LineId, HeaderId:Header.HeaderId,
 MessageText:ConvRate.PrcErrorMessage]
 msg = Message.locate(messageMap)
 if ( msg==null ) {
 // create new error message for Line
 msg = Message.insert(messageMap)
 msg.PricingMessageId = getNextId()
 msg.MessageName = ConvRate.PrcMessageName
 msg.MessageTypeCode = Line.MessageTypeCode
 }
 }
 else {
 Ch.CurrencyCode = Line.AppliedCurrencyCode
 Comp.UnitPrice.Value *= ConvRate.ConversionRate?:1
 }
}
//end base price override

Ch.RunningUnitPrice+=Comp.UnitPrice.Value
if ( Ch.PricedQuantity!=null ) {
 Comp.createDataObject('ExtendedAmount')
 Comp.ExtendedAmount.Value = Comp.UnitPrice.Value*Ch.PricedQuantity.Value
 Comp.ExtendedAmount.CurrencyCode = Comp.UnitPrice.CurrencyCode
 def fromUomCode = null
 def fromUomValue = null
 if ( 'COVERED_STANDARD' == Line.ItemType && Line.PhantomFlag ){
 fromUomCode = Ch.CoverageDurationUOMCode
 fromUomValue = Ch.CoverageDuration.Value
 }else if ( Line.ItemType in ['STANDARD', 'COMPONENT', 'ROOT'] && null != Line.ServiceDuration?.Value &&
 null != Line.ServiceDurationPeriodCode ) {
 fromUomCode = Line.ServiceDurationPeriodCode
 fromUomValue = Line.ServiceDuration?.Value
 }
 if (null != fromUomCode && null != fromUomValue) {
 Comp.createDataObject('CoverageExtendedAmount')
 if ('ONE_TIME' == Ch.PriceTypeCode) {
 Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount.Value
 }else if ('RECURRING' == Ch.PriceTypeCode) {
 if (Ch.PricePeriodicityCode != fromUomCode) {
 def fromUom = getOKCUomMappings(fromUomCode)
 def toUom = getOKCUomMappings(Ch.PricePeriodicityCode)

 if(null == fromUom || null == toUom){
 Line.MessageTypeCode = 'ERROR'
 Ch.MessageTypeCode = Line.MessageTypeCode
 def uomMsg = Message.insert([
 MessageName:'QP_PDP_PARTIAL_PERIOD_UOM_NA',
 MessageTypeCode: 'ERROR',
 ErrorType:(Ch.HasPercentPriceFlag?'PERCENTAGE':'AMOUNT'),
 PricingMessageId: getNextId(),
 ParentEntityCode: 'LINE',
 ParentEntityId: Line.LineId,
 HeaderId: Header.HeaderId])
 def chargeName = getChargeDefinitionName(Ch.ChargeDefinitionId)
 if(null == fromUom){
```

```
def fromUomTranslation = getUomTranslation(fromUomCode)
uomMsg.MessageText = getFndMessage('QP', 'QP_PDP_PARTIAL_PERIOD_UOM_NA',
['CHARGE_DEF_NAME':chargeName?.Name , 'UOM_CODE':fromUomTranslation?.UnitOfMeasure])
}

if(null == toUom){
def toUomTranslation = getUomTranslation(Ch.PricePeriodicityCode)
uomMsg.MessageText = getFndMessage('QP', 'QP_PDP_PARTIAL_PERIOD_UOM_NA',
['CHARGE_DEF_NAME':chargeName?.Name , 'UOM_CODE':toUomTranslation?.UnitOfMeasure])
}
}

if(null != fromUom && null != toUom){
durationConvRate = getDurationConversionRate(fromUom, toUom, CoverageAssociation, Line)
if(null == durationConvRate){
def fromUomTranslation = getUomTranslation(fromUomCode)
def toUomTranslation = getUomTranslation(Ch.PricePeriodicityCode)
Line.MessageTypeCode = 'ERROR'
Ch.MessageTypeCode = 'ERROR'
Message.insert([
MessageName:'QP_PDP_PARTIAL_PERIOD_CONVERS',
MessageText: getFndMessage('QP', 'QP_PDP_PARTIAL_PERIOD_CONVERS',
['UOM': toUomTranslation?.UnitOfMeasure, 'COV_UOM': fromUomTranslation?.UnitOfMeasure]),
MessageTypeCode: Line.MessageTypeCode,
ErrorType:(Ch.HasPercentPriceFlag?'PERCENTAGE':'AMOUNT'),
PricingMessageId: getNextId(),
ParentEntityCode: 'LINE',
ParentEntityId: Line.LineId,
HeaderId: Header.HeaderId])
} else {
if( RollupCh != null )
RollupCh.PartialPeriodDurationConversionRate = durationConvRate
Ch.PartialPeriodDurationConversionRate = durationConvRate
Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount?.Value * durationConvRate
}
}
} else
Comp.CoverageExtendedAmount.Value = Comp.ExtendedAmount.Value * fromUomValue
}
Comp.CoverageExtendedAmount.CurrencyCode = Comp.ExtendedAmount.CurrencyCode
}
}

finer(AlgmName+': created '+Comp.PriceElementCode+' charge component '+Comp.ChargeComponentId
+' with unit price '+Comp.UnitPrice.Value+' '+Comp.UnitPrice.CurrencyCode+', extended amount
 '+(Comp.ExtendedAmount!=null ? Comp.ExtendedAmount.Value+' '+Comp.ExtendedAmount.CurrencyCode : null))
```

13. Click **Save and Close**.
14. On the Manage Algorithms page, click **Actions > Publish**.

Test Your Set Up

1. On the Manage Algorithms page, open Price Sales Transaction for editing.
2. On the Edit Algorithm page, click **Test**.
3. Click **Actions > Add Row**, then set the value:

Attribute	Value
Test Case Name	Test Base Price Override

4. In the Test Input area, in the row that contains PriceRequest in column Variable Name, click the **pencil** in the Variable Value column.
5. In the Edit Variable dialog, delete all the code, then add this code:

```
<?xml version="1.0" encoding="UTF-8"?>
<PriceRequestInternal:PriceRequestInternalType xmlns:ns0="http://xmlns.oracle.com/adf/svc/
types/"xmlns:PriceRequestInternal="http://xmlns.oracle.com/apps/scm/pricing/priceExecution/
pricingProcesses/pricingInternal/PricingInternal"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="PriceRequestInternal:PriceRequestInternalType">
<PriceRequestInternal:Header>
 <PriceRequestInternal:CustomerId>1028144</PriceRequestInternal:CustomerId>
 <PriceRequestInternal:HeaderId>101</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:CalculatePricingChargesFlag>true</
PriceRequestInternal:CalculatePricingChargesFlag>
 <PriceRequestInternal:CalculateShippingChargesFlag>false</
PriceRequestInternal:CalculateShippingChargesFlag>
<PriceRequestInternal:CalculateTaxFlag>false</PriceRequestInternal:CalculateTaxFlag>
 <PriceRequestInternal:SellingBusinessUnitId>103</PriceRequestInternal:SellingBusinessUnitId>
 <PriceRequestInternal:SellingLegalEntityId>204</PriceRequestInternal:SellingLegalEntityId>
 <PriceRequestInternal:TransactionTypeCode>ORDER</PriceRequestInternal:TransactionTypeCode>
</PriceRequestInternal:Header>
<PriceRequestInternal:PricingServiceParameter>
 <PriceRequestInternal:PricingContext>SALES</PriceRequestInternal:PricingContext>
</PriceRequestInternal:PricingServiceParameter>
<PriceRequestInternal:Line>
 <PriceRequestInternal:HeaderId>101</PriceRequestInternal:HeaderId>
 <PriceRequestInternal:InventoryItemId>100000017351102</PriceRequestInternal:InventoryItemId>
 <PriceRequestInternal:InventoryOrganizationId>204</PriceRequestInternal:InventoryOrganizationId>
 <PriceRequestInternal:LineId>1001</PriceRequestInternal:LineId>
 <PriceRequestInternal:LineCategoryCode>ORDER</PriceRequestInternal:LineCategoryCode>
 <PriceRequestInternal:LineQuantity unitCode="Ea" xmlns:tns="http://xmlns.oracle.com/adf/svc/
errors/">2</PriceRequestInternal:LineQuantity>
 <PriceRequestInternal:LineQuantityUOMCode>Ea</PriceRequestInternal:LineQuantityUOMCode>
 <PriceRequestInternal:LineTypeCode>ORA_BUY</PriceRequestInternal:LineTypeCode>
</PriceRequestInternal:Line>
<PriceRequestInternal:LinePrcOverrideEff_Custom>
 <PriceRequestInternal:SalePrcOverrideVal_Custom>1000</PriceRequestInternal:SalePrcOverrideVal_Custom>
 <PriceRequestInternal:FulfillLineId_Custom>1001</PriceRequestInternal:FulfillLineId_Custom>
 <PriceRequestInternal:EffLineId_Custom>123</PriceRequestInternal:EffLineId_Custom>
</PriceRequestInternal:LinePrcOverrideEff_Custom>

<PriceRequestInternal:ChangeSummary logging="false" xmlns:sdo="commonj.sdo"/>
</PriceRequestInternal:PriceRequestInternalType>
```

where

Value	Is An Example For
1028144	CustomerId
103	SellingBusinessUnitId
204	SellingLegalEntityId
100000017351102	InventoryItemId

Value	Is An Example For
204	InventoryOrganizationId
Ea	unitCode
Ea	LineQuantityUOMCode
`<PriceRequestInternal:SalePrcO Custom>1000</ PriceRequestInternal:SalePrcOv Custom>`	Specifies to override the net price to $1000. Modify this value for your test, as necessary.

To get the values that you must use in your environment, see section Getting Identifiers for Test Payloads, below in this topic.

6. Click **Run Test > Test Output**, then make sure the output payload includes the details you expect.
7. Verify that Last Execution Status displays Succeeded, or a similar success indication.

   If it displays Failure, then click **Show Exception Details** to examine and troubleshoot.

   If you receive this error, then use an XML editor to validate that your code uses the correct XML format.

   ```
   Error: Unable to parse the variable[PriceRequest] using the service
   definition [Sales.PriceRequestInternal]. Please check the variable value or service schema
   ```

8. Click **Save and Close**.
9. On the Manage Algorithms page, click **Actions > Publish**.

Create a Test Sales Order

1. Sign out, then sign in with the privileges that you need to manage sales orders.
2. Click **Create Order**, enter values for the order header, add an order line, then add a value in your extensible flexfield, such as 199.
3. Save the sales order.
4. Verify that the price of the item includes the price you entered in the extensible flexfield and SalePrcOverrideVal.

Get Identifiers for Test Payloads

1. Make sure you have the privileges that you need to manage sales orders.

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

   - Set attributes to values you must test:

Attribute	Example Value
Customer	Fantastic Laptops

Attribute	Example Value
Business Unit	Vision Operations

- Add an order line that includes the item you're testing. For this example, add AS54888.
- Submit, then notice the sales order number that Order Management creates, such as 12345.

2. Get values for the order header. Run a SQL query on the database you use to store the sales order that you created in step 1.

```
select header_id, sold_to_party_id as CustomerId, org_id as SellingBusinessUnitId, legal_entity_id as
 SellingLegalEntityId
from doo_headers_all where Order_Number = $12345;
```

where

- 12345 is the order number you noted in step 1.
- The query returns values that you can use in your payload for these header attributes:

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
sold_to_party_id	Header.CustomerId	Header_id equals 100413 and CustomerId equals 1006.
org_id	Header.SellingBusinessUnitId	SellingBusinessUnitId equals 204.
legal_entity_id	Header.SellingLegalEntityId	SellinglegalEntityId equals 204.

3. Get values for the order line. Run this SQL query.

```
select inventory_item_id as InventoryItemId, inventory_organization_id as InventoryOrganizationId,
 ordered_uom as LineQuantityUOMCode from doo_fulfill_lines_all where header_id = $56789;
```

where

- 56789 is the header Id in the query result from step 2.
- The query returns values that you can use in your payload for these order line attributes:

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
inventory_item_id	Line.InventoryItemId	Header_id equals 100413 and InventoryItemId equals 149.

Value in SQL	Provides Value for Attribute in Test Payload	Example Values
inventory_organization_id	Line.InventoryOrganizationId	InventoryOrganizationId equals 204.
ordered_uom	Line.LineQuantity.UOMCode Note that you must replace UOMCode in two different locations in the payload. In this example, the value is Ea.	LineQuantityUOMCode equals USD.

Related Topics

- How Profiles, Segments, and Strategies Work Together
- Manage Pricing Segments
- Assign Pricing Strategy
- Manage Price Lists
- Use SQL to Query Order Management Data

Adjust Price According to Ship to Location

Adjust the price that Pricing calculates on the order line according to the ship to location you set on the order header.

For this example, assume:

- You must adjust the price for the AS54888 item according to the ship-to site.
- Add a 10% mark up when shipping to 1100 Stout Street, Denver.
- Add a 15% mark up when shipping to 5748 Shoreline Drive, Vancouver.
- ShipToPartySiteId for Denver is 869784657.
- ShipToPartySiteId for Vancouver is 153429673.
- You already added the AS54888 to Price List for Computer Service and Rentals. You set Base Price to 2500. For details about setting up this price list, see *Manage Price Lists*.

Here's your set up:

1. In the Pricing Administration work area, click **Tasks > Manage Service Mappings**.
2. On the Manage Service Mappings page, open the Sales service mapping.

3. Map the order header:
    - On the Edit Service Mapping page, click Sources, then click the **row** that contains OrderHeader in the Source column.
    - In the Details area, on the Entity Mappings tab, click the **row** that contains Line in the Entity column.
    - On the Attribute Mappings tab, click **Actions > Add Row**, then set the values:

Attribute	Value
Attribute	ShipToPartySiteId
View Object Attribute	ShipToPartySiteId

4. Map the order line:
    - In the Details area, on the Entity Mappings tab, click the **row** that contains Line in the Entity column.
    - On the Attribute Mappings tab, click **Actions > Add Row**, set values, then click **Save and Close**:

Attribute	Value
Attribute	ShipToPartySiteId
View Object Attribute	ShipToPartySiteId

5. Modify the matrix class:
    - Click **Tasks > Manage Matrix Classes**.
    - On the Manage Matrix Classes page, open the Price List Charge Adjustment matrix class.
    - Add a new condition column:

Attribute	Value
Name	Ship To
Source Code Name	ShipTo
Comparison	=
Compare to Attribute	Line.ShipToPartySiteId
Allow Null	Contains a check mark

Attribute	Value
Null Is Wildcard	Contains a check mark
Domain	Domain Type is None Data Type is Number

- Click **Save and Close**.

6. Modify the price list:
   - Click **Price Lists**.
   - Search for, then open Price List for Computer Service and Rentals.
   - On the Price List page, search for AS54888.
   - Select the row that has the item, then click **More Actions > Add Matrix Pricing**.
   - Add the Ship To column.
   - Click **Add Row**, then add them:

Ship To	Adjustment Type	Adjustment Amount	Adjustment Basis
869784657	Markup Percent	10	List Price
153429673	Markup Percent	15	List Price

   To get the ship to Id for your environment, use REST API, a web service, or run an SQL query on the Oracle database.
   - Click **Save > Activate**.
7. Test your set up:
   - Sign into Order Management with the privileges that you need to manage sales orders.
   - In the Order Management work area, create a new sales order:

Attribute	Value
Customer	Computer Service and Rentals
Ship-to Address	1100 Stout Street, Denver

Attribute	Value

- Add the AS54888 to an order line, then notice the Amount attribute displays 2500.
- Click **Save**, then verify that Amount displays 2750, which is the 2500 you set up for the base price plus the 10% mark up.

  Pricing applies the matrix adjustment when you save or submit a sales order.
- Change the Ship-to Address to 5748 Shoreline Drive, Vancouver, then click **Save**.
- Verify that Amount displays 2875, which is 2500 base price plus 15% mark up.

Related Topics

- Manage Price Lists
- Use SQL to Query Order Management Data