Oracle® Advanced Pricing

Implementation Manual

Release 11*i*

Part No. A95419-03

August 2003

ORACLE®

Oracle Advanced Pricing Implementation Manual, Release 11*i*

Part No. A95419-03

Primary Authors: Reb Bowman, Nitin Hase, Vivian Lee, Maria Viglionese Matheny, John Salvini

Contributing Authors: Manoj Arya, Rajendra Badadare, Vishwajit Bhave, Rajeshwari Chellam, Amy Cui, T. Geresh, Sripriya Gopal, Vivek Gulati, Dharmender Gupta, Leslie Hershey, Jayarama Holla, Ashutosh Itkelwar, Jeff Lee, Yang Li, Shu-Hui Lin, Steven Mosura, Sameer Phatarpekar, Abhijit Prasad, Arun Raghavan, Navneet Ritolia, Alison Schofield, Renganathan Sreenivasan, Kannan Tarakad, Ravi Tata, Boon Tea, Giridhar Tippireddy, Hock-Shan Wong, Linda Xu, Li Yang

Contributors: Elizabeth Looney, Jennifer Covert Mosinski, David Reitan, Rebecca Zarchikoff

Contents

2 Implementation Overview

3 Implementation Methodology

4 Pricing Security

5 Modifiers

6 Multi-Currency Price Lists and Agreements

7 Unit of Measure

8 Multiple Organizations

9 Precedence and Best Price

10 Attribute Management

Send Us Your Comments

Oracle Advanced Pricing Implementation Manual, Release 11*i*

Part No. A95419-03

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us by sending an e-mail to the following address:

- Electronic mail: mfgdoccomments@oracle.com

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Audience for This Guide

Welcome to Release 11*i* of the *Oracle Advanced Pricing Implementation Manual*.

This manual assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle Advanced Pricing

  If you have never used Oracle Advanced Pricing, Oracle suggests you attend one or more of the Oracle Advanced Pricing training classes available through Oracle University.
- The Oracle Applications graphical user interface.

  To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

How To Use This Manual

This manual contains the information you need to implement Oracle Advanced Pricing.

- Chapter 1 provides an introduction to Oracle Advanced Pricing implementation.
- Chapter 2 provides an overview of Oracle Advanced Pricing features and implementation.
- Chapter 3 discusses implementation methodology for Oracle Advanced Pricing.
- Chapter 4 discusses the implementation of pricing security.

- Chapter 5 discusses the implementation of modifiers
- Chapter 6 discuss the implementation of Multi-Currency price lists and Agreements.
- Chapter 7 discusses the implementation of units of measure.
- Chapter 8 discusses Oracle Advanced Pricing for multiple organizations.
- Chapter 9 discusses implementation considerations for precedence and best price.
- Chapter 10 discusses attributes management and mapping.
- Chapter 11 discusses get custom price functionality.
- Chapter 12 discusses implementation considerations for events and phases.
- Chapter 13 discusses the Pricing Engine Request Viewer window.
- Chapter 14 discusses setting profile options.
- Chapter 15 provides information about integrating with Oracle Advanced Pricing.
- Chapter 16 discusses technical considerations for implementing Oracle Advanced Pricing.
- Appendix A provides a list containing the window and navigation paths used for Oracle Advanced Pricing.
- Appendix B provides an essay on optimizing performance for Oracle Advanced Pricing.
- Appendix C is a case study that examines Oracle Advanced Pricing implementation in a fictional high-tech company.
- Appendix D is a case study that examines Oracle Advanced Pricing implementation in a fictional food service company.
- Appendix E provides a listing of Lookups used in Oracle Advanced Pricing.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to

evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Other Information Sources

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of Oracle Advanced Pricing.

If this manual refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those guides.

Online Documentation

All Oracle Applications documentation is available online (HTML or PDF).

- **Online Help** - The new features section in the HTML help describes new features in 11*i*. This information is updated for each new release of Oracle Advanced Pricing. The new features section also includes information about any features that were not yet available when this manual was printed. For example, if your administrator has installed software from a mini-packs an upgrade, this document describes the new features. Online help patches are available on Oracle*MetaLink*.
- **About Documentation** - The Oracle Order Management About Documentation contains new and changed features, software updates, upgrade considerations, new and changed setup steps, new and changed windows, and new and changed public APIs for the latest release of Oracle Order Management. The About Document is available on Oracle*MetaLink*.
- **Readme File** - Refer to the readme file for patches that you have installed to learn about new documentation or documentation patches that you can download.

Related User's Guides

Oracle Advanced Pricing shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user's guides when you set up and use Oracle Advanced Pricing.

You can read the guides online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle Store at http://oraclestore.oracle.com.

Guides Related to All Products

Oracle Applications User's Guide

This guide explains how to enter data, query, run reports, and navigate using the graphical user interface (GUI) available with this release of Oracle Advanced Pricing (and any other Oracle Applications products). This manual also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

You can access this user's guide online by selecting Getting Started with Oracle Applications from any Oracle Applications help file.

User Guides Related to This Product

Oracle Advanced Pricing User's Guide

This guide describes how to set up modifiers, price lists, formulas, pricing agreements, pricing rules, and how to price special orders in Oracle Advanced Pricing.

Oracle Applications Demonstration User's Guide

This guide documents the functional story line and product flows for Vision Enterprises, a fictional manufacturer of personal computers products and services. As well as including product overviews, the book contains detailed discussions and examples across each of the major product flows. Tables, illustrations, and charts summarize key flows and data elements.

Oracle Inventory User's Guide

This guide describes how to define items and item information, perform receiving and inventory transactions, maintain cost control, plan items, perform cycle counting and physical inventories, and set up Oracle Inventory.

Oracle Order Management User's Guide

This guide describes how to enter sales orders and returns, copy existing sales orders, schedule orders, release orders, create price lists, and discounts for orders, run processes, and create reports.

Oracle Self Service Web Applications User's Guide

This guide describes how Oracle Self Service Web Applications enable companies to provide a self-service and secure web interface for its employees, customers and suppliers. Employees can change their personal status, submit expense reports or request supplies; customers can check on their orders; and suppliers can share production schedules with their trading partners. This guide is available in HTML only.

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications, the Oracle8 technology stack, and the Oracle8*i* Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user's guides and implementation guides.

Upgrading Oracle Applications

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be either at Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0, to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

Maintaining Oracle Applications

Use this guide to help you run the various AD utilities, such as AutoUpgrade, AutoPatch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities. This guide also provides information on maintaining the Oracle applications file system and database.

Oracle Applications System Administrator's Guide

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information on how to define security, customize menus and online help, and manage concurrent processing.

Oracle Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

Oracle Applications Developer's Guide

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle Forms Developer 6*i* forms so that they integrate with Oracle Applications.

Oracle Applications User Interface Standards for Forms-Based Products

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for the Oracle Applications products and how to apply this UI to the design of an application built by using Oracle Forms.

Other Implementation Documentation

Oracle Applications Product Update Notes

Use this guide as a reference for upgrading an installation of Oracle Applications. It provides a history of the changes to individual Oracle Applications products between Release 11.0 and Release 11*i*. It includes new features, enhancements, and changes made to database objects, profile options, and seed data for this interval.

Multiple Reporting Currencies in Oracle Applications

If you use the Multiple Reporting Currencies feature to record transactions in more than one currency, use this manual before implementing Oracle Advanced Pricing. This manual details additional steps and setup considerations for implementing Oracle Advanced Pricing with this feature.

Multiple Organizations in Oracle Applications

This guide describes how to set up and use Oracle Advanced Pricing with Oracle Applications' Multiple Organization support feature, so you can define and support different organization structures when running a single installation of Oracle Advanced Pricing.

Oracle Applications Flexfields Guide

This guide provides flexfields planning, setup and reference information for the Oracle Advanced Pricing implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This manual also provides information on creating custom reports on flexfields data.

Oracle eTechnical Reference Manuals

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on Oracle*MetaLink*.

Oracle Manufacturing APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Manufacturing.

Oracle Order Management Suite APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Order Management Suite.

Oracle Applications Message Reference Manual

This manual describes all Oracle Applications messages. This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Training and Support

Training

Oracle offers a complete set of training courses to help you and your staff master Oracle Advanced Pricing and reach full productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle University at any one of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization structure, terminology, and data as examples in a customized training session delivered at your own facility.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Advanced Pricing working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8*i* server, and your hardware and software environment.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using Oracle Applications can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources and customer relationship management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 145 countries around the world.

Your Feedback

Thank you for using Oracle Advanced Pricing and this implementation manual.

Oracle values your comments and feedback. At the beginning of this manual is a Reader's Comment Form you can use to explain what you like or dislike about Oracle Advanced Pricing or this implementation manual. You can send your comments to the following e-mail address: mfgdoccomments@oracle.com.

1

Introduction

This manual addresses issues for the implementation of Oracle Advanced Pricing. This introduction includes the following topics:

Overview of Oracle Advanced Pricing

Oracle Advanced Pricing is a rules-based application that uses an engine component to service the pricing requirements of Oracle applications seeking to price customer-facing transactions. Oracle Advanced Pricing provides you with pricing actions that you can set up for application to customer revenue transactions. Oracle Advanced Pricing also enables you to define a set of sophisticated pricing rules, and to set up pricing controls that can be used in conjunction with the rules to precisely govern how and when pricing actions are applied to transactions.

The pricing engine is the mechanism that applies pricing actions to transactions. The pricing engine is a software component of Oracle Advanced Pricing that is called by an application such as Oracle Order Management or iStore. Applications make calls to the pricing engine when transactions need pricing services or need to be priced.

A calling application must provide the pricing engine with information about the product to be priced and the customer - this information is contained in an internal format called a pricing request structure. The pricing engine extracts from its tables the applicable pricing rules, pricing actions, and control information that have been set up. The engine then selects the proper actions and computes their effect. This information is returned by the engine to the calling application.

Oracle Advanced Pricing provides these capabilities to the pricing engine using open API calls. These APIs, along with others, are documented in the *Oracle Order Management Suite APIs and Open Interfaces Manual.*

Oracle Advanced Pricing Versus Basic Pricing

The term basic pricing refers to a component of Oracle Order Management that provides pricing functionality when Oracle Advanced Pricing is not installed. Oracle Advanced Pricing extends and expands the capabilities of basic pricing. Oracle Advanced Pricing and basic pricing have common software components. To determine the appropriate mode in which to run, the pricing system software components examine the installation type (either full or shared).

Users of basic pricing are installed shared and are not licensed to use Oracle Advanced Pricing capabilities. When in basic mode, the pricing system software components restrict exposure of Advanced features in the setup windows. Because the information necessary to drive Oracle Advanced Pricing functionality cannot be set up in a pricing implementation running in basic mode, use of Oracle Advanced Pricing features is also inhibited.

Users who have licensed Oracle Advanced Pricing are installed Full. The user interface setup screens enable setup for all information needed to drive features provided by Oracle Advanced Pricing. Because all information can be set up, Oracle Advanced Pricing features are fully enabled.

The following table depicts the primary differences between Oracle Advanced Pricing and basic pricing capability included in Oracle Order Management:

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Price list	Enables you to set up: ■ Price List Header ■ Price List Lines ■ One currency per list ■ Effective date at list ■ Rounding factors ■ Attach payment terms ■ Attach freight terms· ■ Attach freight carriers ■ Percentage price for service items ■ Service price ■ Active Flag ■ Mobile Download ■ Global Flag to support Security feature ■ Security enabled rules checking	Advanced Pricing includes all Basic price list features and adds the capability to define point and range price breaks on a price list.
Copy price list	Basic Pricing provides this capability. Security-enabled rules checking.	Same as Basic
Adjust price list	Basic Pricing provides this capability. Security enabled rules.	Same as Basic
Add items to price list	Supported. (User must have Maintain access privilege to Add Items.)	Same as Basic. (User must have Maintain access privilege to Add Items.)

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
GSA pricing	Supported	Same as Basic
Secondary price lists	One secondary price list is supported.	Advanced Pricing adds capability to define and use unlimited secondary lists. The chaining of secondary lists is NOT supported. A profile option enables a qualifier check for secondary price list.
Price List Maintenance	Not available in Basic Pricing.	Advanced Pricing provides search and maintenance capability for a single price list or across multiple price lists. Use the Bulk Change feature for mass updates or update individual price list lines. Updates can be made to price list lines for: ■ Value ■ Static and Dynamic Formula ■ Price List Line effective dates Price List Maintenance is available through the HTML user interface. See the *Oracle Advanced Pricing User's Guide*, Using the Price List Maintenance feature for more information.
Pricing attributes on Price List	In Basic Pricing, you can use one data source, called a context, per order line. Each context can have 100 attributes.	Advanced Pricing adds capability to support multiple contexts. You can use seeded values or you may define your own contexts. You are limited to 100 attributes per context.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Multi-Currency Conversion Lists	Basic Pricing does not include multi-currency conversion lists.	Advanced Pricing allows users to maintain prices in a single list with one base currency and define multiple currency conversion rates and currency specific markup/markdown equations. A Multi-Currency Conversion window enables users to define Currency To conversion criteria. Seeded conversion types include: Fixed, Formula, User-defined, Spot, EMU fixed, transaction and corporate. Note: Some of these seeded conversion types require Oracle General Ledger to be installed. ■ Users can define markup criteria per currency definition. ■ A concurrent program Update Price Lists with Multi-Currency Conversion Criteria is provided.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Pricing Security	The Pricing Security Administrator responsibility can grant access privileges across Operating Units to the following pricing entity types: Standard Price List Modifier Agreement Price List The following access levels are supported by site level profile settings: ■ View Only ■ Maintain Privilege You can grant access privileges to the following Grantee Types: ■ Global ■ Operating Unit ■ Responsibility ■ User Pricing Security supports authorization roles for creating maintaining, and viewing pricing data. Pricing Security supports dividing the data used by the pricing engine. Security Privileges are set up and managed in the HTML user interface.	Advanced Pricing includes the Basic Pricing Security features plus the following: The Pricing Administrator can define "sets" of pricing entities to which access roles can be granted to any grantee type and grantee.
Qualifiers and groups	Not available in basic pricing. Defaulting is available for price lists and modifiers.	Included in addition to defaulting for basic pricing.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Qualifiers: Price Lists	Qualifiers are supported via defaulting in Basic Pricing. You cannot define qualifiers other than the default values. Seeded defaults include customer, order type and others from Order Management: ■ Uses current method of customer assigned in RA_ Customers where price list is assigned. ■ Precedence is populated in system and can be updated. ■ Includes defaulting of TCA (Trading Community Architecture) attributes.	Includes Basic features and adds the following features for the full qualifier feature: ■ Ability to attach qualifier groups or enter qualifiers for the price list. ■ All seeded context values for Advanced Pricing as well as user-defined qualifiers (requires Attribute Mapping) are supported. ■ Precedence is derived from Precedence Number field on the Contexts Setup window and can be configured by the user in Advanced Pricing. ■ Multiple with and/or relations between qualifiers is possible. ■ Qualifiers enable you to define unlimited price lists for an item. ■ Note: You cannot define Order Amount as a qualifier
Product hierarchy price list notes	Basic Pricing supports Product Context for Item number, Item Category, and All_Items.	Advanced Pricing includes the product contexts in Basic Pricing, and adds capability to define additional contexts or attributes using Attribute Mapping feature. Note: You can define addtional attributes for the product context, but you cannot create additional product contexts.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Price breaks on price lists	Not supported in basic pricing.	Advanced Pricing enables price breaks on price lists. The following price break quantity 'drivers' are supported: ■ Quantity ■ Amount (excluding Item Amount) ■ Weight ■ Other user defined attributes The following break types are supported for Application Method of Unit Price: ■ Point break ■ Range break Block Pricing (Application Method of Block Price): ■ Define a price for the entire set of a block. Set up 'lump sum' or flat rate pricing for minimum quantities. ■ Point break ■ Range Breaks: Recurring, Value ■ Set up various price break combinations using Point and Range break types with multiple application methods for the set of the Block.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Formulas	Basic Pricing enables you to define Static Formulas. Static Formulas require a concurrent manager program to be run that populates the list price column in the price list window, prior to that price being available for engine calls. Basic Pricing gives you the capability to attach formulas to price lists. Formulas in Basic Pricing can include mathematical operators, numeric operands, and PL/SQL Functions such as min/max may be imbedded in a formula. Sixteen seeded formulas are included: ■ Eight Cost-to-Charge formulas ■ Eight Cost-to-Charge with Markup formula. These formulas can be updated.	Advanced Pricing adds the following features: ■ You can attach formulas to modifiers as well as price lists. However, static formula generation is available only for price lists in Advanced Pricing. ■ Dynamic Formula Expansion: Dynamic expansion enables the pricing engine to dynamically calculate the formula price at based on variable values available at run time. Dynamically expanded formulas can be attached to either price lists or modifiers. Note: Price lists that have a dynamic formula with a Modifier Value as a component cannot be attached to a price list line.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Formula components	Component types include: numeric values, one product context (item), factor list - one pricing context throughout formula with up to 100 attributes. Context must match context on price list line that formula is being attached to. Adjustment factors (factor list): basic pricing supports multiple factors. However, there is limitation in basic pricing of one pricing attribute context throughout formula with up to 100 attributes. Seeded context must be used. Context must match context on price list line to which the formula is attached.	Advanced Pricing includes all formula component types available in Basic Pricing, and adds the following: ■ List price of an item in a specific price list (product and service) pricing attribute. ■ Modifier Value is entered in the Modifier Value field on the modifier setup window for a modifier line as a component to a formula. ■ A FUNCTION type which calls Advanced Pricing API Get_Custom_Price. ■ Adjustment Factors: Advanced Pricing supports multiple adjustment factors. Both context - seeded and user-defined contexts and attributes can be used as adjustment factors. ■ Multiple pricing contexts for pricing attributes.
Modifier list types	Basic Pricing supports the following Modifier List Types: ■ Discount ■ Surcharge ■ Freight and Special Charges Security-enabled rules checking.	Advanced Pricing provides all Modifier List Types provided in Basic Pricing, and adds the following: ■ Promotion ■ Deal
Modifier application methods	Basic Pricing supports the following modifier application methods: ■ Manual ■ Automatic ■ Overrideable	All Oracle Order Management Basic methods and provides the Ask For method.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Active box on Modifier header	Not enabled in Basic Pricing.	Enabled in Advanced Pricing.
Effectivity date controls	Effectivity Date: order date only	Order date, plus Advanced Pricing adds: ■ Ship Date ■ Order and Ship Date Advanced Pricing also adds fields for Promotion Version, parent promotion, and parent version.
Modifier levels and level code	Order, order line level.	Basic level codes and adds Line Group.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Modifier header qualifiers and attributes	Basic Pricing provides seeded qualifier contexts including: ■ Customer ■ Price List ■ Unlimited price lists (new OM feature) ■ Seeded attributes within contexts. Customer context includes these attributes: Customer class (defined in RA customers); Site; and Customer Name. Note: Modifier List Type of Freight and Special Charges supports additional modifier header qualifiers and attributes. Security-enabled rules checking.	Includes Basic Pricing features, plus Advanced Pricing adds these capabilities: ■ User assignment and override of precedence is possible. ■ Qualifiers can be used with both seeded and user defined contexts possible. ■ Users can define attributes with user-defined contexts. Up to 100 attributes can be defined for each context, and there is no limit to the number of contexts that users can define. ■ Seeded qualifier attributes can be redirected to other data sources using attribute mapping feature (see attribute mapping topic). ■ Users can control qualifier precedence. ■ Entered Context: Seeded and user-defined is possible. ■ User control of attribute precedence possible. ■ Attaching qualifier groups to modifier headers possible.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Modifiers: line types	Types available include: ■ Discount: Fixed amount, Percent, New Price ■ Surcharge: Fixed amount, Percent, New Price ■ Freight Charges: Fixed amount, Percent, Lumpsum ■ Price Break	Advanced Pricing includes the Basic Pricing types and adds:· ■ Coupon Issue ■ Item Upgrade ■ Other Item Discount ■ Terms Substitution ■ Promotional Goods
Modifiers: line qualifiers and attributes	Basic Pricing provides the following fixed qualifiers including: ■ Agreement Name ■ Agreement Type ■ Order type ■ Customer PO Usable Product Attributes include: ■ Item ■ Item categories ■ All items Usable pricing attributes:· ■ Limited to one context when product attribute is ALL Items Note: Modifier Line Type of Freight and Special Charges supports additional modifier line qualifiers and attributes.	Advanced Pricing includes the line level qualifiers provided in Basic Pricing and adds: ■ Define unlimited number of qualifiers. ■ Can attach qualifier groups as qualifiers. ■ Can use seeded contexts and define additional contexts. ■ User-defined contexts active. ■ Multiple qualifiers with AND/OR conditions can be created. Product attributes in Advanced Pricing include those defined in Basic Pricing and adds user-defined “alternative” product hierarchy contexts from outside Oracle inventory structure. An unlimited number of pricing attributes can be created in Advanced Pricing. The user can change the precedence.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Modifiers: price breaks	Line level price breaks in basic include: ■ Percent ■ Amount ■ Fixed Price ■ Equal operator ■ Break Type Code: Limited to point type only. ■ Volume Type: Item Amount, Item Quantity	Line level price breaks in Advanced Pricing includes all price break features in basic Pricing and adds the following: ■ Equal, between arithmetic operator ■ Point and Range ■ Context: Seeded and user-defined ■ Recurring Define automatic price breaks based on Net Amount.
Modifiers other differences	Defaulted to single available bucket (bucket 1). User control of incompatibility feature inactive in basic pricing. User control of phase/event mapping inactive in Basic.	Oracle Advanced Pricing adds: multiple buckets, seeded or user defined buckets, user control of phase event mapping, user control of incompatibility/exclusivity modifier control feature, user control of incompatibility resolve method by setting choice of best price or precedence, accrual features, formula in a modifier feature, and active exclude item (product attribute).

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Agreements	Basic Pricing agreement features enable you to: ■ Set payment terms: Invoicing rule, Accounting rule ■ Set freight terms: Freight carrier ■ Create Standard and Agreement Price List ■ Define using customer part numbers. ■ Make revisions to original terms. ■ Enter Revision numbers, date, reasons - only at line level. ■ Set Effective dates. ■ Create Volume breaks. Security enabled rules checking.	Same as Basic Pricing.
Attribute mapping	Attribute mapping feature is not available.	Provides attribute mapping that enables pricing to use data from a wide variety of non-standard sources to drive your pricing. These data sources can be within Oracle applications or from outside Oracle applications.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Oracle Order Management integration with Pricing	Order Management integration with Basic Pricing supports the View Adjustments feature and the following features: ■ Manual price override ■ Manual discount override ■ Reason Code ■ Modifier Dates ■ Display qualifier attributes: button on UI ■ Display pricing attributes: button on UI ■ One pricing attribute context ■ Up to 100 attributes can be used in the single context ■ Action >Price Order is available. ■ Action >Price Line is available. ■ Calculate Price Freeze Flag can be set. ■ Blanket Order Support: Create a simple price list from the blanket window. Security enabled rules checking. Global Flag and security rules enforced. If Global box selected on price list, modifier, or agreement price list, the pricing window can be used regardless of which operating unit created it.	Order Management, when integrated with Advanced Pricing, provides all features supported with Basic Pricing and adds: ■ View Adjustment: Relationship button, Item Upgrade, Term Substitution ■ Coupon entry ■ Ask for promotions ■ User entered attributes: Multiple attribute contexts can be used ■ Up to 100 attributes per context ■ Promotional Limits Hold: P lace holds where violated; No holds are placed; Place order on hold when any violation occurs.
Pricing Engine	In Basic pricing, the pricing engine does not return Oracle Advanced Pricing modifiers or features to the calling application. Security rules checking is enabled. Global box and security rules are used.	In Oracle Advanced Pricing, the pricing engine is enabled to return all advanced features.

Table 1–1 Comparison between Basic and Advanced Pricing

Pricing Features	Basic Pricing in Oracle Order Management	Oracle Advanced Pricing
Reports	Reports with Basic Pricing include: ■ Order Discount Detail Report ■ Order Discount Summary Report Security-enabled rules checking across all reports.	Advanced Pricing adds the following reports: ■ Accruals Details Report ■ Attribute Mapping Rules Error Report ■ Cross Order Volume Report ■ Modifier Detail Report ■ Price List Detail Report (including Multi-currency fields when Multi-currency is installed) ■ Pricing Formulas Report ■ Qualifier Grouping Report
Concurrent Programs	Not available in Basic Pricing.	Advanced Pricing adds the following concurrent programs: ■ Build Attribute Mapping Rules ■ Build Formula Package ■ Cross Order Volume Load ■ Purge Pricing Engine Requests ■ QP: Maintains the denormalized data in QP Qualifiers ■ Update Price Lists with Multi-Currency Conversion Criteria ■ Update Promotional Limit Balances

A table of differences between Oracle Order Management basic pricing functionality and the pricing capabilities of Release 10.7 and Release 11.0 of Oracle Applications is documented in the *Oracle Order Management Suite Implementation Manual.*

Terminology

Calling Application

See invoking application.

Customer Hierarchy

When installed with Oracle Order Management, the customer hierarchy in Oracle Advanced Pricing is seeded to enable roll up of individual customers according to the following structure, which is based on RA_Customer:

- Customer name
- Customer class
- Site
- Ship to
- Bill to
- Agreement name
- Agreement type
- GSA
- Sales channel
- Account type

You can use elements of the customer hierarchy shown by referencing them as qualifiers for either modifier or price list objects.

Certain attributes from the Trading Community Architecture (TCA) customer tables are also seeded for use as qualifier attributes. These are:

- Party ID
- Customer account ID (sold_to_org_id)
- Ship to party site
- Invoice to party site

Additional levels of product hierarchy capabilities can be defined in Oracle Advanced Pricing. For example, you may want to define flexfields on customer tables to house additional customer groupings. To define such levels, see Chapter 10, "Attribute Management".

The previously mentioned hierarchy is built using the Oracle Customer Master and Trading Community Architecture. When Oracle Advanced Pricing is installed without Oracle Order Management or without the standard customer tables, the seeded customer qualifiers listed are not available. In that case, you must define an alternative table structure location where the customer hierarchy exists and list the attributes it contains. It is not necessary for the table structure supporting your alternative structure to exist within Oracle Applications.

Mapping an alternative customer hierarchy can be accomplished using attribute mapping. For more information on attribute mapping, see Chapter 10, "Attribute Management".

Invoking Application

An invoking application is an application that calls the Oracle Advanced Pricing engine to obtain pricing calls. For example, both Oracle Order Management and iStore invoke the Oracle Advanced Pricing engine to apply pricing to customer transactions.

Pricing Engine

The pricing engine is a program module of Oracle Advanced Pricing called by an invoking application as transactions are processed that must be priced.

Pricing Request

A pricing request is the specific information provided to the Pricing engine when the engine is called by the invoking application. In general, this includes who the customer is, what the product is, what attributes may be associated with the customer or product that may be used by the pricing engine, the pricing date, and other pricing data attributes that may be required by the pricing engine.

Product Hierarchy

The Oracle Advanced Pricing product hierarchy is pre-seeded with Item context (based on the Oracle Applications Item Master, MTL_SYSTEM_ITEMS table). This context delivers a two level product hierarchy consisting of:

- Item number
- Item category

Oracle Advanced Pricing provides two additional capabilities that extend this hierarchy:

- You can define a product hierarchy level more specific than Item by using pricing attributes on a price list. This provides a product hierarchy level below item.
- Oracle Advanced Pricing also recognizes a super category of items called item ALL. Item ALL consists of all the items in a price list.

You can use item, item and pricing attribute, or item categories as defaults to control the operation of price lists and modifiers.

Additional levels of product hierarchy capabilities can be defined in Oracle Advanced Pricing. For example, you may want to define flexfields on the customer tables to house additional customer groupings. For more information about defining such levels, see Chapter 10, "Attribute Management".

The above hierarchy is built using the Oracle Inventory Item Master. When Oracle Advanced Pricing is installed without Oracle Order Management or the standard item master tables being present, the seeded item attributes listed above will not be available. In that case, you must define an alternative table structure location where the product hierarchy exists (contexts) and attributes it contains. It is not necessary for the table structure supporting your alternative product hierarchy structure to exist within Oracle Applications.

Mapping an alternative product hierarchy can be accomplished using attribute mapping. For more information on attribute mapping, see Chapter 10, "Attribute Management".

Oracle Advanced Pricing Features

The following is a summarized list of Oracle Advanced Pricing features supported by Oracle Applications:

Qualifiers

Qualifiers determine who is eligible for a modifier. Qualifiers and qualifier groups can be linked to price lists and modifiers to define rules for who can receive a particular price, discount, promotion, or benefit. They can assign discounts and promotions to:

- Specific customers
- Customer groups
- Order types
- Order amount

Some example qualifiers are:

- Customer class = VIP
- Order type = Special

You can construct complex qualifier sets consisting of multiple qualifiers that are evaluated together in Boolean AND and OR relationships.

Qualifier Groups

Qualifier groups enable you to define multiple qualifiers relationships in preparation for association with either price lists or modifiers. You can save these qualifier groups and copy them to one or more price lists and modifiers.

Price Lists

Price lists relate a selling price to a product. Price lists contain one or more price list lines, price breaks, pricing attributes, qualifiers, and secondary price lists. The price list information includes the price list name, effective dates, currency, pricing controls, rounding factor, and shipping defaults such as freight terms and freight carrier. See the *Oracle Advanced Pricing User's Guide*, Price Lists chapter for information on setting up and using price lists.

Oracle Advanced Pricing provides you with the capability to instruct the Pricing engine make the selection of the price list based on a qualifier rule.

As an alternative to using price list qualifiers, you may default a price list on an order based on any one of the following:

- The sold-to customer
- The ship-to customer
- The bill-to customer
- Order type
- Agreement

Defaulting provides equivalent functionality to the price list defaulting found in Oracle Order Entry/Shipping Release 10.7 and 11.0. You can default some of the same elements you can reference in qualifier rules. Defaulting is available only for price lists. You can use only one default with a price list, and cannot combine defaults using AND/OR relationships.

You can define multiple price lists. Alternatively, you may enter a specific price list on the order header or at the order line level. For each price list, you can also

designate secondary price lists that the engine searches when it cannot find an item on the primary list. Multiple secondary price lists may be searched for each primary list. You may define several price lists as secondary to a primary price list. You can not define multiple levels of secondary price lists.

Price lists may be specified in different currencies. During order entry, if you enter a currency on the order, the pricing engine selects price lists with currency that matches the currency you entered on the order.

Price List Maintenance: Advanced Pricing provides search and maintenance capability for a single price list or across multiple price lists. Use the Bulk Change feature for mass updates or update individual price list lines.

Price List Maintenance is available through the HTML user interface. See the *Oracle Advanced Pricing User's Guide*, Using the Price List Maintenance feature for more information.

Pricing Attributes

Pricing attributes are data elements used in addition to item identifiers. These attributes control what is being priced on a price list or a modifier. Oracle Advanced Pricing is delivered with seeded pricing attributes. The seeded attributes for Oracle Advanced Pricing are the same as those for basic pricing. For more information on these seeded attributes, refer to *Oracle Advanced Pricing User's Guide.*

In some situations, an item identifier does not identify the product to a level where a price can be assigned. Pricing attributes enable you to define separate conditions for a product where the product must be priced under different conditions. A pricing attribute can be any condition you use to further qualify an item.

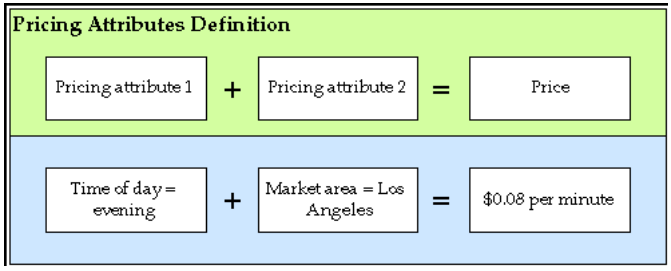
The following are examples of pricing attributes used to specify conditions of a product or service that affects the price. While these examples are appropriate to price lists, the same concept holds for modifiers. Pricing attributes can also be used with formulas.

- Product ID = HMO Plan 15; Pricing Attribute = State = Connecticut; List Price = $200.00 per month.
- Product = HMO Plan 15; Pricing Attribute = State = New York; List Price = $250.00 per month.
- Product ID = Motor Oil; Pricing Attribute = Grade = Regular; List Price = $3.50 per Quart
- Product ID = Motor Oil; Pricing Attribute = Grade = Premium; List Price = $4.25 per quart

- Product ID = Usage Charge; Pricing Attribute = Time of Day = Evening (7:00pm to 11:00pm); list price =.08 per minute
- Product ID = Usage Charge; Pricing Attribute = Time of Day = Late night (6:00am to 5:30am) list price =.05 per minute

Pricing attributes can be used in combination with each other and are passed to the pricing engine at run-time. The following image depicts an example of how to define two attributes, Time of Day and Market Area. Both of these attributes must be obtained by the calling application and then passed to the pricing engine at run time. The pricing engine uses the attributes to match to the combination of product ID and conditions defined by the pricing attributes.

Figure 1–1 Pricing Attributes Definition



Maintaining Price Lists

You can maintain price lists using any one of the following functions:

- Copy price list
- Adjust price list
- Add items to price list

These capabilities are invoked for a window within the Oracle Advanced Pricing Responsibility Menu, which then submits concurrent manager jobs for each step.

The Price List Maintenance feature (available from the HTML user interface) enables you to make changes to price lists and price list lines for a single price list or across many price lists. For more information, see *Oracle Advanced Pricing User's Guide*, Using the Price List Maintenance feature.

Agreements

Agreements enable you to define prices, payment terms, and freight terms that you negotiated with specific customers. You can:

- Define your agreements using customer part numbers and inventory item numbers.
- Make revisions to the original terms and maintain these changes and their reasons under separate revision numbers.
- Attach an already existing price list to the agreement or define new prices.
- Assign optional price breaks by quantity.
- Set effectivity dates for agreement terms.
- Set payment terms including invoice rule and accounting rule.
- Set freight terms including the freight carrier.
- Apply agreement terms to sales orders by reference agreements.

GSA Pricing

GSA Pricing enables you to define a GSA price list for your GSA customers. The GSA Price List actually uses the modifiers window and uses the new price. You create a discount that adjusts the base price of the item to the GSA price.

Formulas

Formulas enable a list price to be adjusted according to an algebraic relationship with other variables. Oracle Advanced Pricing enables pricing attributes to be passed as variables to the formula processing at run time. Formulas enable you to define a mathematical expression that the pricing engine uses to determine the list prices of items. A full complement of mathematical operators and numeric operands can be used. An example of a formula is:

List price of service call = (price from price list * distance) + adjustment factor

Distance is a numeric value passed to the pricing engine as a pricing attribute at run time. Adjustment factor is the result of a value in a factor table, based on class of service, which is a pricing attribute variable passed to the engine at run time.

When processing formulas, the pricing engine begins by locating a price list line which is linked to a formula. It then applies the mathematical expression to generate a final list price. In Oracle Advanced Pricing, formulas may be static; that is, the variables in the formula must be pre-populated with data by running a concurrent manager job before the formula can be used. Oracle Advanced Pricing

provides a dynamic mode of formula operation. In dynamic formulas, the required data to be substituted into formula variables is collected by the pricing engine at run time.

Modifiers

The term modifier is essentially synonymous with the term adjustment. Modifiers are pricing actions that, when applied to a transaction, adjust the selling price up or down. The specific action that a modifier takes is defined by its type. In Oracle Advanced Pricing, a modifier has two levels of functionality that define its action: a list type and a line type. A modifier list type enables you to define behavior characteristics that are common to all lines, that enables you to define a modifier with several different lines, each line representing a specific pricing action.

The following are the modifier list types supported in Oracle Advanced Pricing:

- Discount
- Surcharge
- Freight/Special Charges
- Promotion
- Deal

For each list type that you define, you can associate certain line types. The available line types are:

- Discount: Creates a negative price adjustment.
- Surcharge: Creates a positive price adjustment.
- Freight charge: Creates a freight charge.
- Item upgrade: Replaces a specific item ordered with another item for the same price.
- Terms Substitution: Replaces freight charges, shipping charges, and payment terms with more favorable charges.
- Other item discount: Gives a price adjustment or benefit to a specified item on an order when the customer orders one or more other items on the same order.
- Promotional goods: Adds a new item with a price adjustment or benefit when the customer orders one or more other items on the same order.
- Coupon issue: Issues a coupon on one order for the customer to redeem for a price adjustment or benefit on a later order.

- Price Break: Applies a variable discount or surcharge price adjustment to a pricing request based meeting the condition of a break type. You can use both point and range type breaks.

Not all line types can be used with all list types. See the *Oracle Advanced Pricing User's Guide* for a listing of valid modifier list and line type combinations.

You can define a modifier that the pricing engine automatically applies, or you can manually enter a modifier. With proper setup, modifiers can be defined as manual or overridable.

Modifiers can be used to compute price breaks. You can define breaks at the line level to be computed as percent, amount, or fixed price. Both point and range breaks are supported.

Freight and Special Charges

The freight and special charges capability of Oracle Order Management enables you to capture, store, update, and view costs associated with a shipment, order, container, or delivery. You can either itemize or summarize such charges on your orders. This capability includes functionality to pass customer charge information to Oracle Receivables for invoicing.

Freight and special charges enables you to:

- Apply charges as part of the order.
- Limit the application of charges to orders that meet certain criteria.
- Apply charges until the point of ship confirmation/invoicing.
- Review charges at anytime.
- Create many charge types (duty, handling, freight).
- Support charges at the order or line level.

When using freight and special charges, you set up freight and special charges as pricing modifiers. The pricing engine applies the qualified freight and special charges to order lines. You can view the application of freight and special charges to orders. Oracle Order Management captures costs at shipping and converts them to charges. Freight and special charges appear on invoices.

With Oracle Advanced Pricing, the full functionality of qualifier rules can be used to determine which orders should have freight and logistics charges applied to them, and to exclude certain orders from having charges. Additionally, formulas can be used to mark freight charges up or down before they are placed on the order.

Attribute Mapping

Oracle Advanced Pricing enables you to refer to data sources and attributes that are not seeded in the delivered product. This capability is called attribute mapping.

Pricing Security

Oracle Advanced Pricing provides an additional level of security called "pricing security" to enhance the existing functional security. Pricing security enables you to restrict pricing activities such as updating and viewing pricing entities to users granted specific access privileges.

Get_Custom_Price API

Oracle Advanced Pricing provides an API that enables the pricing engine to execute user supplied code to obtain a price, as an alternative to housing the price in a price list or in a formula. This gives you the capability to obtain a starting or beginning list price from sources that are outside Oracle Advanced Pricing's tables, yet are accessible to code that you write.

The Get_Custom_Price is called by the pricing engine while evaluating a formula that contains a formula line (step) of type Function.

The technical information necessary to use this API is documented in *Oracle Order Management Suite APIs and Open Interfaces Manual*.

Process Flow for Implementation

The process flows for implementing from a fresh install and implementing from an upgrade both assume that Oracle Applications, including Oracle Order Management has been successfully installed, that Oracle Advanced Pricing has been installed Shared, and that all necessary patches have been applied.

Implementing from Fresh Install

The following table lists the recommended steps for implementation, assuming a fresh install (no prior implementation of Oracle Order Entry/Shipping exists). The recommended implementation steps differ when upgrading from a prior release.

Table 1–2 Implementation Steps from Fresh Install

#	Process Step	Step Description
1	Analyze and understand business pricing scenarios.	An high understanding of pricing business requirements should be established before beginning an implementation of Oracle Advanced Pricing. For more information, see Chapter 3, "Implementation Methodology".
2	Determine data sources and columns needed for product and customer hierarchy definition.	Oracle Advanced Pricing ships with seeded values. You may need to define additional levels or alternative data sources. For more information, see Chapter 3, "Implementation Methodology".
3	Develop logical pricing model solutions.	Plan how you will use Oracle Advanced Pricing for each pricing scenario. For more information, see Chapter 3, "Implementation Methodology".
4	Set up and test prototype pricing solutions.	Prior to implementing a production system, you should set up Oracle Advanced Pricing prototype solutions for all the pricing scenarios you have identified, and enter test orders against them to determine that they are handled properly (the Vision Sample database shipped with the software can be used to for this).
5	Make defaulting decisions.	These decisions must be made prior to product setup. For more information, see Chapter 3, "Implementation Methodology".
5	Perform Oracle Advanced Pricing product setup tasks.	This step involves taking results of steps 1-4 and creating entries in the pricing system tables that enables Oracle Advanced Pricing to act on your plans. For a detailed task list, descriptions, sequencing, and instructions, see Chapter 2, "Implementation Overview".
6	Create backup of system as setup.	This involves creating a backup of system as setup.
7	Conduct pre-production system functionality and load tests.	Prior to initial production, a system test should be conducted. This test should exercise all product setups by including examples of all pricing scenarios defined during pre-implementation planning. Verify that testing includes volume to stress system to levels that are experienced during production operations.

Upgrading from Oracle Applications Release 10.7 or Release 11 to Oracle Advanced Pricing

When upgrading from either Release 10.7 or Release 11, the upgrade of pricing data to Release 11*i* initially establishes tables as basic pricing. The pricing upgrade occurs within the overall flow of upgrading from Order Entry shipping.

While the upgrade process establishes a working 11*i* system, because Oracle Advanced Pricing functionality is new, the upgrade process cannot create an 11*i* database that uses the capabilities of Oracle Advanced Pricing. The database created by the upgrade process has only data setups in it sufficient to drive basic pricing.

Rapid Implementation From an Upgrade

The following table depicts a process flow for a suggested rapid implementation process whereby planning and testing of Oracle Advanced Pricing implementation is performed prior to the conversion. The setup and activation of Oracle Advanced Pricing functionality proceeds rapidly after the upgrade is complete.

Table 1–3 Rapid Implementation Steps from an Upgrade

Pre- or Post-Conversion?	Process Step	Step Description
Pre-conversion	Analyze and understand business pricing scenarios.	Establish an exact understanding of pricing business requirements before beginning an Oracle Advanced Pricing implementation.
Pre-conversion	Determine data sources and columns needed for product and customer hierarchy definition.	See Implementation Methodology for more details. Oracle Advanced Pricing ships with seeded values. However, you may need to define additional levels or alternative data sources. For more information, see Chapter 3, "Implementation Methodology".
Pre-conversion	Develop logical pricing model solutions.	For each Pricing scenario, plan how you will use basic pricing. For more information, see Chapter 3, "Implementation Methodology".
Pre-conversion	Install 11*i* Oracle Advanced Pricing and calling applications in a test instance.	Software environment for testing should be available prior to pricing prototype.

Table 1–3 Rapid Implementation Steps from an Upgrade

Pre- or Post-Conversion?	Process Step	Step Description
Pre-conversion	Set up and test prototype pricing solutions	Prior to implementing a production system, set up prototype Oracle Advanced Pricing solutions for all identified pricing scenarios, and enter test orders against them to determine that they are handled properly. The Vision Sample database shipped with the software can be used to facilitate this process.
Pre-conversion	Plan pricing cut over policies.	Decide how to approach orders at the time Oracle Advanced Pricing setups are activated.
Conversion	Perform upgrade.	For more information, see *Oracle Order Management Suite Implementation Manual*.
Conversion	Perform post upgrade steps.	For more information, see *Oracle Order Management Suite Implementation Manual*.
Post-conversion	Begin product operation using basic pricing.	For more information, see *Oracle Order Management Suite Implementation Manual*.
Post-conversion	Perform Oracle Advanced Pricing product setup tasks.	Examine the results of planning and decision making defined in earlier steps. Create entries in the Pricing system tables that enable Oracle Advanced Pricing to act on your plans. For detail task list, descriptions, sequencing, and instructions, see Chapter 2, "Implementation Overview". Create these setups initially outside the production system.
Post-conversion	Conduct pre-production system functionality and load tests.	Conduct a system test prior to initial production. This test should exercise all product setups by including examples of all Pricing scenarios defined during pre-implementation planning. Testing should stress system to levels that are experienced during production operations. Initially, these tests should be performed outside the production system.
Post-conversion	Cut over to Oracle Advanced Pricing setups.	Set up testing completely, setup data is added to production system. Active flags set.

2

Implementation Overview

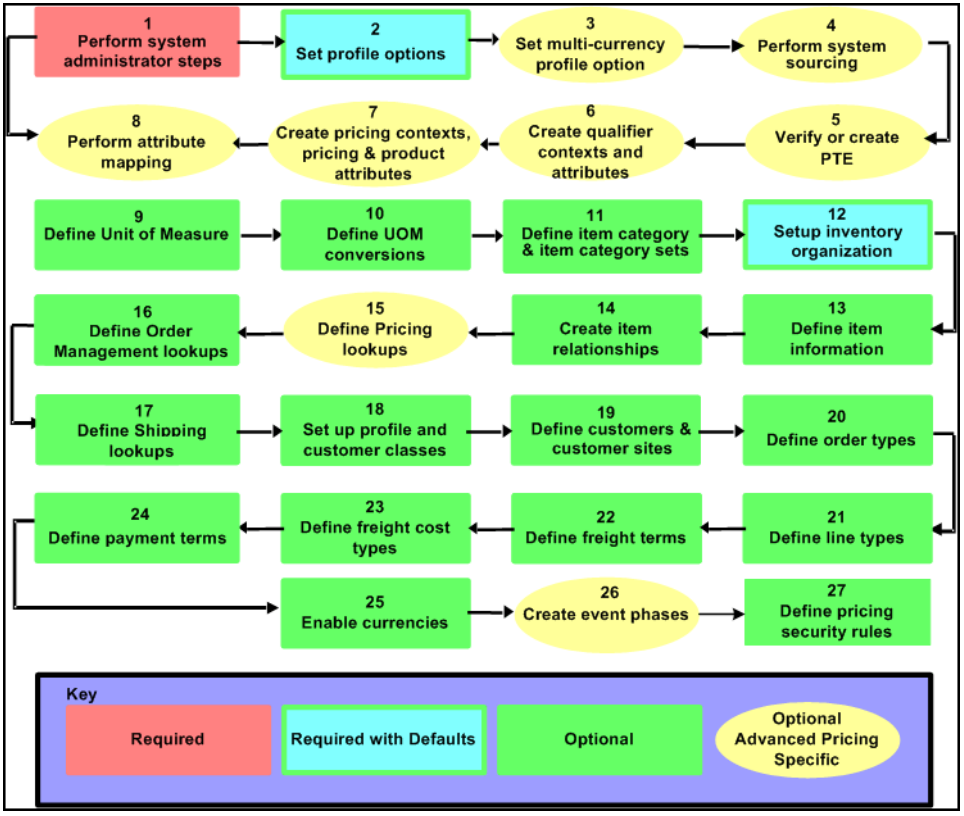
This chapter provides an overview of the necessary steps for implementing Oracle Advanced Pricing. The following topics are discussed:

Setup Flow

The following image depicts the setup flow for Oracle Advanced Pricing. Some of the steps outlined are required and some are optional. If you have already completed a common-application setup (setting up multiple Oracle Applications products) some of the following steps may be unnecessary.

A Required Step With Defaults is required only if you want to change the seeded default values. Review those defaults and decide whether to change them to better suit your business needs. Do optional steps only if you plan to use the related feature or complete certain business functions.

Figure 2–1 Setup flow for Oracle Advanced Pricing



Setup Steps

Step 1: Perform System Administration Steps

Default: None

Required

Assign users who set up Oracle Advanced Pricing to the Oracle Pricing Manager responsibility.

For more information on completing system administration steps, see *Oracle System Administrator User's Guide*, Responsibilities.

Do this step for each user who will access the Oracle Pricing Manager responsibility.

Step 2: Set Profile Options

Required

During implementation, you set a value for each user profile option to specify how Oracle Advanced Pricing controls access to and processes data.

The system administrator sets and updates profile values.

For more information on setting profile options, see: *Oracle Applications System Administrator's Guide*, Setting User Profile Options, and the *Oracle Advanced Pricing Implementation Manual*, Profile Options.

Step 3: Set Multi-Currency Profile Options

Default: None

Optional

The profile option QP: Multi-Currency Installed enables the multi-currency price list feature. Multi-Currency Price Lists enables you to maintain a single price list for multiple currencies. Once the profile option is set to Y, the price list and agreement forms are converted to multi-currency. Changing the profile option QP: Multi-Currency Installed back to N (No) may cause undesired results if conversion criteria were used. Oracle does not support this.

Step 4: Perform System Sourcing

Default: Predefined Oracle Advanced Pricing record

Warning: Changing the system source code can severely affect pricing engine behavior.

This step is required if:

- Your pricing data application source is anything other than Oracle Advanced Pricing.
- You are integrating Oracle Advanced Pricing with an application other than Oracle Order Management.

The pricing engine uses request types to determine the source of pricing data to be used when pricing a particular transaction. Request type is used to identify the type of transaction being priced. Whenever the pricing engine prices a request, the request must be stamped with the request type so the pricing engine can identify the type of transaction. The source system is recorded on all price and modifier lists and is used to identify which application created this pricing data.

Use the Pricing Transaction Entities Associations window to control which pricing data is used to price transactions.

Do this step for each source system you want to map to a request type.

Step 5: Verify or Create PTE

Default: Order Fulfillment

Required

A Pricing Transaction Entity (PTE) is an ordering structure that has associated Request Types and Source Systems. Request Types and Source Systems in the same PTE share pricing setup data. Additional Source System and Request Types can be added to existing PTE's.

There are very rare instances where it is necessary to create a new PTE. A new PTE needs to be created only if the new request type uses a different ordering structure and a different set of source systems that is not already predefined.

Step 6: Create Qualifier Contexts and Qualifier Attributes

Default: Predefined Oracle Advanced Pricing qualifier contexts

Optional

If you skip this step, users will be able to chose only predefined Oracle qualifier contexts and qualifier attributes for price and modifier eligibility.

Qualifiers provide a highly configurable and flexible method of defining the rules which your business uses to manage pricing. Qualifiers are used by the pricing engine to determine eligibility for price lists and modifiers.

Step 7: Create Pricing Contexts, Pricing Attributes, and Product Attributes

Default: Predefined Oracle Advanced pricing and product attribute contexts.

Optional

Pricing and product attributes are a feature of Oracle Advanced Pricing which enables you to define necessary item attributes in order to price or apply a modifier and attributes used in formulas.

If you do not do this step, users will be able to select only predefined Oracle Advanced Pricing contexts and associated attribute values for benefit options.

Step 8: Perform Attribute Mapping

Default: Predefined Oracle Advanced Pricing attribute mapping rules

Optional

Do this step if you have defined any additional qualifiers or pricing attributes in steps 6 and 7, or if you wish to change the defaulting mapping that has been seeded for a seeded qualifier or pricing/product attribute. Oracle provides predefined rules for attribute mapping, for both qualifiers and pricing contexts, that enable a list of values when selecting eligibility criteria.

Qualifier and pricing attribute mapping is required to supply a value for a qualifier or non-user entered pricing attribute before pricing a transaction. A mapping rule is set up to derive the value for the qualifier or pricing attribute from the transaction itself or from another attribute of the transaction. Attribute mapping builds additional information about a transaction that can be used to qualify for or derive a price, benefit, or charge for the transaction.

Attribute mapping includes rules that enable you to configure mapping to source qualifiers and pricing attributes according to your business needs.

Step 9: Define Unit of Measure

Default: None

Optional

Units of measure (UOM) are used in Oracle Advanced Pricing to determine the unit value for what the pricing engine is pricing, modifying, returning a benefit, or creating an accrual.

For more information on defining units of measure, see: *Oracle Advanced Pricing Implementation Manual*, Unit of Measure.

Do this step if you have not installed and set up Oracle Inventory or completed this common-applications setup for another Oracle Product.

Step 10: Define Unit of Measure Conversions

Default: None

Optional

You must define conversion rates between the base unit of measure and other units of measure within a UOM class if you want to price and discount an item in a UOM other than its primary UOM. Oracle Advanced Pricing uses these conversions to automatically convert transaction quantities to the primary pricing unit of measure defined on the price list when pricing cannot find a price in the transaction unit of measure. In addition, all price adjustments, benefits, and charges need to be defined in the same unit of measure as the unit of measure used on the price list.

For more information on defining unit of measure conversions, see *Oracle Inventory User's Guide,* Defining Unit of Measure Classes.

Do this step if you have not installed and set up Oracle Inventory or completed this common-applications setup for another Oracle Product.

Step 11: Define Item Category Sets and Item Categories

Default: Seeded structure name of item categories, category set inventory items, and associated default seeded category code combination MISC

Optional

Item categories have been seeded as an item attribute in pricing attributes and can be used for defining price lists lines and modifier lines. On the Price List and Modifier forms, the item categories are select in the product attribute field. The value set for the item category attributes uses the item categories defined in Oracle Inventory. Oracle Advanced Pricing uses only one flexfield structure. You can setup multiple categories and multiple category sets.

These categories and category sets should be defined for the defaulted standard flexfield structure. You can create other item categories in the product attributes

item context by creating a new attribute in the item context and attaching a value set.

For more information on defining item categories and item category sets, see *Oracle Inventory User's Guide,* Defining Category Sets and Categories, and *Oracle Application Flexfields User's Guide,* Key Flexfields in Oracle Applications, Item Categories Flexfield.

Do this step if you have not installed and set up Oracle Inventory or completed a common-applications setup for another Oracle product. If you do not plan on using Oracle category functionality for associated price or benefits, you can skip this step.

Step 12: Set Up Inventory Organization

Default: None

Required with defaults

You must define at least one item validation organization in Oracle Inventory. This is the organization that items are validated and viewed against when entering items in the Price List and Modifier Setup forms.

For more information on setting up inventory organizations, see *Oracle Inventory User's Guide*, Setting Up Oracle Inventory.

Do this step if you have not installed and set up Oracle Inventory or completed a common-applications setup.

Step 13: Define Item Information

Default: None

Optional

Define the items that you wish to price and discount and assign them to the validation organizations defined in step 10. If you want to define qualifier rules which include the seeded qualifiers Line Volume or Line Weight you must set the volume or weight attributes of each item as this is used by attribute mapping to derive the transaction line, weight, or volume.

For more information on defining item information, see *Oracle Inventory User's Guide,* Items.

Do this step if you have not installed and set up Oracle Inventory or completed this common-applications setup for another Oracle product.

Step 14: Create Item Relationships

Default: None

Optional

This step is required if you wish to give item upgrade benefits. You must define a Promotional Upgrade item relationship from the ordered item to the item you wish to give as an upgrade. Define your item relationships for the item validation organization.

Set up promotional upgrade items as follows:

- The ordered item and the promotional item need to have the same base unit of measure and unit of measure conversions.
- The modifier unit of measure and the pricing unit of measure on the order line need to be the same.

If those entities are not the same, the substitution can fail.

See *Oracle Inventory User's Guide,* Item Relationships

Step 15: Define Pricing Lookups

Default: Lookup type dependent

Optional

Lookup codes supply many of the lists of values in Oracle Advanced Pricing.

Lookup code values are the valid entries that appear in the list of values. They simplify information selection, and ensure that users enter only valid data into Oracle Advanced Pricing. You can add new lookup values at any time. You can set the Enable flag to No so that it will no longer appear in the list of values, or you can use start and end dates to control when a value will appear in a list. For a complete list of Lookup types, see: *Oracle Advanced Pricing Implementation Manual*, Lookups.

The following table depicts lookup types with descriptions:

Step 16: Define Oracle Order Management Lookups

Default: Lookup type dependent

Optional

For a list of valid default values for these lookups please refer to *Oracle Order Management User's Guide,* Lookups Appendix.

The following table depicts lookup types with their descriptions:

Table 2–1 Lookup types with their descriptions

Lookup Type	Lookup Description
Define sales channel	Required if you price, give benefits or charge by sales channel.
Define order categories	Required if you price, give benefits or charge by order category.
Define line categories	Required if you price, give benefits or charge by order line category.
Define order sources	Required if you price, give benefits or charge by order line category.
Define shipment priorities	Required if you price, give benefits or charge by shipment priority
Define ship methods	Required if you price or give benefits, including upgrading shipping method or charge by shipment method.

Do this step if you have not installed and set up Oracle Order Management or completed a common-applications setup.

Step 17: Define Shipping Lookups

Default: Lookup type dependent

Optional

For a list of valid default values for these lookups see *Oracle Shipping Execution User's Guide*.

Do this step if you have not installed and set up Oracle Order Management or completed a common-applications setup.

Step 18: Set Up Customer Class and Profile Class

Set Up Customer Class

Default: None

Required if you price, give benefits or charge by customer class.

For more information on setting up customer class, see *Oracle Receivables*, Defining Lookups.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Set Up Profile Classes

Default: None

Required if you price, give benefits or charge by customer account type.

For more information on setting up profile classes, see: *Oracle Receivables*, Profile Classes.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Step 19: Define Customers and Customer Sites

Default: None

Required if you price, give benefits or charge by customer.

For more information on defining customers, see *Oracle Receivables*, Defining Customers.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Define Customer Sites

Default: None

Required if you price, give benefits or charge by customer site.

For more information on defining customer sites, see *Oracle Receivables*, Defining Customer Sites.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Step 20: Define Order Type

Default: None

Required if you price, give benefits or charge by order type.

For more information on defining order types, see *Oracle Order Management*, Defining Order Types.

Do this step if you have not installed and set up Oracle Order Management or completed a common-applications setup.

Step 21: Define Line Type

Default: None

Required if you price, give benefits, or charge by order line type.

For more information on defining line types, see *Oracle Order Management*, Defining Order Line Types.

Do this step if you have not installed and set up Oracle Order Management or completed a common-applications setup.

Step 22: Define Freight Terms

Default: None

Required if you price, give benefits, including upgrading freight terms, or charge by freight terms.

For more information on defining freight terms, see *Oracle Order Management*, Defining Freight Terms.

Do this step if you have not installed and set up Oracle Order Management or completed a common-applications setup.

Step 23: Define Freight Cost Type

Default: None

Required if you price, give benefits or calculate charges using freight cost types.

For more information on defining freight cost types, see *Oracle Order Management*, Freight Cost Types.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Step 24: Define Payment Terms

Default: None

This step is required if you price, give benefits, or charge by payment terms.

For more information on defining payment terms, see *Oracle Receivables*, Defining Payment Terms.

Do this step if you have not installed and set up Oracle Receivables or completed a common-applications setup.

Step 25: Enable Currencies

Default: All major currencies predefined with Oracle Applications

Optional

The system administrator completes this step. The codes are ISO standard codes for currencies. You must enable the specific currencies you want to use on your price and modifier lists.

For more information on enabling currencies, see *Oracle General Ledger,* Currencies.

Do this step if you have not installed and set up Oracle General Ledger or completed a common-applications setup.

Step 26: Perform System Sourcing

Default: Predefined Oracle Advanced Pricing record

> **Warning:** Changing the system source code can severely affect pricing engine behavior.

This step is required if:

- Your pricing data application source is anything other than Oracle Advanced Pricing.
- You are integrating Oracle Advanced Pricing with an application other than Oracle Order Management.

The pricing engine uses request types to determine the source of pricing data to be used when pricing a particular transaction. Request type is used to identify the type of transaction being priced. Whenever the pricing engine prices a request, the request must be stamped with the request type so the pricing engine can identify the type of transaction. The source system is recorded on all price and modifier lists and is used to identify which application created this pricing data.

Step 27: Create Events and Phases

Default: Seeded Oracle Advanced Pricing phases

This step is required if you must create additional pricing phases or change the seeded pricing phases. If your business manages pricing requires certain types of benefits at a particular point or event in the transaction process flow, do this step.

For more information on phases and events, see *Oracle Advanced Pricing Implementation Manual*, What are Pricing Events?.

Step 28: Set Profile Options

Required

During implementation, you set a value for each user profile option to specify how Oracle Advanced Pricing controls access to and processes data.

The system administrator sets and updates profile values.

For more information on setting profile options, see *Oracle Applications System Administrator's Guide,* setting User Profile Options, and *Oracle Advanced Pricing Implementation Manual*, Profile Options.

Step 29: Set up Pricing Security

Optional

Oracle Advanced Pricing provides pricing security in addition to the existing functional security. Pricing security enables you to grant privileges that control users' access to pricing entities such as price lists, pricing agreements, and modifiers.

During implementation, the Oracle Pricing Administrator can set up pricing security for pricing entities as follows:

- Assign or reassign pricing entities to operating units.
- Set Global Usage to share or not share a pricing entity across operating units.
- Assign privileges to pricing entities to control who (the Grantee) can view or maintain the specified entity.
- Set up default security profile options that set the access privileges for new pricing entities. See Chapter 4, "Pricing Security" for more information on setting up security.

3

Implementation Methodology

This chapter discusses implementation methodology for Oracle Advanced Pricing. The following topics are covered:

Overview

The methods outlined in this chapter can be used by licensed customers of Oracle Advanced Pricing, Oracle implementation consultants, and Oracle pre-sales consultants.

Oracle Advanced Pricing Concepts

Before you develop a pricing solution model for customer requirements, you must understand how Oracle Advanced Pricing works. Pricing concepts are presented as an aid to understanding the functional capabilities of Oracle Advanced Pricing. There are four major pricing concepts. These are:

- Pricing rules
- Pricing actions
- Pricing controls
- Pricing extensibility

When using Oracle Advanced Pricing, you implement pricing concepts by setting up forms that describe your pricing rules, pricing actions, controls, and the links to any extensions you employed. These forms are called pricing objects.

Pricing Rules

In Oracle Advanced Pricing, the term pricing rules should not be confused with the same term from Release 11.0. In Release 10.7 and Release 11, pricing rules referred to developing a price with a formula. In Release 11*i* Oracle Advanced Pricing, the term pricing rules describes the rules-based logic the pricing engine uses to apply a pricing action to a transactions.

Pricing rules are built around the customer and product hierarchies. Pricing policies are often tied to the customer. Because customers belong to single or multi-level groups, Oracle Advanced Pricing enables you to define rules that associate pricing actions with a group or level of a group that a customer belongs to.

With Oracle Advanced Pricing you can define rules dependent on data other than the pricing hierarchy. For example, discounts are often progressive-based on volume.

In Oracle Advanced Pricing, you set up pricing rules using the Qualifier window to describe customer and non-product oriented rules. These qualifier objects are then attached to pricing actions. The product hierarchy definitions are contained in the

action objects. For example, if a price list calls for all items in item category A to be priced at $1.00, the product definition is setup as part of the Price List window rather than using the Qualifier window. If you wish to specify the same price list as usable only by the VIP customer class, specify the customer class in the Qualifier window.

Pricing Actions

Pricing actions are specific pricing activities that the engine applies to transactions. For example, the pricing engine can establish a list price by applying a price list action to an order line. Similarly, discounts that lower the list price down to a net selling price are a modifier action.

Oracle Advanced Pricing provides pricing actions that can be used to handle pricing problems.

Pricing Actions can be directly related to specific pricing forms, or objects. These objects are:

- Price lists
- Agreements
- Formulas
- Modifiers

Modifiers are a class of pricing action that modifies the net price either up or down. Modifiers are sometimes called adjustments. Modifiers have several types. Each of these modifier types applies a specific pricing action with specific behavior characteristics.

Pricing Controls

Pricing controls are used to control how pricing applies actions. For example, effectivity dates can be used to control when rules are in effect and when they are not. They can also be used to define when an action that a rule points to can be applied.

Pricing also supports several other types of controls. For example, you can setup phases, which are groups of pricing actions. You can then tie these phases to physical events in order management. Incompatibility groups are another example of pricing controls. You can assign modifiers to incompatibility groups such that only one modifier within an incompatibility group can be selected by the pricing engine.

There are many different controls in pricing. All are important, but some require more careful planning prior to implementation.

Pricing Extensibility

Pricing is often driven by customer factors, by trade customs, or by sales channel requirements. Almost no two companies implement pricing in the same way, even if they compete in the same industry.

To address this variability, Oracle Advanced Pricing is delivered with extensibility features that enable you to extend, in a non-invasive, low-cost manner, its operation to meet your business needs.

The extensibility features fall under two categories:

- Flexible attribute mapping
- APIs

Attribute Mapping

Oracle Advanced Pricing, when licensed with Oracle Order Management, Oracle iStore, or other Oracle products, provides you with defaults for the customer and product hierarchy and other commonly referenced pricing rule drivers. These can be used without extending the product.

For customers whose product or customer hierarchy is more complex, Oracle Advanced Pricing provides flexible attribute mapping. This feature enables you to add or change levels of the customer and product hierarchy from those that are delivered as seeded with the product. Indeed, you can use attribute mapping to substitute an alternative source and structure for either the product or customer hierarchy, rather than using those supplied as defaults when the product is installed.

You can use attribute mapping to link Oracle Advanced Pricing to data maintained outside Oracle Advanced Pricing and outside Oracle Applications. In addition to the customer and product hierarchy related data mentioned, you can use attribute mapping to make pricing read a currency rate from an external source used in a pricing formula computation.

APIs

Oracle Advanced Pricing delivers a set of APIs that you can leverage using your own code. This enables you to significantly extend the use of Oracle Advanced Pricing.

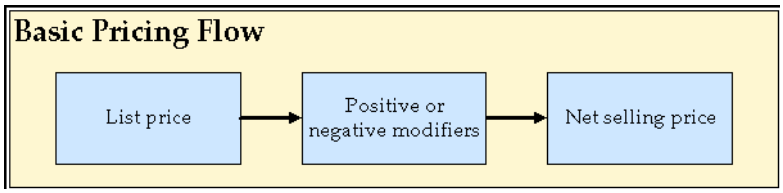
One API used for this purpose is Get_Custom_Price. This API can be linked to a price list using a formula with a term of type Function. You can use this API to make Oracle Advanced Pricing obtain a cost from a purchase order that is used in a calculation that yields a selling price based on actual cost.

Oracle Advanced Pricing APIs are documented in *Oracle Manufacturing Open Interfaces and API Manual.* For users who want to write code for these APIs, a downloadable ARU is available on Oracle Metalink that contains coding examples. The identifying ARU number is 1580332.

Basic Flow of Pricing

Oracle Advanced Pricing has a basic flow that it is important to understand. The following image depicts this flow:

Figure 3–1 Pricing Flow



Oracle Advanced Pricing prices any transaction by first identifying a list price. Price lists define a list price. Pricing determines whether the price found on the price list must be adjusted. Modifiers control the price of these adjustments, and can modify price either up (positive) or down (negative). Once the pricing engine has selected one or more modifiers that adjust the price, it computes the final price or net selling price.

Pricing Forms

Pricing is a collection of business pricing objects, each of which has defined functionality. These objects and their associated functionality are a set of tools that you can use to construct a pricing solution. You interact with these Oracle Advanced Pricing business objects by using Forms. Generally, each object has one or more forms that you must setup in order to obtain its functionality.

The following is an introduction to pricing forms and what they control. For further discussion of Oracle Advanced Pricing forms, see *Oracle Advanced Pricing User's Guide.*

Pricing forms are specific tools within the Oracle Advanced Pricing product used to provide pricing solutions. Each object has certain functions that it performs as part of the solution.

Methodology Steps

The recommended implementation methodology for Oracle Advanced Pricing consists of three steps:

- Analyze pricing needs.
- Develop pricing solutions.
- Test pricing solutions.

Each of these steps is discussed in this chapter, and important considerations for each are addressed.

Analyzing Pricing Needs

Developing a complete and correct understanding of the customer pricing requirements is the first step in implementing pricing. To develop a pricing solution with Oracle Advanced Pricing, you must break down the pricing requirements into their component parts. Once you analyze customer requirements, underlying pricing requirement elements, relationships between the elements, and calculations, you can associate each of these elements and their relationships with the correct business object. From this you can construct a pricing solution.

Define Pricing Requirements

For example, customers in the Northeast Region may order any two digital widgets and receive 10% off of either analog widget item ABC or DEF ordered at the same time. This is a promotional discount given for orders received between October 1 and December 31.

Analyze Pricing Requirements

Break down the requirement into the underlying component pricing requirements. This is accomplished through gathering enough information to precisely define the pricing action itself, and the conditions under which this action is to be taken. Break down the preliminary problem statement further:

- Distinguish between the digital widgets and other widgets. Digital widgets are discounted separately from analog widgets.

- The pricing solution must allow for either 1 or 2 of each digital widget to qualify for the free item.
- The pricing solution model must allow for adding multiple lines to the order to allow line 1 and 2 to be combined to equal the mix and match criteria.
- The 10% discount is taken off the selling price.

Create Rule and Action Statements

The next step is to develop rule and action statements that define the pricing actions to be taken and under what conditions that pricing action to be taken. First define the action:

Pricing action: Give a 10% discount taken off normal selling price. This action statement is independent of customer, product, or dates.

- Pricing rule: Discount action taken only for all customers in the Northeast Region.
- Pricing rule: Discount action applied only to analog widgets with item numbers ABC or DEF.
- Pricing rule: Discount action taken only when analog widgets are ordered at the same time as digital widgets.
- Pricing rule: Discount action taken only for orders received between October 1 and December 31.

Pricing Structures

In the example it is depicted how a pricing requirement expressed by an end user can be broken down into a pricing action statement and pricing rules that describe the conditions under which the action should be taken. Pricing policies may be defined in broader terms. In this case, before each action is defined along with its associated rules, it is necessary to understand the overall pricing structure of the company.

Define the Pricing Requirement

Rick's Souvenir Company sells two different consumer product lines. These products are sold through different sales channels. Rick's Pet Store sells

- Edible products
- Collectible products

Rick's edible products are sold through the following:

- Grocery store retail outlets
- Mass merchandiser retail outlets

Rick's collectible products are completely unrelated to the edible products, and are produced in a separate manufacturing facility. They are sold through the following:

- Hobby and toy stores
- Mass merchandiser retail outlets

Given the overall configuration of the business described, the pricing policies of the company are most likely closely tied to the structure. The following information is discovered after further investigation of Rick's Souvenir Company:

Edible Products

The edible products sold to retail outlets and mass merchandisers receive price breaks depending on the quantity ordered. For example, product 12345 San Francisco Rice Cakes is sold as follows:

Table 3–1 Pricing schedule for San Francisco Rice Cakes

Volume	Price Per Case
1-49	55
50 - 199	50
200 - 499	45
500 and over	40

All edible products are priced similarly for retail and mass merchandisers. All retail and most mass merchandisers also are subject to receive promotional discounts that are offered to induce higher volume and or build market awareness at certain times of year. As an example, promotional discounts are offered on San Francisco Rice Cakes for orders placed between August 1 and October 15 as an inducement to retailers to build up inventory of San Francisco Rice Cakes for Halloween.

One very large mass merchandiser customer, HugeCo Inc., has elected to eschew promotional pricing in favor of what they term every day low price. They negotiated a net price for each item. In the case of San Francisco Rice Cakes, this builds in an additional 10% off the price per case sold to HugeCo to reflect the theoretical average deal rate that other customers receiving promotional pricing receive for all promotions given throughout the year. This gives HugeCo a different pricing schedule for San Francisco Rice Cakes which is as follows

Table 3–2 Pricing schedule for San Francisco Rice Cakes

Volume	Price per case
1-49	50
50 - 199	45
200 - 499	40
500 and over	35

Collectible Products

The collectible products consist of decks of cards where each card has the picture of a popular sports figure. The decks of cards are oriented to particular sports, such as baseball, soccer, and football. Collectible sports cards are limited production runs, and are targeted towards consumers who collect and trade the cards. The pricing of one of the collectible card products, the New Millennium Set, varies depending on sales channel. No promotional pricing is used, and quantity discounts are not given. However, for various business reasons, collectible sports card products including item number 65432 New Millennium Set is priced higher to the hobby dealers than to mass merchandisers. A case containing 24 complete New Millennium Sets is priced at $375 to hobby dealers, and the price for mass merchandise outlets is $325.

Decompose the Pricing Requirement

Rick's Edible Products have three separate pricing policies or structures: one for grocery, another for mass merchandisers other than HugeCo, and one especially for HugeCo. Within the grocery and mass merchandisers (non-HugeCo) channels, there are two levels or pricing: list price and promotional pricing.

HugeCo also has two levels. In HugeCo's case, these levels are list price and everyday low price. Because HugeCo receives the everyday low price, it should not also receive promotional pricing.

Collectible sports cards have different selling channels each with a different pricing policy. These are hobby list price and mass merchandiser list price. Because there is no promotional price, HugeCo's everyday low price is the same as other mass merchandisers' price.

Create Rule, Action, and Control Statements

The following examples are limited to the two products already defined, edibles and collectibles: However, this same approach is used for defining other product pricing.

Edible Product

Pricing action: Give volume sensitive list price (see table for San Francisco Rice Cakes for prices at specific order volume levels).

- Pricing rule: For product San Francisco Rice Cakes.
- Pricing rule: For all customers belonging to retail customer class and to all customers in mass merchandiser customer class.

Pricing action: Give promotional discount of 10%.

- Pricing rule: For all customers belonging to retail customer class and to all customers in mass merchandiser customer class.
- Pricing rule: For product San Francisco Rice Cakes.
- Pricing rule: Exclude HugeCo from previous rule.
- Pricing rule: Apply to all orders received between August 1 and October 15.

Pricing action: Give HugeCo everyday low price schedule (see table).

- Pricing rule: For product San Francisco Rice Cakes.
- Pricing rule: For customer HugeCo.
- Pricing control: Apply to all orders for HugeCo received from January 1 to December 31.

Collectible Product

Pricing action: Give list price of $375 per case.

- Pricing rule: For product New Millennium Set.
- Pricing rule: For all orders for customers belonging to hobby dealers.

Pricing action: Give list price of $325 per case.

- Pricing rule: For product New Millennium Set.
- Pricing action: For orders for customers belonging to mass merchandiser class.

Pricing Terminology

In analyzing pricing requirement prior to implementing Oracle Advanced Pricing, it is important to consider that pricing terminology varies greatly across companies. Terminology such as tiered pricing, block pricing, off-invoice, price ceilings, price floors, and many others vary in their definition from one company to the next.

The best way to avoid misleading terminology is to insist that each term be decomposed down to its underlying pricing action and the rules that define the precise conditions under which that action is taken.

Analysis of Pricing Needs

In this next step, you examine creating and testing a pricing solution, and then consider key implementation decisions to be made.

Develop Pricing Solutions

Once you have defined your pricing actions and rule statements, you can define a pricing solution within Oracle Advanced Pricing. You begin defining the implementation solution by designing an overall structure for pricing. The following sections contain information necessary for decisions you must make when creating this structure.

Implementation Decisions: Single versus Multiple Currency Price Lists

Oracle Advanced Pricing supports both single currency price lists and multiple currency price lists. For single currency price lists, there is one currency defined per price list. For multi-currency price lists, a Multiple Currency Conversion list is attached to the price list defining multiple currencies and conversion factors and rules for converting prices.

It is very important to determine which price list strategy your organization supports. Advanced Pricing provides a profile option and concurrent program to convert existing prices lists from a Single Currency Price List to Multiple Currency price lists. However, once this profile is enabled, and price lists are converted, users should not return to NON Multi-Currency price lists. Changing the profile back to No may cause undesired results if conversion criteria was used. Oracle does not support changing the setting back to No.

For further discussion on using Multiple Currency Price Lists, see: *Oracle Advanced Pricing User's Guide*, Multiple Currency Price Lists.

Single Currency Price Lists

Single Currency Price Lists are the default setting for Advanced Pricing. These are used when your business requires that you maintain different prices for different currencies, and there is no relationship between prices defined.

Multiple Currency Price Lists

Multiple Currency Price Lists enables businesses that have pricing strategies based on a single price for an item in a base currency and use exchange rates or formulas to convert that price into the ordering currency. At engine run time, the pricing engine will take the currency from the order and search for a price list(s) with base or conversion currencies matching this currency. The pricing engine converts the price from the base currency and calculates the ordering currency based upon the established conversion rules.

Implementation Decisions Related to Customer and Product Hierarchy

Customer Hierarchy

Customer hierarchies are single- or multi-level groups with which a customer may be associated. Pricing rules are structured around these groupings which then affect the pricing actions a customer receives.

Key Implementation Decision: Are the seeded context values on Oracle customer tables sufficient to contain customer hierarchy?

When installed with Oracle Order Management, the customer hierarchy in Oracle Advanced Pricing is seeded to roll up individual customers according to the following structure, which is based on RA_Customer:

- Customer name
- Customer class
- Site
- Ship to
- Bill to
- Agreement name
- Agreement type
- GSA
- Sales channel
- Account type

If the seeded qualifier context values from the Oracle customer tables are adequate, then attribute mapping for customer hierarchy are not necessary.

If the seeded qualifier context values from the Oracle customer tables are not sufficient, there are two options for Release 11*i*. Both approaches requires you create attribute mapping to attribute mapping to inform pricing of the location of the customer hierarchy data elements you wish to reference in pricing qualifiers. You must create a default sourcing rule for each qualifier attribute that you define since the pricing engine uses sourcing rules to locate the attribute values. You can use expand the your hierarchies by:

- Use flex fields on customer table to hold flattened hierarchy.
- Use tables outside Oracle to house the customer hierarchy.

When defining your customer hierarchy, the lowest level in your hierarchy has the lowest flexfield segment sequence number. For example, the pricing engine must choose between a price for an item negotiated with a specific customer and a price for the same item given to all customers in a customer class. You want the engine to select the customer-specific price. Set the customer qualifier as more specific than the customer class qualifier, and the engine will select your desired price. This flattens your pricing hierarchy across the Qualifier descriptive flexfield. Each hierarchy may be defined within a context or across contexts depending on how many levels you have in your pricing hierarchies.

Trading Community Architecture (TCA) Attributes

Oracle Advanced Pricing enables you to use certain customer TCA attributes as qualifier attributes. Because modifiers can be set up based on TCA qualifier attributes, TCA qualifier attributes must be derived from Oracle Order Management attributes before calling Oracle Advanced Pricing. These attributes are seeded with the qualifier context as Customer context. The seeded qualifier attributes are:

- Party ID
- Customer account ID (same as sold_to_org_id)
- Ship to party site
- Invoice to party site

Party ID, ship to party site, and invoice to party site are visible to Oracle Order Management Users with Oracle Advanced Pricing even if they do not have Oracle Contracts. These attributes can be used as qualifiers to derive price lists and modifiers. Information about party ID, ship to party site, and invoice to party site are automatically derived by the sourcing rules. Therefore, customers do not see this information in any of the Oracle Order Management windows.

Product Hierarchy

Product hierarchies are single- or multi-level groups to which a product may be associated with. Pricing rules are structured around these groupings which then affect the pricing actions a customer receives.

Each level in your product hierarchy where your business sets its pricing and discounting should be defined as a product attribute in the seeded product context ITEM. Do this step after you specify your requirements to take best advantage of the flexibility of Oracle Advanced Pricing for managing pricing in your enterprise

The product hierarchy in Oracle Advanced Pricing is seeded with ITEM context based on the Oracle Applications Item Master, MTL_SYSTEM_ITEMS table. For this context, the flexibility of Oracle Advanced Pricing enables you to define your product hierarchy as follows:

- Item number
- Item category
- Item All
- Pricing attributes to specify a level more detailed than just item

When defining your product pricing hierarchy, the lowest level in your hierarchy has the lowest flexfield segment sequence number. If the pricing engine must choose between an item price and an item category price, set the item segment as more specific than the product group segment. This flattens the pricing hierarchy in the item context of your Pricing Attribute descriptive flexfield. Each level in your hierarchy at which you wish to manage pricing is represented as a segment.

Item categories and category sets

You must design and define any item categories and category sets for pricing and discounting. If you use the seeded context values on the Oracle item master, you can use only the predefined item categories context flexfield structure for the Item Categories Key flexfield within Oracle Advanced Pricing. However, you can change the default category code combination for the seeded inventory category. You are limited to the standard seeded item category structure only. Oracle Advanced Pricing does not consider any new structures that you set up for your Item Categories flexfield.

Pricing Attributes

Attributes define exactly what is being priced or modified. Attributes can be factors that affect the price of the item, additional definition that does not require creating an item, or discounting at a level higher than item in the product hierarchy.

Creating pricing attributes in different contexts enables attributes to be grouped according to their business. The item context is reserved for defining the product pricing hierarchy. Every level in the product hierarchy at which pricing and discounting is set should be defined as a segment in this context.

Key implementation decision: Can I use seeded context values on the Oracle Item Master Tables along with pricing attributes to define my hierarchy?

This decision depends on whether the item/item category and pricing attributes define your hierarchy (for levels above Item). If this is the case, attribute mapping is not necessary.

If the seeded context values are not sufficient, you have two options for Release 11*i*. Both approaches require you create attribute mapping to establish these contexts. This enables you to extend your product hierarchy.

- Use flexfields on item table to hold the flattened hierarchy.
- Use tables outside Oracle to house the product hierarchy.

If Oracle Advanced Pricing is installed without Oracle Order Management or the standard item master tables, you must define an alternative table structure location to support the product hierarchy with new contexts and attributes. The table structure does not need to reside within the Oracle Application tables.

Key implementation decision: Must I implement pricing attributes, multiple price lists, or both? What pricing and qualifier attributes do I need, and how do I decide which is which?

Complex pricing scenarios can be best be solved with a combination of pricing attributes and multiple price lists. Not only can a combination of the two be easier to understand and maintain, but it can improve engine performance.

Pricing attributes further control what is being priced or modified on a price list or modifier list. Each level in your product hierarchy should be defined as a pricing attribute in the seeded context, item. You must create a default sourcing rule for each pricing attribute that you define since the pricing engine uses sourcing rules to locate pricing attribute values.

For multiple price lists, qualifiers offer and/or capability as well as =, between, and not= operators.

For example:

- If you charge different list prices for the same item, depending on conditions in effect at time of order, you can use pricing attributes.

- If conditions are product oriented (example Item 123 Grade A is priced differently from same item Grade B) then product attributes are indicated
- If conditions are not product oriented, but depend on customer's membership in hierarchy or combination of customer and other factors like order type, time of day, then multiple price lists are indicated

Structuring Your Pricing Rules and Actions

Key implementation decisions: What qualifier statements and groups do I need? What customer groupings are required in order to develop qualifiers that implement my pricing rules? Which pricing actions are needed to process my scenarios? What product groupings are needed for my pricing actions?

The structure of your customer and product hierarchies is crucial in setup and implementation of Oracle Advanced Pricing, because these are the main source of your pricing rules.

Qualifiers are the primary window structure within pricing where you define the rules that determine which pricing actions are applied to a transaction. Although a qualifier can be used for any pricing rule, the qualifier window is generally used to define pricing rules related to attributes defined the customer hierarchy or to other elements not related to the product hierarchy. Although you emphasized that the product hierarchy is basic for pricing rules when you set up Oracle Advanced Pricing, you normally define the product hierarchy rule as part of the setup of the specific action.

When you implement Oracle Advanced Pricing, think in terms of structure for your pricing actions and qualifiers used to select those actions. Use the example of Rick's Souvenirs you used previously in the chapter.

Rick's Souvenirs has two broad product lines each sold through two sales channels. One of the channels overlaps. Pricing followed the channel lines, with the exception of one large customer. Draw some inferences as to how to structure pricing actions and rules based on this information.

Customer Groupings

By examining the pricing action and rules statements you can clearly see that two levels of customer hierarchy must be tied to pricing rules. These levels are:

- Customer class = grocery, mass merchandiser, hobby
- Customer name = HugeCo

Because customer class and customer name are both seeded elements of the customer hierarchy, no extension to the customer hierarchy is necessary.

Pricing Actions

The action/rule statements derived earlier show that the need for price list and promotional discounting actions that vary by product and are conditioned by channel.

Edible product

- Price list with breaks for grocery and merchandiser
- Promotional pricing discounts for grocery and mass merchandiser
- Alternative price list with breaks for HugeCo

Collectible Products

- Price list for hobby
- Price List for mass merchandisers

Product groupings: Lists are set for the individual products and in this case you have specific item numbers. However, the promotional discount of 10% applies to all edible products. Because item number and item category are both seeded elements in the product hierarchy you need not consider extending the product hierarchy.

Price Lists

Edible Product

Begin by structuring a single price list that handles list pricing. The following table depicts the price list.

Table 3–3 Price list for Edible Products price list example

Context	Attribute	Unit Price	Value	Pricing Attribute	Value To	Value From
Product	Item	55	12345	Volume	1	49
Product	Item	50	12345	Volume	50	199
Product	Item	45	12345	Volume	200	499
Product	Item	40	12345	Volume	500	9999999

In the previous example, you set up a price list action in Oracle Advanced Pricing that gives channel specific pricing for edible products.

Collectible Product

Now set up a second price list to handle the collectible product 65432 New Millennium Set.

Table 3–4 Example of Collectible Product price list

Context	Attribute	Unit Price	Value	Pricing Attribute	Value
Product	Item	$375	65432	Customer class	Hobby
Product	Item	$325	65432	Customer class	Mass Merch

The price list is qualified for both customer classes of both Mass Merch and Hobby. Because the price is different for the two classes, within the price list, you used customer class as a pricing attribute. This enables you to define the collectible item on the list twice, once for each channel.

Combined Edible and Collectible Price Lists

You structured two lists. The two price lists can be combined. The following is the combined price list:

Table 3–5 Example of Edible and Collectible price lists

Context	Attribute	Unit Price	Value	Pricing Attribute	Value To	Value From
Product	Item	55	12345	Volume	1	49
Product	Item	50	12345	Volume	50	199
Product	Item	45	12345	Volume	200	499
Product	Item	40	12345	Volume	500	9999999
Product	Item	$375	65432	Customer class	Hobby	Not applicable
Product	Item	$325	65432	Customer class	Mass Merch	Not applicable

Promotional Discount Modifier

Two adjustments are required for a total solution. First, promotional pricing for edible products that is given to both grocery and mass merch class customers. Second, HugeCo receives special everyday low pricing instead of the standard

pricing. A new, additional requirement is the everyday low pricing received by HugeCo must be accounted for as a discount.

First you need a modifier for promotional pricing. The promotional discount is the same percentage for the entire customer class. In the following table, the customer class is Mass Merch and Hobby for the edible products promotional discount.

Table 3–6 Modifier for promotional pricing

Context	Attribute	Value	Type	Method	Amount	Incompatibility Group
Product	Category	Edible	Discount	Percent	10	GRP01

An incompatibility group assignment is added because the modifier should not be used for HugeCo. Build the modifier for HugeCo:

Table 3–7 Incompatibility group assignment added to modifier

Context	Attribute	Type	Method	Unit Price	Value	Pricing Attribute	Value To	Value From	Incomp. Group
Product	Item	Discount	Amount	50	12345	Volume	1	49	GRP01
Product	Item	Discount	Amount	45	12345	Volume	50	199	GRP01
Product	Item	Discount	Amount	40	12345	Volume	200	499	GRP01
Product	Item	Discount	Amount	35	12345	Volume	500	9999999	GRP01

You defined a modifier for HugeCo that gives HugeCo a new price based on volume. By using a modifier action rather than a price list action, you met the finance requirement to account for the difference in list price and the everyday low price given to HugeCo as a discount. By using Application Method of amount you are substituting the unit prices on the modifier for the unit prices found on the price list shown in either example 1 or example 3 at each volume level. Because modifiers also support price breaks, you structured the HugeCo modifier as a price break.

Having accomplished this, you are ready to turn to your next implementation decision area: Pricing controls.

Establishing Pricing Controls

Pricing Controls describes a group of features in Oracle Advanced Pricing that enable you to specify how the pricing engine interprets your pricing rules and actions. From an implementation perspective, one is a decision that you should make in your implementation process.

Key Implementation Decisions: Must I implement incompatibility groups? How do I structure them?

In the example you used earlier in this chapter, by assigning the two modifiers for HugeCo to an incompatibility group, you can force the pricing engine to only choose one.

Must I use Oracle Advanced Pricing's exclusivity feature?

Exclusivity enables you to specify a modifier that, if selected by the pricing engine, is the only modifier applied to a transaction.

Do I instruct Oracle Advanced Pricing to use precedence or best price to decide between incompatible modifiers?

Oracle Advanced Pricing offers two methods to choose between incompatible modifiers: precedence and best price. Precedence is the usually the best choice.

Must I implement GSA Pricing?

GSA pricing enables you to establish a floor level price. It is used to comply with General Services Administration Pricing rules. GSA pricing operates as an integration with Oracle Order Management. For more information regarding GSA Pricing, refer to Chapter 16, "Technical Considerations".

Must I use multiple pricing buckets? If so how many levels do I need?

Pricing Buckets give you the capability to handle discounts with multiple subtotals.

Do the seeded pricing phase/Oracle Order Management event relationships work or must I reconfigure them?

Pricing modifiers and price lists must be associated with a specific pricing phase identifier. This enables application invoking the pricing engine to specify the phase identifier, which in turn guides the pricing engine to select only qualified pricing actions that set up in the phase. In Oracle Order Management, the integration with prices ties certain physical events in the order processing cycle to specific phases. This gives order management the capability to specify particular pricing phases to

the engine. The mapping of pricing phases to order management events can be configured by the user.

More detailed information on the pricing phases and how to use them, see: *Oracle Advanced Pricing Implementation Manual,* What are Pricing Phases? on page 12-4.

What effectivity dates must I establish?

Oracle Advanced Pricing provides very extensive effectivity dating capabilities. When the pricing engine is invoked by a calling application such as Oracle Order Management, it is passed a pricing date. The engine uses this date as a reference point to compare to effectivity dates found on various pricing forms.

The pricing date can be defaulted in Oracle Order Management. For specifics, see *Oracle Order Management User's Guide.*

What unit of measure considerations are necessary for Oracle Advanced Pricing setup?

If the Pricing engine attempts to determine a base price for a transaction, then the engine attempts to search qualified price lists for a price in the unit of measure that is in transaction line to be priced. If the engine finds such a UOM, the transaction UOM becomes the pricing unit of measure.

If that search is not successful, the engine searches for a conversion factor to convert the UOM on the incoming transaction to the primary UOM on the price list line.

The pricing engine returns only those modifiers defined in the same unit of measure as the pricing unit of measure, or where there is no product UOM specified. The latter may be the case if the modifier is not product specific.

Verify that all modifiers and price lists used together have common units of measure. Verify that if the unit of measure on the price list and modifier is different from the ordering UOM, that a conversion factor has been set up.

Do I want to control spending of Promotions?

Promotional Limits functionality enables you to set maximum values for benefits that a customer can receive for a promotion, deal or other modifier. By limiting the amount of a benefit that can be received, you can keep promotional spending within budget and prevent promotion budget overruns.

For more information on Promotional Limits, see: *Oracle Advanced Pricing User's Guide*.

Testing Your Pricing Solutions

The final step in the Oracle Advanced Pricing implementation methodology is to test your pricing solutions.

1. If you upgraded from Release 10.7 or 11 to Release 11*i*, and are running Oracle Advanced Pricing in Basic Mode with upgraded data, establish a separate test instance of your system and do your testing there.
2. Verify that you tested each scenario. Develop a documented script for each scenario that describes the setup and expected result, and then gives a step by step outline.
3. Begin with simple scenarios, and then work your way up to more complex ones.
4. Verify that you receive your expected numerical results.

   For additional information on debugging and troubleshooting, see: Diagnostics and Troubleshooting on page 16-31. For information on performance, see: Oracle Advanced Pricing Setup Considerations on page B-4.
5. Once your setups work in your test environment, you can begin replicating these setups into your production environment. Verify that the price list and modifier flags are set to inactive. Select beginning effectivity dates with caution, so as to avoid premature pricing engine use of new setups for pricing production transactions.

4

Pricing Security

This chapter describes the steps required to implement pricing security for Oracle Advanced Pricing. The following topics are discussed:

Overview of Oracle Pricing Security

In Oracle Applications, a basic level of security called *functional security* is used to manage users' access to each application and control their access to windows, functions, and reports within an application.

Typically, the System Administrator administers functional security and assigns operating unit, responsibility, and system access to users. See the *Oracle Applications System Administrator's Guide* for more information about functional security.

Oracle Advanced Pricing provides an additional level of security called *pricing security* in addition to the existing functional security. Pricing security enables you to restrict pricing activities such as updating and viewing pricing entities to users granted specific access privileges.

Pricing security can be set up and maintained by a user who is assigned the Oracle Pricing Administrator responsibility. Pricing security is set up and maintained in the HTML user interface. The Oracle Pricing Administrator has the authorization to access and update all pricing entities for all functional users. Pricing entities include price lists, pricing agreements, and modifiers.

With pricing security, you can implement a higher level of control by:

- Assigning pricing entities to operating units.
- Assigning privileges to pricing entities that control who (the Grantee) can view or maintain the specified entity.
- Setting default security access rules for new pricing entities with security profile options.

Assigning pricing entities to Operating Units

A pricing entity can be assigned ownership to a specific operating unit. You can restrict usage to one operating unit or allow usage by all operating units.

Assigning security privileges to control users' access to pricing entities

You can use security privileges to control users' access to pricing entities in the following ways:

- Grant view-only or maintain access privileges to functional users at the Global, Operating Unit, Responsibility, or User level.
- Grant temporary access - for example, to auditors or temporary employees - for a specified date range.

- Assign or reassign Operating Unit ownership to price lists and modifiers and control which operating units can use them for pricing transactions.
- Create Entity Sets (a set consists of grouped pricing entities) and assign access privileges to the entire set. The Entity Set function is only available with license to Advanced Pricing.

> **Warning: Before setting the profile option QP: Security Control to ON, you must create privileges for existing pricing entities.**

Pricing Security Terminology

The following terms are used in Oracle pricing security:

Pricing Entity Security The highest level of security administration for Oracle Pricing. This level of security is in addition to Functional Security and PTE plus Source System Code security. Functional security is established for each user by responsibility set up. The Oracle Pricing Administrator is a new Responsibility which has complete access to all pricing entities without restriction and is used for global administration of secured access to pricing entities. This security is administered in the Oracle HTML user interface.

Pricing Entity: A pricing entity can be a price list, modifier list, or pricing agreement.

Entity Set: A set of pricing entities that can be used as an Entity Type to which you can grant privileges with Maintain or View-Only access levels.

Entity Type: A term used to describe one of the following pricing entities: Standard Pricelist, Modifier List, Pricing Agreement, and Entity Set.

Entity Usage: Grants the entity's usage to one or all operating units so it can be used during pricing engine calls.

Global Usage: When Global Usage is set to Yes for a pricing entity, it can be used across all operating units for processing orders. If No is selected, the entity's usage is restricted to the operating unit that created or owns it.

When security is turned on, a Global box indicating Global Status is dynamically added to the header region of all price lists and modifiers. A user with Maintain access privileges can update the Global box. The Oracle Pricing Administrator can also update the Global Usage settings in the Entity Usage pages.

Grantee: The specific user or users for a Grantee Type that are given permission to view or maintain a pricing entity. Used in combination with a Grantee Type.

Grantee Type: The level to which privileges are granted:

- Global: Includes all users with access to pricing menus.
- Operating Unit: Includes users within the named operating unit.
- Responsibility: Includes users within the named responsibility.
- User: Specifies a named user.

Access Level: Provides Maintain or View-Only access to a pricing entity:

- View-Only: Enables the user to view but not update the pricing entity.
- Maintain: Enables the user to view and update pricing entities. Not all of the entities support delete capabilities.

Getting Started

What happens after the upgrade to pricing security?

After you upgrade to pricing security, pricing security is not switched on automatically. Pricing users with functional access can still fully view and maintain existing price lists and modifiers as before the upgrade.

Before turning security on, it is recommended that you review and complete the following setup steps for implementing pricing security - **otherwise, pricing users may be unable to query any price lists or modifiers in the pricing windows**. After you have completed the security setup steps, you can set the QP: Security Control profile option to ON.

> **Note:** You must be assigned the Oracle Pricing Administrator responsibility to set up and maintain the pricing security features for all functional pricing users.

Setup Steps for Implementing Pricing Security

After you have upgraded to pricing security, complete the following steps to successfully set up and use pricing security. **Do not just turn the QP: Security Control profile option to ON without completing the implementation steps and guidelines**. Otherwise, no price list or modifier list will be visible in the system until you grant a usage to the entity.

Step 1: Mapping Complete Security Access Requirements

Identify and map all price lists, modifiers, and agreement price lists to:

- Operating units that should own and maintain them.
- The users in those operating units who require View-Only or Maintain access (view and update) to pricing entities.
- Operating units that can use them when pricing transactions.

Step 2: Assigning Ownership of Pricing Entities to Operating Units (Entity Usage page)

The next step is to assign pre-existing price lists and modifiers to an operating unit. You can also select Global Usage settings that determine if the entity is restricted to that operating unit or available across all operating units. See "Assigning

Ownership of Pricing Entities to Operating Units (Entity Usage page)" on page 4-10 for more information.

Step 3: Creating Privileges (Privileges page)

The next step is to create all the access privileges for all users in all operating units. Based on your mapping of users, you can assign security privileges to grant view or maintain access to a pricing entity. See "Creating Privileges" on page 4-17 for more information.

Step 4: Creating Entity Sets (Entity Sets page)

The next step is to create entity sets. (This is an optional step.) You must have a license for Advanced Pricing to use this feature. Entity sets make it convenient to group multiple entities of the same entity type by specified criteria, then grant access to the entity set.

For example, you may want to create a set called Summer Set that contains all active modifiers with Summer Promotion in the modifier name. Then you can assign privileges to the entity set in the Privileges page. See "Creating Pricing Entity Sets" on page 4-27 for more information.

Step 5: Setting up Default Security Profile Options for New Pricing Entities

You can use the following profile options to set the default security privileges for newly-created pricing entities:

- QP: Security Default ViewOnly Privilege
- QP: Security Default Maintain Privilege

These profile options are delivered in default settings that maintain the existing functional security features of Oracle Pricing.

Before changing these profile settings, the Oracle Pricing Administrator must map the complete security access requirements for each pricing entity. No security profile option should be changed until these steps have been completed. See "Setting up Default Security Profile Options for New Pricing Entities" on page 4-41 for more information.

Step 6: Setting the QP: Security Control profile option to ON

The QP: Security Control profile option is the "switch" that turns security on or off for your installation. Before setting the profile option QP: Security Control to ON, it is recommended that you have completed all the preceding implementation steps.

See "Setting the QP: Security Control profile option to ON" on page 4-41 for more information.

Changes to Pricing windows after Upgrading and Turning Security On

This section summarizes the changes that occur to pricing entities after you upgrade to pricing security and turn security on. Some of the changes, such as the new Global box on price lists and modifiers, are only visible to users after pricing security is turned on.

Changes to Existing Pricing windows

After the upgrade to security, all existing price lists and modifiers are assigned the default entity usage of Global Usage. Global usage enables the pricing entity to be used across all operating units.

When pricing security is turned on, a Global box - which identifies the global usage status - is dynamically added to the header region of all price lists and modifiers. The Global box is visible to end-users and can be updated (cleared or selected) by users with Maintain access privileges.

Changes to Price Lists

After the upgrade, you can review the operating unit and global usage settings for an entity in the Entity Usage page. The following table outlines the information that displays:

Table 4–1 Default Entity Usage after Upgrade: Entity Usage page

Entity Name	Type	Global Usage	Owned by Operating Unit
Name of the Entity (for example, Summer Pricelist)	Type of Entity (for example, Standard Pricelist)	Yes	Blank (not assigned to an operating unit)

The following other changes occur to price lists after the upgrade to pricing security:

- A price list assigned Global Usage can be selected by the pricing engine even if the price list has been assigned to a specific operating unit.

  If the price list is unavailable for global usage (for example, a user clears the Global box in the price list header), then the pricing engine will select this price

list only if the current operating unit is the same as the one that created the price list.

- Once security is turned on, all new price lists have their view and update properties determined by the pricing security profile options.
- Users who have view-only privileges on a price list as per pricing security rules will be in view-only mode on the price list window. To update a price list, the user requires specific maintain-access privileges.
- The Public API, QP_PRICE_LIST_PUB.PROCESS_PRICE_LIST will only update price lists as per price list security rule.
- Users selecting using Price Lists > Copy Price Lists can copy price lists. A copied price list is assigned the default privilege from the security profile options. The copied price list will belong to the operating unit of the user who created it, regardless of the operating unit assigned to the original price list.

Changes to Modifier windows

- A modifier assigned Global Usage can be selected by the pricing engine even if the modifier has been assigned to a specific operating unit.

  If the modifier is unavailable for global usage (for example, a user clears the Global box in the modifier header), then the pricing engine will select this modifier only if the current operating unit is the same as the one that created the modifier.
- After pricing security is turned on, the default view and maintain properties for all new modifiers are determined by the security profile options.
- Users need at least view-only access privileges to display or query modifiers in the Define Modifier window. A user with view-only access privileges can view all list and line limits for a modifier including attributes and transactions for the limit.
- Users with view-only access privileges cannot modify the header information, lines, list or line qualifiers, pricing attributes, and related modifier information. A message or hint will display to notify the user about the view-only status.
- Modifier lines of the type Promotional Goods can attach to price lists that are viewable, as per pricing security, in the Get Price column list of values (LOV) in the Get region.
- The Public API, QP_MODIFIERS_PUB.PROCESS_MODIFIERS will only update modifiers as per modifiers security rule.

- In the Modifier Incompatibility Setup window, only those modifier lines belonging to a modifier list that can be viewed or maintained will get queried as per pricing security rules. Modifiers opened by clicking the Modifiers button may be viewed or maintained depending on the privileges defined by the Pricing Security Administrator.
- A copied modifier will inherit the default privileges set by the security profile options. The copied modifier will always belong to the operating unit of the user that created it, regardless of the source operating unit.

Changes to Order Management

All price list (LOV) list of values in Oracle Order Management will call the Pricing API, Get_Pricelists(), to return a list of valid price lists. The API returns the price lists owned by the same operating unit as the operating unit of the current user and those price lists where the Global box is selected.

Changes to other Pricing windows

The following table outlines the impact of pricing security and security privileges on various windows in Advanced Pricing:

Table 4–2 Impact of Pricing Security on Advanced Pricing windows

For the following:	Security Privileges are enforced:
Copy Price Lists	Yes. User needs at least view-only access.
Copy Modifier	Yes. User needs at least view-only access.
Modifier Incompatibility setup	Yes, can be updated if user has maintain access.
Pricing Organizer	Yes. User with view access can view if modifier displayed.
Pricing Mass Maintenance	Yes. User needs maintain access.
Adjust Price List	Yes. User needs maintain access.
Add Items to Price Lists	Yes. User needs maintain access.
Multi-currency conversion	No security at present.
Formulas	No security at present.
Agreement Header	Agreement inherits security rules of attached price list.
Price List report	Yes. User must have at least view-only access.
Modifier Detail report	Yes. User must have at least view-only access.

Assigning Ownership of Pricing Entities to Operating Units (Entity Usage page)

After the upgrade to pricing security, each newly-created price list and modifier is assigned to the operating unit of the user who created it. Assigning operating unit ownership to pricing entities restricts usage of that entity to within that operating unit. This prevents the use of an entity across all your operating units.

Since pre-existing price lists and modifiers are not assigned a default operating unit, the Oracle Pricing Administrator can:

- Assign or reassign ownership of pre-existing price lists and modifiers to the appropriate operating unit.
- Grant or revoke Global Usage of pricing entities which enables the pricing entity to be accessed across all operating units.

Warning: It is recommended that the Oracle Pricing Administrator assigns ownership to all price lists and modifiers prior to upgrading or implementing Oracle Pricing Security. This can be done using the Bulk Update Entity Usage feature in the Entity Usage page.

Global Usage and the Global box

In the Entity Usage page, the Global Usage column for a given pricing entity reflects the status of the Global box on the price list and modifier windows.

When global usage is enabled for an entity, that entity can be shared across operating units - its usage is not restricted to the assigned operating unit.

When security is turned on, a Global box is added to the header of all modifiers and price lists to indicate the global usage status for the entity:

- If selected, global usage is enabled for the entity.
- If not selected (cleared), global usage is not enabled for the pricing entity, and the entity's usage is restricted to the assigned operating unit.

The Global box, which is not visible to users until the profile QP: Security Control is turned on, is not protected at the window level from a user with Maintain privileges; so a user with Maintain access privileges can select or clear the Global box. However, users with view-only privileges cannot change the Global box.

If a user creates a new pricing entity (such as a price list) and clears the Global box, then the new entity can be used in pricing only by the operating unit of the creating user. If the Global box is left selected (the default value), then the entity can be used across all operating units when pricing transactions, even though the creating operating unit owns it.

Alternately, the Pricing Administrator can also update the Global box for one entity at a time or in bulk using the Bulk Update Entity Usage page available from the Entity Usage page.

Warning: Pricing users with Maintain access should be advised of the impact of clearing or selecting the Global box.

Default Ownership for Newly Created Pricing Entities

Each price list and modifier that is *newly* created is assigned a unique system-generated Operating Unit identifier. When querying newly-created pricing entities in the Pricing Security pages, the operating unit ownership is displayed in the Owned By Operating Unit field.

Creating Pricing Entity Usage

Complete the following planning and implementation steps before turning on pricing security. This mapping should be completed by someone with complete knowledge about the various pricing users and their operating units, all price lists, modifier lists and any specific business requirements for granting access to any of the many pricing entities.

When mapping your current and proposed security requirements, you may want to prepare separate listings for each entity type: Standard price list, Agreement price list, Modifier, and Pricing Entity Set. Based on the security policy of your organization, the Oracle Pricing Administrator can grant access privileges to the pricing entities, once entity usage has been set up.

For each entity, decide whether:

- The pricing entity can be used by all operating units in pricing transactions.
- The pricing entity should only be used by a specific operating unit.

Identify which entities are to be used across all operating units

For each entity, decide whether:

- The pricing entity can be used by all operating units in pricing transactions.
- The pricing entity should only be used by a specific operating unit.

Identify which entities are to be restricted to only one operating unit

For each entity, decide whether:

- The pricing entity can be used by all operating units in pricing transactions.
- The pricing entity should only be used by a specific operating unit.

Pricing entities used by multiple operating units

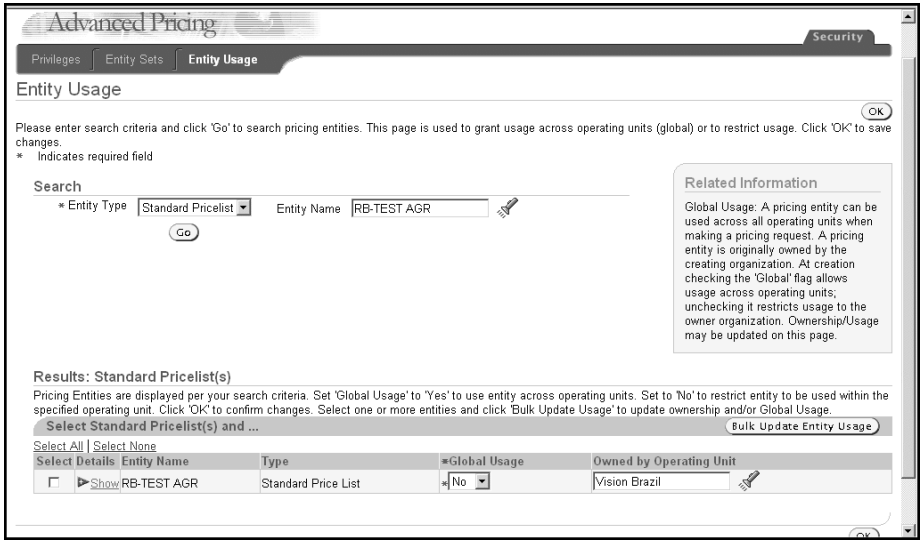
For pricing entities that can be used by *multiple* operating units but not *all* operating units:

- Select Yes for Global Usage.
- In the Pricing Entity page, create qualifiers for the specific operating units.

To create pricing entity usage:

1. Navigate to the Entity Usage page where you can:
   - Assign operating unit ownership to pricing entities such as price lists and modifier lists.
   - Assign Global Usage values of Yes or No.

Figure 4–1 Entity Usage page



2. In the Search region, select your search criteria:
   - Entity Type: Select an entity type such as Standard Pricelist or Modifier.
   - Entity Name: Optionally, enter an Entity Name to search for a particular price list or modifier.
3. Click Go to display the search results in the Results region of the page. For each listed entity, the following information is displayed:
   - Details: Click the Expand icon to view additional details about the selected Entity such as its Active Status, Start and End Dates, Description, and Currency.
   - Entity Name: Displays the unique name that identifies the selected entity.

- Type: Describes the Entity Type selected such as Standard Price List or Modifier.
- Global Usage: Indicates the current usage status of the pricing entity.
- Owned by Operating Unit: Displays the name of the Operating Unit associated with the Entity.

Note: For fresh upgrades or installations of Oracle Advanced Pricing, the Global Usage is Yes and Owned by Operating Unit fields are blank.

To update Operating Unit and Global Usage for a pricing entity:

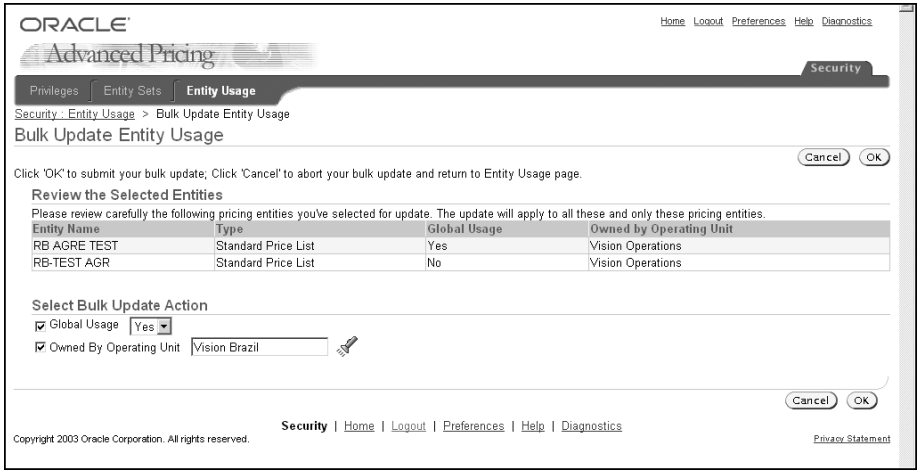
1. For each pricing entity listed in the Results region, you can assign a Global Usage and Owned by Operating Unit value. To make bulk changes to multiple pricing entities, use the Bulk Update Entity Usage feature. See "Using Bulk Update Entity Usage" on page 4-15 for more information.
2. To make the entity available across all operating units, select Yes for Global Usage. Alternately, select No to restrict the entity's use to within the specified operating unit.
3. Select the operating unit in the Owned by Operating Unit field.
4. Click OK to save your changes.

Using Bulk Update Entity Usage

Use the Bulk Update Entity Usage page to quickly apply the same changes across selected pricing entities; for example, to assign the same operating unit across all price lists.

1. Select the pricing entities from the Results region. Alternately, to select all pricing entities on a page, click Select All. If additional entities are listed on subsequent pages, click the Next link, then click Select All. Repeat this process until all the entities to be updated are selected.
2. After the pricing entities are selected, click Bulk Entity Usage to display the selected entity names in the Bulk Update Entity Usage page.

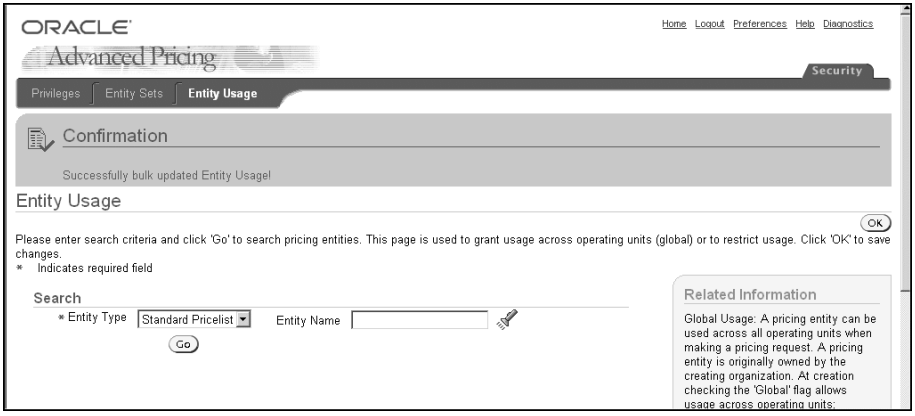
Figure 4–2 Bulk Update Entity Usage page



Review the pricing entities listed in the Review the Selected Entities region. Any updates will apply to the listed pricing entities.

3. In the Select Bulk Update Action region, select the Global Usage box and select Yes or No to update all the entities global usage to Yes or No.
4. Select the Owned by Operating Unit box to update all the entities with the specified Operating Unit.
5. Click OK. If successful, a Confirmation message advises that you have successfully bulk updated the entity usage.

Figure 4–3 Bulk Update Entity Usage: Confirmation page



6. Click Ok to save your changes.

Creating Privileges

Security privileges define who can access each pricing entity and the level of access permitted: View Only or Maintain.

You must be assigned the Oracle Pricing Administrator responsibility to grant the following privileges:

- Grant access privileges to functional users at the Global, Operating Unit, Responsibility, or User level:
  - Global: Includes all users with access to pricing menus.
  - Operating Unit: Includes users within the named operating unit.
  - Responsibility: Includes users within the named responsibility.
  - User: Specifies a named user.
- Grant Access level of View Only or Maintain.
- Grant temporary access - for example, to auditors or temporary employees - and give them automatic Start and End effective dates.
- Using Entity Set, you can define rules for a pricing entity that will restrict a specific user or group of users to accessing entities that are for a specific customer, group of customers or a specified customer hierarchy level. See "Creating Pricing Entity Sets" on page 4-27 for more information.

You can define rules for a pricing entity that will restrict a specific user or group of users to accessing entities for a specific customer, group of customers or a specified customer hierarchy level.

You can assign privileges using the following setup pages:

- Privileges page: To search for and update existing privileges.
- Express Create Privilege page: To create an access privilege for one specific pricing entity.
- Bulk Create Privileges page: To select multiple pricing entities and create access privileges for a grantee.

Precedence Levels for Multiple Privileges

A user may have access privileges by virtue of their Responsibility. If the user has View Only access to a pricing entity by virtue of their Responsibility, but requires Maintain access, a Maintain privilege can be granted to the user. A Maintain access

privilege is a higher privilege than View Only, and therefore, the higher Maintain privilege prevails for the named user.

If a user has a Maintain access privilege to a given entity at any level of their user hierarchy (responsibility, operating unit, and Global), they will have Maintain access regardless of any other privileges.

For example: if a user has Maintain access at their operating unit level but a view-only access at their user level, their Maintain access privilege will have precedence.

Implementation Suggestions for Privileges

Complete the following planning and implementation steps before turning on pricing security. This mapping should be completed by someone with complete knowledge about the various pricing users and their operating units, all price lists, modifier lists and any specific business requirements for granting access to any of the many pricing entities.

1. Identify and list all users with Functional Access to Advanced Pricing Menu

Identify all Responsibilities within your installation that have functional access to the Oracle pricing menus. This will assist in determining whether a pricing entity can be granted access by users with these Responsibilities. When an access privilege is granted at the Grantee Type: Responsibility, then all users with this responsibility will have this privilege.

Add to the listing of all Responsibilities with access to pricing menus, all individual users, by name. Some users may not require Maintain privileges to any pricing entities, but may actually require view-only access. These users should be identified and associated to the pricing entities to which they require view access.

This mapping assists in granting an access privilege to a specific user. A user may have access privileges by virtue of their Responsibility. If the user, whose Responsibility has been granted an access privilege of ViewOnly to a pricing entity, needs to have Maintain access, a privilege may be granted to the user for Maintain which is a higher privilege than that granted to his or her Responsibility.

2. List all users by new access privileges

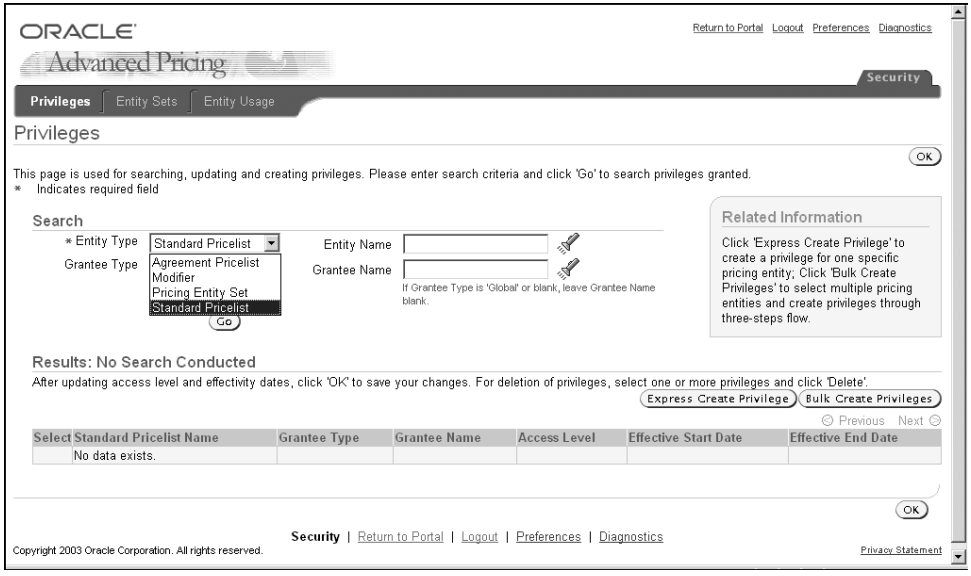
It is recommended that a listing of all users and their access privileges be maintained by the Pricing Administrator. Once mapping has been completed and access privileges granted, you can query the privileges granted in a variety of ways

using the Privileges page of the Security pages. A search by Entity Type such as Standard Price List displays all Standard Price Lists by Entity Name, Grantee Type, Grantee Name, Access Level (ViewOnly or Maintain}, and Effective Dates. Your listing of new access privileges can be checked against the results.

To create privileges:

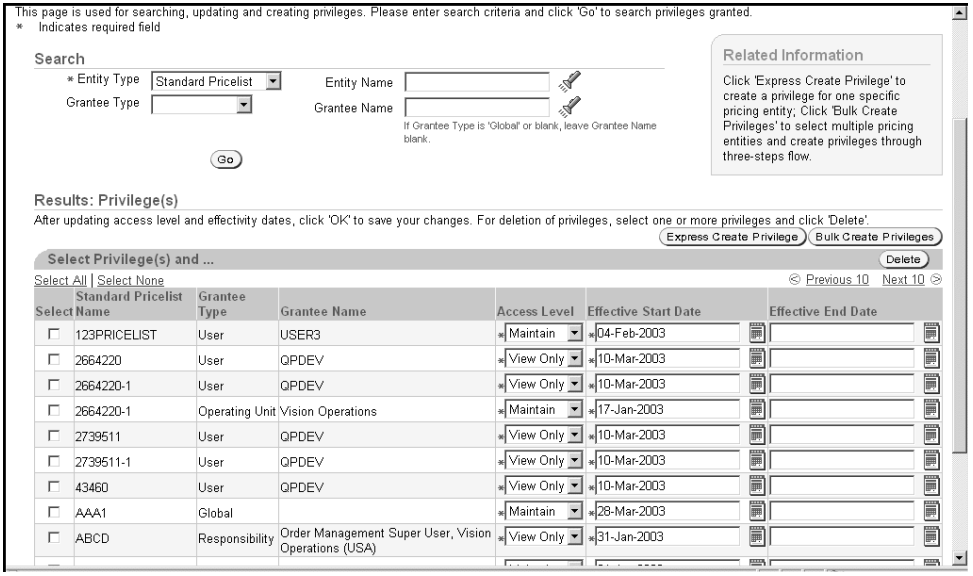
1. Navigate to the Privileges page.

Figure 4–4 Privileges page



2. In the Search region, select an Entity Type. Optionally, select additional search criteria such as Entity Name, Grantee Type, or Grantee Name to filter your search results. To view the available values for an Entity Name or Grantee Name, click the Search icon.
3. Click Go to display the search results in the Results region of the Privileges page.

Figure 4–5 Results: Privilege(s) page



If the message *No data exists displays* in the Results: Privilege(s) region then no privileges exist for the entity.

If search results display, then you can view or update the privileges directly in the Results: Privilege(s) region.

4. To revoke privileges, select the line to delete and click Delete.
5. To assign or update an Access Level, select Maintain or View Only.
6. Enter or update the Effective Start and End Date and click OK to save your changes.

To use Express Create Privilege:

1. To create a privilege for one specific pricing entity, select the entity and click the Express Create Privilege button to display the Express Create Privilege page.

Figure 4–6 Express Create Privilege page

2. In the Select Security Entity region, select the Entity Type and Entity Name of the pricing entity to be granted privileges.
3. In the Select Grantee region, select one of the following Grantee Types and a Grantee Name:
   - Responsibility: Grants the privilege to a specific responsibility such as Pricing User, Guest User (the specific Grantee Names depend on the setup for your specific business).
   - User: Grants the privilege to a specific user such as John Smith in the Pricing Department.
   - Global: If Grantee Type is Global, leave Grantee Name blank. This makes the privilege available to all users with functional access to pricing menus.
   - Operating Unit: Grants the privilege to a specific operating unit. For example, select Vision1 to give a privilege to all users belonging to operating unit Vision1.

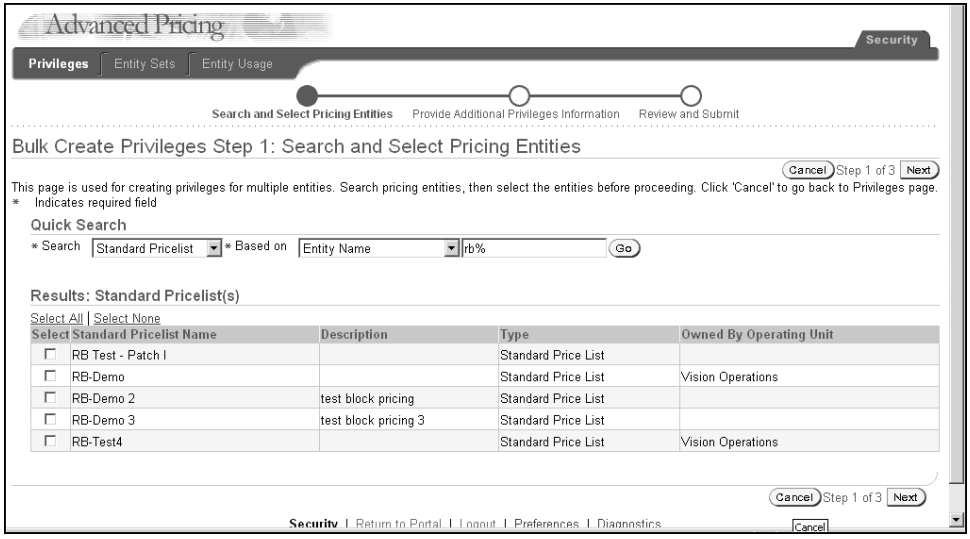
4. In the Select Access Level region, select the Access Level to be granted to the Grantee:
   - Maintain: Enables users to delete, view, and update pricing entities.
   - View Only: Enables users to view but not update the pricing entity.
5. In the Specify Duration region, select the Start and End Date. For example, to provide temporary access to a temporary employee, you could enter a Start Date of 02-Jul-2004 and an End Date of 31-Aug-2004. Alternately, accept the system dates.
6. Click OK.

Using Bulk Create Privileges

Using the Bulk Create Privileges page, you can quickly create and assign privileges to multiple entities for a specific entity type. For example, you could grant access to several price lists to the Operating Unit: Vision France.

1. From the Privileges page, click Bulk Create Privileges to display the Bulk Create Privileges Step 1: Search and Select Pricing Entities page.

Figure 4–7 Bulk Create Privileges page



2. In the Quick Search region, do a search by Entity Type to find the pricing entity or entities to be granted privileges. For example, select Standard Pricelist to search for standard price lists.
3. Optionally, select additional search criteria to refine your search. In the Based on field, select Owned by Operating Unit or Entity Name then enter related details.

   For example, to find the Summer Pricelist, select Standard Pricelist as the Entity Type, then select Entity Name and enter Summer Pricelist to specify your search criteria.
4. Click Go to display the search results in the Results region.
5. From the search results, select the entities to be assigned privileges.

6. Click Next to display the Bulk Create Privileges Step 2: Provide Additional Privileges Information page.

Figure 4–8 Bulk Create Privileges Step 2: Provide Additional Privileges Information page

7. Select one of the following Grantee Types and select an associated Grantee Name. To display the available values for a Grantee Name, click the Search icon:

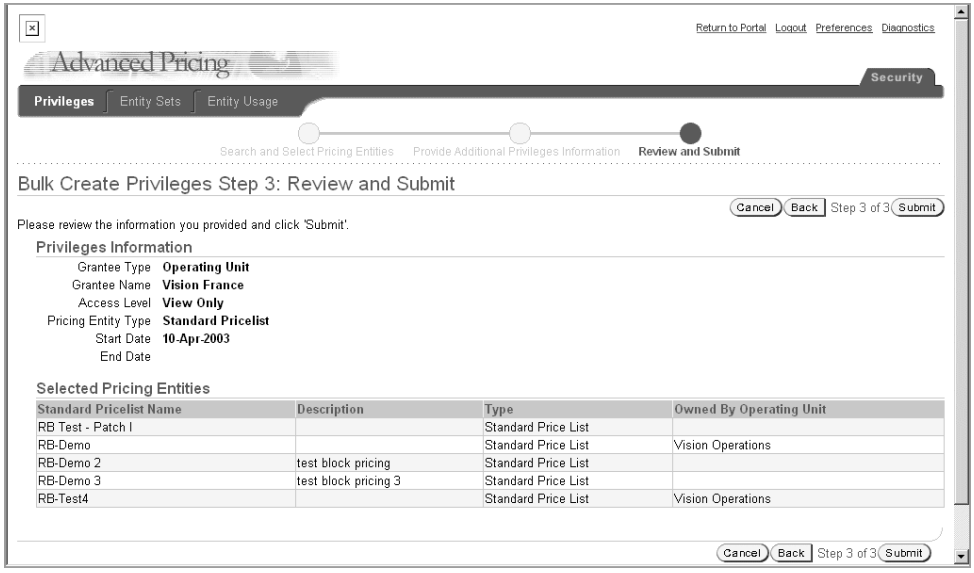
   - Responsibility: Grants the privilege to a specific responsibility such as Pricing User, Guest User (the specific Grantee Names depend on the setup for your specific business).
   - User: Grants the privilege to a specific user such as John Smith in the Pricing Department.
   - Global: If Grantee Type is Global, leave Grantee Name blank. This makes the privilege available to all users across operating units.
   - Operating Unit: Grants the privilege to a specific operating unit. For example, select Vision1 to assign the pricing entity to operating unit Vision1. Users not from operating unit Vision1 are unable to access this pricing entity.

8. Select the Access Level to be granted to the Grantee:

- View Only: Enables users to view but not update the pricing entity.
- Maintain: Enables users to delete, view, and update the pricing entity.

9. Select the Start and End Date in the Specify Duration region. For example, to grant temporary access to a summer employee, you could enter a Start Date of 02-Jul-2004 and an End Date of 31-Aug-2004. Alternately, accept the default system dates.
10. Click Next to display the Bulk Create Privileges Step 3: Review and Submit page.

Figure 4–9 Bulk Create Privileges Step 3: Review and Submit page

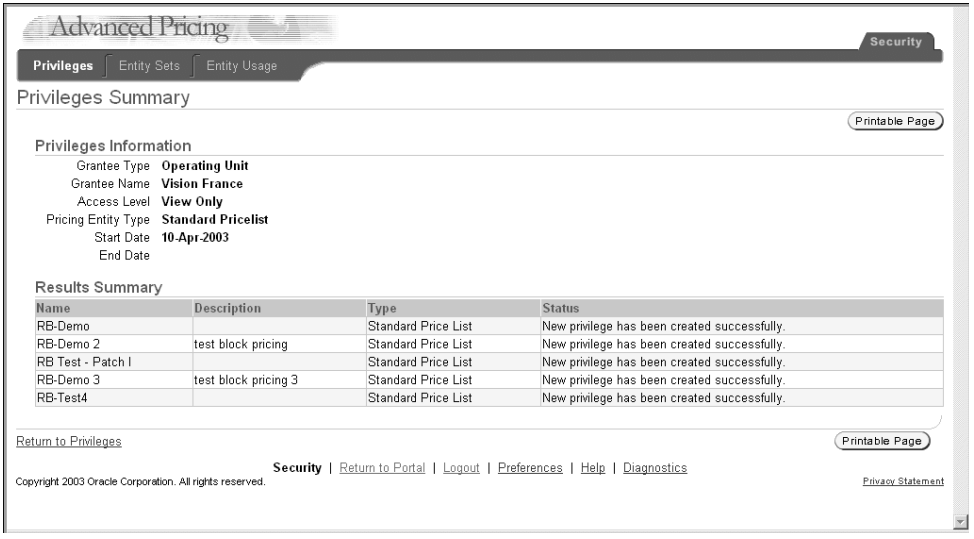


11. Review the information in the following regions before submitting your changes to ensure it is correct:
    - Privileges Information region: Displays the privilege information.
    - Selected Pricing Entities region: Displays the following information about the pricing entities to be granted the privileges listed in the Privileges Information region: Entity Name, Description, Type, Owned By Operating Unit.

12. If changes are required, click Back, or click Cancel to stop the process completely.

13. If the information is correct, click Submit. The Privileges Summary page displays the Privileges Information and Results Summary.

Figure 4–10 Privileges Summary page



Creating Pricing Entity Sets

You can create "sets" of pricing entities that can be used as a single entity when creating a privilege(s) for a given grantee. Each entity set can contain multiple pricing entities of the same entity type such as a set for price lists and another for modifiers.

For example, you could create an entity set consisting of all price lists based on a specific customer name, then grant maintain access (in the Privileges pages) to a specific user for the entity set.

The following outlines the steps for creating and using an Entity Set:

a. Create an Entity Set using the Create Entity Set page. In this page, you can select the criteria for your set. Only header level criteria can be used in creating the set.

b. Use the Entity Set as the grant object (with object type as ENTITY SET) and grant access roles to any grantee type and grantee.

It is important to identify the selected criteria in the description of the entity set. Once an entity set is defined, you cannot copy the entity set or change its criteria. If changes are required, a new entity set must be created. The Entity Set feature is only available to licensed users of Oracle Advanced Pricing.

Note: Entity sets cannot be copied or updated. You can revoke or add privileges as needed. However, the entity set cannot be deleted if there are any existing privileges on that entity set.

Example of Entity Set Usage

You create a new entity set named SET1 for all active modifiers for USD currency containing Wireless in the customer name. Next you query on the set name SET1 in the Entity Sets page. After clicking the Go button, no records are displayed in the Results region. This occurs because there are no records that currently exist meeting these criteria.

Next, you create a privilege for entity set SET1 and assign view only access for the Vision Operating Unit. Next, a user creates a new modifier: MOD 1 in the USD currency for the customer Totally Wireless and makes the modifier active.

The MOD 1 modifier will automatically be assigned to the SET1 entity set and inherits view only access.

For entity sets:

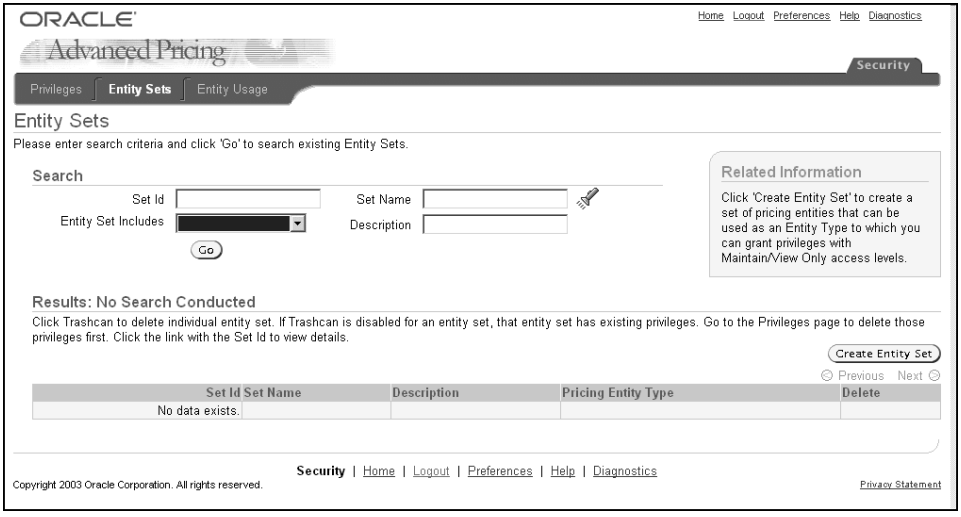
- It is possible to create an Entity Set for a specified set of criteria that does not currently exist in the system.
- It is possible to create access privileges for this entity set even when no records currently exist in the system.
- Any new records created that meet the set criteria are automatically assigned to the set and inherit the privileges assigned to the set.

If this entity set is used in an access privilege, the newly created entity will be included in the set and will have those privileges.

To create an entity set:

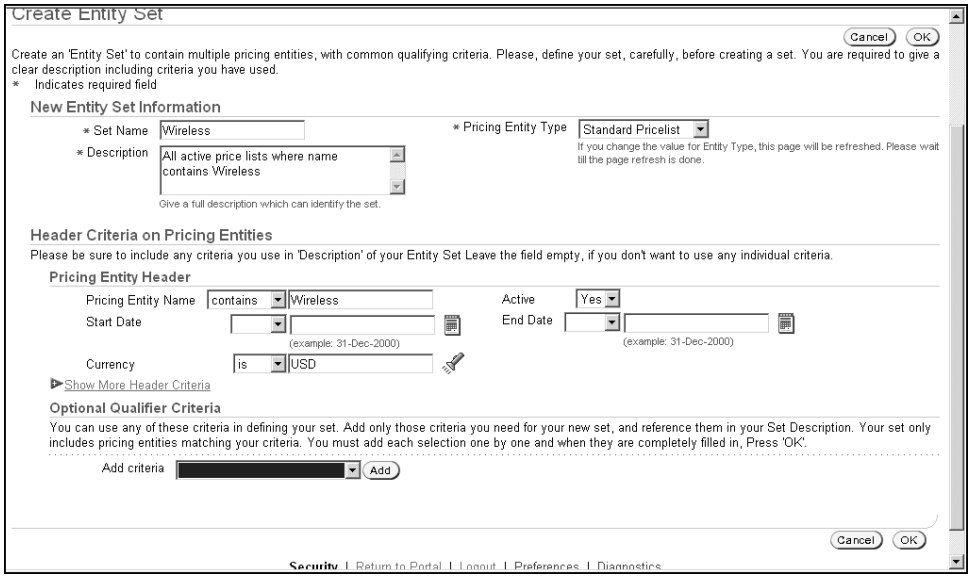
1. Navigate to the Entity Sets page.

Figure 4–11 Entity Sets page



2. Click Create Entity Set to display the Create Entity Set page.

Figure 4–12 Create Entity Set page



3. In the New Entity Set Information region, enter a Set Name that uniquely identifies the entity set you are creating.

4. Enter a Description that is simple, meaningful, and includes all the criteria selected for this entity set.

> **Note:** An Entity Set can only contain one unique pricing entity type. For example, Entity Set1 cannot contain both entity type Standard Price List and Modifier. To delete an entity set, you must first revoke all privileges on this set and then delete it.

5. Select one of the following Pricing Entity Types to be included in the entity set: Agreement Pricelist, Modifier, Standard Pricelist. Only one Pricing Entity Type can be included in an entity set.

6. In the Pricing Entity Header region, select the criteria for the pricing entities to be included in the set. The criteria to define the set should be included in the Description for the set.

7. For Pricing Entity Name, select an operator - is, is not, contains, starts with, ends with - then enter specific details about the Pricing Entity Name to be included in the set.

   For example, if you select Pricing Entity Name *is* Summer Price List, then the price list named Summer Price List will be included in the entity set. (Assuming Standard Price List was selected as the Pricing Entity Type.)

8. Optionally, select a Start and End Date and Currency as additional criteria.

9. Additionally, in the Optional Qualifier Criteria region, select criteria from Add Criteria field to add additional criteria and click the Add button.

   Add only the criteria needed for your new entity set - remember to add the additional criteria to the Set Description. Your entity set will include only those pricing entities exactly matching your criteria.

10. When you have completed your entries, click OK.

    You can use an Entity Set to restrict maintain access to a few individual users. You can then use the same set to give view-only access privilege to all other users. The set inherits the access level of the privilege for all users who are given access to the set.

    You can experiment with this function and find many ways to make use of it.

To view existing entity sets:

1. Navigate to the Entity Sets page to find an existing entity set.

2. In the Search region, enter the Set Id of the entity set and select the Set Name. Click the Search icon to view the available Set Names. Optionally, to reduce the search results, select or enter additional search criteria:

   - In the Entity Set Includes field, select Standard Price List, Agreement Price List, or Modifier.
   - Enter a Description or partial description of an entity set.

3. Click Go to display the search results in the Results: Entity Sets region. The following details display for each entity set retrieved:

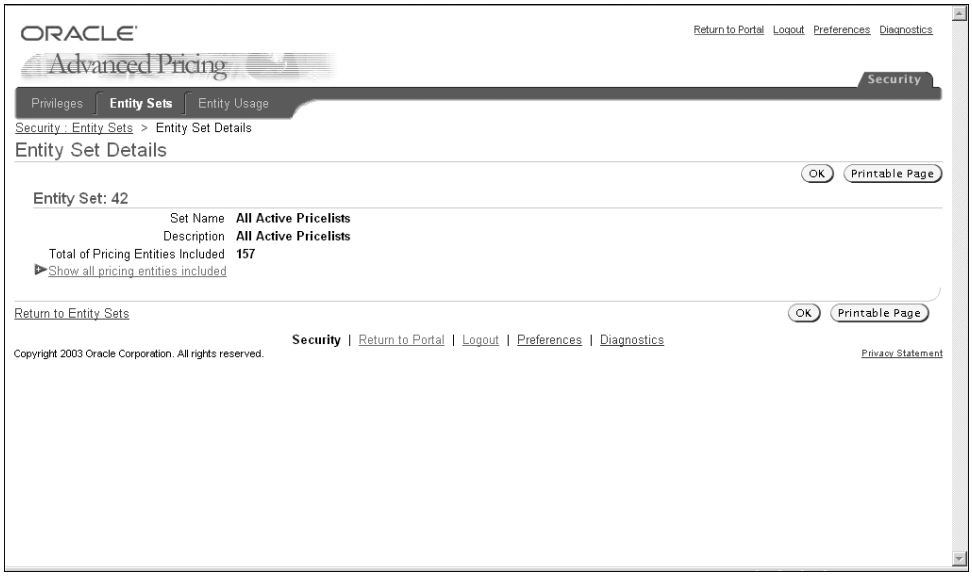
   - Set Id: A system assigned value that uniquely identifies the entity set.
   - Set Name: A name for the entity set entered by the user when the entity set is created.
   - Description: Describes the entity set entered by the user when the entity set is created.

- Pricing Entity Type: Sets contain all the same entity types meeting the selected criteria.

4. To view details about a specific entity set, click the Set Id of an entity set.

   The Set Entity Details page displays additional details about the selected entity set including the Set Name, the Description, and the Total of Pricing Entities Included.

Figure 4–13 Entity Set Details page



5. Click the Show all pricing entities included link to view the entities included in the entity set. The listing displays a table with the following information about each entity: Name, Description, Type (Standard Pricelist, Modifier, or Standard Agreement), and Owned By Operating Unit.

6. You can either click OK or click Printable Page to display the information in a convenient format that can be printed.

To delete an entity set:

1. Navigate to the Entity Sets page and do a search for an existing entity set.
2. In the Results: Entity Set(s) region, click the Delete icon to delete a specific entity set.

   If the Delete icon is grayed out, the entity set still has privileges assigned to it. Before the entity set can be deleted, you must first revoke the privileges, and then delete the entity set.

Setting up Default Security Profile Options for New Pricing Entities

Security profile options are used to define the default security privileges for newly-created price lists and modifiers. These profiles should be left in default setting (maintaining current functionality) and not be changed until you have decided which users should have automatic privileges of View Only and of Maintain whenever a pricing entity is newly created.

These privileges are automatically created as soon as the creating user saves the new entity. The following discussion will assist you in choosing the combination of settings to meet your security policy.

The following profile options are used to assign the default view-only or maintain access privileges to newly created price lists or modifiers:

- QP: Security Default ViewOnly Privilege: Determines the default *view-only* privileges for NEWLY CREATED price lists and modifiers. View and maintain responsibilities are controlled separately by different profile options. This profile option enables you to set the view-only privileges at one of the following levels: Global (Default), Operating Unit, Responsibility, User, or None. This controls which users (if any) can view newly-created price lists and modifiers.
- QP: Security Default Maintain Privilege: Controls the default *maintain* privileges for NEWLY CREATED price lists and modifiers. For example, if the profile option is set to Operating Unit, then the maintain privileges for that price list or modifier are restricted to the pricing users of the operating unit where the price list or modifier was created. This profile option enables you to set maintain privileges at one of the following levels: Global (Default), Operating Unit, Responsibility, User, or None.

Before setting the security profile options and changing the defaulting privilege profiles, complete all security setup requirements.

Note: To change the access privileges for *pre-existing* price lists and modifiers, use the Security Privileges window.

Warning: The profile option QP: Security Control turns pricing on and off. If you are upgrading or freshly installing the security feature for the first time, ensure you have completed the following steps before you set the default security profile options and turn on security for your installation:

- You have assessed and mapped out the behavior your business requires when a new price list or modifier is created. See "Assigning Ownership of Pricing Entities to Operating Units (Entity Usage page)" on page 4-10 for more information.
- Assigned an operating unit owner for existing pricing entities.
- Granted privileges at all levels based on your security policy and needs.
- Before turning the profile option QP: Security Control to ON, ensure that you have completed all the set up and implementation steps first--**otherwise, users will be unable to query any price lists or modifiers in the pricing windows**.

Security profile options and existing pricing entities

The two security profile options - QP: Security Default Maintain Privilege and QP: Security Default ViewOnly Privilege - have no effect on the behavior of existing pricing entities. Access to existing pricing entities depends on the privileges already granted by the Oracle Pricing Administrator using the Security Privileges and related pages.

Resolving conflicts between multiple access levels

If the user has two different access privileges to the same pricing entity, the access level of Maintain always prevails. For example, if a pricing user has Maintain access at the User level to certain price lists, and view-only access at the Responsibility level, the user will have Maintain privileges to those price lists.

In all cases, the highest access level--the Maintain access privilege--prevails over the View-Only privilege. This rule applies regardless of what operating unit id the user is in.

Security Profile Option Settings Compared

The following section lists possible combinations of security profile option settings that define the default view and maintain access privileges for newly created pricing entities. Review the combinations of profile option settings and select the combination that suits the requirements for your installation. When security is turned on, a price list and modifier that is newly created will be assigned the default view and maintain security privileges from the profile option settings.

Security Profile ON: Behavior when creating a new Pricing Entity

The following shows behavior by combinations of profile settings when setting up new price lists and modifiers. Available values are: None, User, Responsibility, Operating Unit, and Global.

Table 4–3 Security Profile ON: Behavior when creating a new Pricing Entity

QP: Default View Only Privilege	QP: Default Maintain Privilege	Behavior while being created	After saving and exiting the Entity's (Price list or Modifier) setup windows
None	None	Entity can be viewed/updated while being created.	1. The new entity cannot be viewed or updated by anyone.
None	User	Entity can be viewed/updated while being created.	2. The new entity can be viewed and updated by the user who created it only.
None	Responsibility	Entity can be viewed/updated while being created.	3. The new entity can be viewed and updated by users with the same responsibility as the user who created it only.
None	Operating Unit	Entity can be viewed/updated while being created.	4. The new entity can be viewed and updated by all users within the same OU as the user who created the entity only.
None	Global	Entity can be viewed/updated while being created.	5. The new entity can be viewed and updated by all users.

Security Profile ON: Behavior when creating a new Pricing Entity for Combination: Values for User

The following show behavior by combinations of profile settings when setting up new price lists and modifiers. Available values are: None, User, Responsibility, Operating Unit, and Global.

Table 4–4 Security Profile ON: Behavior when creating a new Pricing Entity for Combination (QP: Default View Only privilege is User)

QP: Default View Only privilege	QP: Default Maintain Privilege	Behavior while being created	After saving and exiting the Entity's (Price list or Modifier) setup windows
User	None	Entity can be viewed/updated while being created.	The user who created it can view the new entity. Nobody can update it.
User	User	Entity can be viewed and maintained by user who created it.	The new entity can be viewed and updated by the user who created it only.
User	Responsibility	Entity can be viewed and maintained by user who created it.	Similar to the None/Responsibility settings. Except that, the user can still view the entity even if he/she is exempted from the responsibility.
User	OU	Entity can be viewed and maintained by user who created it.	Similar to None/Operating Unit settings. Except that, the user can still view the entity even if he/she is exempted from the Operating Unit.
User	Global	Entity can be viewed and maintained by user who created it.	Same as None/Global settings. The new entity can be viewed and updated by all users.

Security Profile ON: Behavior when creating a new Pricing Entity for Combination: Values for Responsibility

The following show behavior by combinations of profile settings when setting up new price lists and modifiers. Available values are: None, User, Responsibility, Operating Unit, and Global.

Table 4–5 Security Profile ON: Behavior when creating a new Pricing Entity for Combination: Values for Responsibility

QP: Default View Only Privilege	QP: Default Maintain Privilege	Behavior while being created	After saving and exiting the Entity's (Price list or Modifier) setup windows
Responsibility	None	Entity can be viewed and maintained by user who created it.	All the users can view the new entity with the same responsibility as the user who created it. Nobody can update it.
Responsibility	User	Entity can be viewed and maintained by user who created it.	All the users can view the new entity with the same responsibility as the user who created it. And, only the user who created it can update it.
Responsibility	Responsibility	Entity can be viewed and maintained by user who created it.	Same as None/Responsibility settings. The new entity can be viewed and updated by users with the same responsibility as the user who created it only.
Responsibility	Operating Unit	Entity can be viewed and maintained by user who created it.	All the users can view the new entity with the same responsibility as the user who created it. And, all the users within the same OU as the user who create it can also update it.
Responsibility	Global	Entity can be viewed and maintained by user who created it.	Same as None/Global. The new entity can be viewed and updated by all users.

Security Profile ON: Behavior when creating a new Pricing Entity for Combination: Values for Operating Unit

The following show behavior by combinations of profile settings when setting up new price lists and modifiers. Available values are: None, User, Responsibility, Operating Unit, and Global.

Table 4–6 Security Profile ON: Behavior when creating a new Pricing Entity for Combination: QP: Default View Only Privilege is Operating Unit

QP: Default View Only Privilege	QP: Default Maintain Privilege	Behavior while being created	After saving and exiting the Entity's (Price list or Modifier) setup windows
Operating Unit	None	Entity can be viewed and maintained by user who created it.	All the users within the same OU as the user who created it can view the new entity. Nobody can update it.
Operating Unit	User	Entity can be viewed and maintained by user who created it.	All the users within the same OU as the user who created it can view the new entity. And, only the user who created it can update it.
Operating Unit	Responsibility	Entity can be viewed and maintained by user who created it.	All the users within the same OU as the user who created it can view the new entity. And, all the users with the same responsibility as the user who created it can update it.
Operating Unit	Operating Unit	Entity can be viewed and maintained by user who created it.	Same as None/OU settings. The new entity can be viewed and updated by all users within the same OU as the user who created the entity only.
Operating Unit	Global	Entity can be viewed and maintained by user who created it.	Same as None/Global settings. The new entity can be viewed and updated by all users.

Security Profile ON: Behavior when creating a new Pricing Entity for Combination: Values for Global

The following show behavior by combinations of profile settings when setting up new price lists and modifiers. Available values are: None, User, Responsibility, Operating Unit, and Global.

Table 4–7 Security Profile ON: Behavior when creating a new Pricing Entity for Combination: QP: Default View Only Privilege is Global

QP: Default View Only Privilege	QP: Default Maintain Privilege	Behavior while being created	After saving and exiting the Entity's (Price list or Modifier) setup windows
Global	None	Entity can be viewed and maintained by user who created it.	All the users can view the new entity. But nobody can update it.
Global	User	Entity can be viewed and maintained by user who created it.	All the users can view the new entity. Only the user who created it can update it.
Global	Responsibility	Entity can be viewed and maintained by user who created it.	All the users can view the new entity. And, all the users with the same responsibility as the user who created it can update it.
Global	Operating Unit	Entity can be viewed and maintained by user who created it.	All the users can view the new entity. And, all the users within the same operating unit as the user who created it can update it.
Global	Global	Entity can be viewed and maintained by user who created it.	Same as None/Global. The new entity can be viewed and updated by all users

The Oracle Pricing Administrator can assign or change ownership of a pricing entity using the Entity Usage page.

The pricing administrator can make changes to Global Usage of price lists and modifiers one by one, or use the Bulk Update Entity Usage function in the Entity Usage page to make changes more quickly.

Warning: **It is important that the Oracle Pricing Administrator assigns ownership to all price lists and modifiers prior to upgrading or implementing pricing security. This can be done using the Bulk Update Entity Usage feature in the Entity Usage page. Otherwise, users will be unable to query any pricelists or modifiers.**

Setting the QP: Security Control profile option to ON

When the profile QP: Security Control is first set to ON, a Global check box is dynamically added to the header region of all price lists and modifiers. The Global box is visible to end-users and can be updated (cleared or selected) by users with Maintain access privileges.

The Global box indicates if global usage is enabled for the entity. When the Global box is selected, the entity is available across all operating units in your organization.

You can update the Global box for each price list and modifier window singly, or do bulk updates in the Bulk Update Entity Usage page. See "Assigning Ownership of Pricing Entities to Operating Units (Entity Usage page)" on page 4-10 for more information.

Prior to setting profile QP: Security Control to ON, pricing entities are not identified by an operating unit. It is very important that the Oracle Pricing Administrator assigns ownership to all price lists and modifiers prior to upgrading to or implementing pricing entity security. You can use the Bulk Update Entity Usage feature in the Entity Usage page to assign or reassign global usage values.

After turning pricing security on, all newly created pricing entities are assigned a unique default operating unit identification that makes the creating operating unit the owner of the pricing entity.

The following table shows the behavior of existing pricing entities when QP: Security Control is set to ON and no pre-security is assigned:

Table 4–8 Security Profile ON: Behavior of Existing Pricing Entities

QP: Default View Only Privilege	QP: Default Maintain Privilege	Privileges from Pricing Security Administrator	Behavior
Not applicable	Not applicable	No privileges granted	Entity cannot be viewed or updated by anybody except the Oracle Pricing Administrator through the security management pages selected from the Oracle HTML user interface.
Not applicable	Not applicable	Maintain	Entity can be viewed and updated by the user with Maintain access privileges.

5

Modifiers

This chapter discusses considerations for the implementation of modifiers. The following topics are covered:

Introduction

Modifiers are pricing actions that drive your pricing processes in addition to price lists, formulas, and agreements. Modifiers can adjust net price either up or down. Modifier actions include discounting, adding surcharges, charging for shipping, or adjusting price based on promotional pricing actions.

Implementation Decisions

As you analyze and understand a client's business pricing scenario, there are some key implementation decisions to address in order to develop a logical pricing solution. These decisions will be discussed throughout the chapter.

Key Implementation Decision: How do I use modifiers in pricing actions? Which modifiers best demonstrate my pricing processes?

Oracle Advanced Pricing comes seeded with five types of modifier lists and nine modifier types. The modifier list types are: discount, surcharge, deal, promotion, and freight and special charges. For an explanation of modifier types, refer to Chapter 1 Introduction.

The following table displays modifier types you can create within a modifier list.

Table 5–1 Modifier types that can be created within a modifier list

Modifier Type	Discount List	Surcharge List	Deal (must be associated with a promotion)	Promotion	Freight and Special Charges List
Discount	Yes	Yes	Yes	Yes	No
Surcharge	Yes	Yes	Yes	Yes	No
Other Item Discount	No	No	Yes	Yes	No
Terms Substitution	No	No	Yes	Yes	No
Item Upgrade	No	No	Yes	Yes	No
Price Break	Yes	Yes	Yes	Yes	No

Table 5–1 Modifier types that can be created within a modifier list

Modifier Type	Discount List	Surcharge List	Deal (must be associated with a promotion)	Promotion	Freight and Special Charges List
Promotional Good	No	No	Yes	Yes	No
Coupon Issue	No	No	Yes	Yes	No
Freight and Special Charge	No	No	No	No	Yes

Modifier Levels and Application Methods

The pricing engine uses modifier level codes to determine pricing request eligibility for specific modifiers. These level codes also determine to which level, line, or summary a modifier should be applied. There are three types of modifier level code: order, line, and group of lines. Line level codes apply only to a specific order line, while group of lines level codes apply to groups of lines in the order. Order level codes apply to an entire order.

For example, a volume-based discount modifier with a line level code will consider only volume on the qualifying line. A volume-based discount modifier with a group of lines level code will process the volume across all qualifying lines in the product hierarchy.

Special Note on Group of Lines Behavior

Group of Lines Based Discounts

Discounts/PRG/OID/PBH with modifier level code group of lines can be set only for volume attributes item quantity and item amount as of now. They cannot be set based on any attribute under VOLUME context.

Discounts/PRG/OID/PBH based on group of lines, with no exclusions on items or item categories order lines that qualify for this group of lines modifier, are considered for group of lines quantity or amount calculation.

Discounts/PRG/OID/PBH based on group of lines, with exclusions on some items or item categories, order lines based on an excluded item or item category that qualify for this group of lines modifier, are not considered for group of lines quantity or amount calculation.

Examples:

Modifier_Shampoo1 (no exclusion):

Group of Lines/Item Category: Shampoo/Volume > 100 Qty

Modifier_Shampoo2 (exclusion):

Group of Lines/Item Category: Shampoo/Volume > 100 Qty/Excluding Item Shampoo1

In the following table, order lines 1 and 2 qualify for Modifier_Shampoo1. To determine whether the ordered quantity is greater than 100, the quantity on order lines 1 and 2 are summed, resulting in 110. The modifier is either given or not given, depending on the result of this check. In this situation the modifier is given because the ordered quantity in item category shampoo is greater than 100.

In the following table, order line 2 qualifies for Modifier_Shampoo2. To determine whether the ordered quantity is greater than 100, the quantity on order line 2 is summed, resulting in 40. Whether the modifier is given depends on this verification. In this case the modifier is not given because the ordered quantity in Item Category: Shampoo is not greater than 100.

Table 5–2 Example of shampoo order

Line ID	Item	Quantity	Category
1	Shampoo 1	70	Shampoo
2	Shampoo 2	40	Shampoo
3	Conditioner	30	Conditioner

Key Implementation Decision: How do I assign modifier application methods?

Four modifier application methods are available to enhance pricing actions. The methods are:

- Amount: Creates a fixed price adjustment on each unit for the amount specified in the value field.
- Percentage: Creates a percentage price adjustment on each unit for the percentage specified in the value field.
- New price: Overrides the selling price of the item and makes it the new price.
- Lumpsum: Creates a price adjustment for the entire sum amount of the line.

The application methods available depend on the modifier line type and level selected. This can be found on the discounts/charges tab. You can use a formula to drive the value of your pricing action.

The following tables depicts the application methods allowed based on various modifier levels.

Table 5–3 Application methods and modifier levels matrix (Line level modifiers)

Line level modifier	Percent Value	Percent Formula	Amount Value	Amount Formula	New Price Value	New Price Formula	Lumpsum Value	Lumpsum Formula
Discount	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Surcharge	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
F&SC	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Other Item Discount (Get)	Yes	No	Yes	No	Yes	No	Yes	No
Price Break	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Promotional Good (Get)	Yes	No	Yes	No	Yes	No	Yes	No

Table 5–4 Application methods and modifier levels matrix (Group of Lines level modifiers)

Group of lines level modifier	Percent Value	Percent Formula	Amount Value	Amount Formula	New Price Value	New Price Formula	Lumpsum Value	Lumpsum Formula
Discount	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	Yes
Surcharge	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	Yes
Other Item Discount (Get)	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	No
Price Break	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	Yes
Promotional Good (Get)	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	No

Table 5–5 Application methods and modifier levels matrix (Order level modifiers)

Order level modifier	Percent Value	Percent Formula	Amount Value	Amount Formula	New Price Value	New Price Formula	Lumpsum Value	Lumpsum Formula
Discount	Yes	Yes	No	No	Yes	Yes	No	No
Surcharge	Yes	Yes	No	No	Yes	Yes	No	No
F&SC	No	No	No	No	No	No	Yes	Yes

Other Modifier Considerations

Key Implementation Decision: How will my customer and product hierarchies affect the structure of my modifiers?

Using qualifiers, you can tie the structure of your modifiers to your customer hierarchy. For more information on this topic, see to Chapter 3, "Implementation Methodology".

You can set up modifiers for item number, item category, or all items. Through pricing attributes, you can further define your product hierarchy.

Excluding Item Categories

By selecting exclude, you can exclude individual items or item categories from being applied to a modifier. If an item is in an excluded item category and in an non-excluded item category, the excluded item category is honored.

For example, a modifier for all items excludes item category (IC) 1. You have an item Z that is in IC1, IC2, and IC3. When the engine assess that Item Z is in IC1, the modifier is not applied, although the modifier is in IC2 and IC3.

Unit of Measure (UOM)

There are no UOM conversions for modifiers. The pricing engine evaluates only modifier lines that have matching units-of-measure (UOMs) or null value that is selected for the pricing UOM. For example, item A has a UOM of each with primary UOM checked for the price list line. The ordered UOM for item A is dozen. The engine will therefore only consider modifier lines with each or with null values. UOM is not a mandatory field for modifier types other than price breaks.

Note: The UOM is not mandatory for promotional modifiers if the volume type is Item Amount. The UOM is mandatory ONLY in the following cases:

- List line type is Promotional Goods, Other Item Discount, or Item Upgrade and the volume type is not Item Amount, Period1 Item Amount, Period2 Item Amount, Period3 Item Amount.
- The Modifier line volume type is Item Quantity.

Pricing phase

The pricing phase determines when the pricing engine applies a modifier. For example, modifiers in the list line adjustment phase are applied only after leaving the line. Modifiers in the all lines adjustment phase are applied only after saving the line. Consider unnecessary pricing engine calls that can affect performance. See "What are Pricing Phases?" on page 12-4 for more information on pricing phases.

Precedence

The values for precedence will default based on the setup in the qualifier context. For more information on precedence, see Chapter 3, "Implementation Methodology"and Chapter 9, "Precedence and Best Price".

Incompatibility level

You can make modifiers incompatible with each other by placing them in the same incompatibility group level. When this is done, the pricing engine applies only one modifier per phase to the order line or order. For more information regarding incompatibility levels, refer to Chapter 9, "Precedence and Best Price".

Buckets

Pricing buckets (pricing group sequences) control how your calculation of discounts and other benefits is sequenced. Grouping determines net selling price. Buckets are applied across phases.

Key Implementation Decision: How many bucket levels do I need?

Determine the number of buckets you need by counting subtotals on your client's invoice. Control the modifier calculation sequence by adding the numbers between beginning list price and final net price. Plan the assignment of discounts to buckets based on these subtotals.

The null bucket calculates the modifier off of the list price and then applies the adjustment to the last bucket's subtotal. Manual and order level modifiers are always in the null bucket.

Oracle Advanced Pricing does not restrict the number of buckets you can define.

Pricing Controls

You can use the following fields on the define modifiers window to set additional pricing controls.

- Effectivity dates at the modifier list and modifier line: Effectivity dates on the modifier line must fall in between the effectivity dates for the modifier list.
- Additional date types for order date and requested ship date: These are additional parameters that can be set to control the effectivity of the modifier list.
- The Comparison Value field: This field holds the approximate value for the benefit item(s) for item upgrade, term substitution, coupon issue, other item discount, and promotional good. This value is used during best price processing (the field does not impact Oracle General Ledger).
- Asked for, automatic, and override check box: There are two automatic check boxes on the define modifiers window; one at the modifier list level and one at the modifier line level. The latter must be checked for automatic modifiers. The list level automatic check box is used for defaulting at the line level.

  Select the override check box to override the value of the price adjustment. adjustment in the calling applications. Modifiers with Asked For flag checked are given only if requested by the user through calling applications.

Note: The Calculate Price flag is passed by the calling application. It enables the calling application to fully or partially freeze a price request. Depending on the value of the flag, price may be completely frozen, or additional modifiers may be applied in certain phases.

Modifier Type Setup

Manual Modifiers

You can define manual adjustments for discounts, surcharges, freight charges, and point price breaks. In order to override the selling price directly on the order line, you must define a manual discount or a manual surcharge. Remember the following when defining manual adjustments:

- The automatic box must be unchecked.
- The override box must be selected for overrideable manual adjustments.
- All manual adjustments are in the null bucket.
- Manual adjustments are applied based on pricing phase.
- The engine returns manual discounts based on the value defined in the profile option QP: Return Manual Discounts. If this is set to Y, then all manual discounts are returned and all automatic discounts not considered are returned as manual discounts. This is the default. If the profile option is set to N, then all manual and automatic discounts undergo incompatibility processing and one per incompatibility group is returned. Discounts (automatic or manual) deleted as part of incompatibility processing are returned as manual discounts.
- For manual discounts that use formula calculation, all attributes for the formula must be sent to the pricing engine in the pricing request. The engine returns the manual discount, which can be applied to the order line.
- In the manual adjustments LOV for the unit selling price on the sales order line ONLY line and line group manual adjustments are displayed.
- After applying manual adjustments to the line the calculate_price_flag remains Y. If you do not want other adjustments applied to the line, set the calculate_price_flag to P or N. If you want to apply order level manual adjustments, do so through the View Adjustments screen. If the calculate_price_flag on any of the lines is P or N, order level adjustments cannot be applied to the order. If there are no applicable manual adjustments for a line, a note displays that indicates there are no applicable discounts available.

To apply a manual adjustment in Oracle Order Management, see the *Oracle Order Management Suite Implementation Manual,* or the *Oracle Order Management Suite User's Guide.*

Other Item Discount

In order for an other item discount to apply, all items must correctly sequenced on the order. The pricing phase must be tied to an event that considers all lines in the order. Other item discounts may not be manual or overrideable. Other item discounts ignore incompatibility for the get line.

Get Region and Benefit Items

Benefit items in the Get region can only be defined at the item level. Users should not enter values for the get price and get UOM fields; the price of the benefit item should be on the order.

Because the pricing engine does not support recurring other item discounts, get quantity should be 1. The engine will apply the discount to all quantities of the item. For example, for every 5 pastries and 2 cookies you order, you get 2 cookies at 50% off. Using this modifier, if you order 10 pastries and 4 cookies, you receive 4 cookies at 50% off. Similarly, if you order 10 pastries and 3 cookies, all 3 cookies are 50% off.

If you discount using the percent application method, percent is based on the bucket value defined in the modifier summary tab. For example, if the other item discount is in bucket 2, then a 5% discount is given after all adjustments for bucket 1 are applied to the get item.

Term Substitution

Payment, freight, and shipping terms can be substituted using the terms substitution modifier.

- The modifier level is always line or order level.
- Define product attribute by item number or item category for line level term substitution.

Item Upgrade

For item upgrade modifiers, the relationships between items and upgraded items must be defined in Oracle Inventory.

- Promotional upgrade is the relationship type.
- The UOM for the upgraded item will be the same as the UOM of the original item.
- The pricing engine only recognizes the original item for additional modifiers; it never recognizes the upgraded item.

Price Breaks

Price breaks can be defined for any pricing attribute in the pricing context volume. The modifier level line or group of lines determines how the pricing engine handles these price breaks.

- For manual price breaks only point price breaks are allowed.
- When you split a line the selling price changes if volume falls into a different price bracket. At that point, the outcome will depend on the value of the calculate price flag prior to the split. To prevent the price from automatically changing, set the calculate price flag field to freeze price prior to splitting the line.

Point Price Break

Consider the following example discount rules:

- Item quantity ordered 1-10: 1% discount
- Item quantity ordered 11-50: 2% discount
- Item quantity ordered 51-999: 5% discount

If this is a line level price break and the ordered quantity is 55, then a 5% discount is applied to the order line. If this is a group of lines modifier, the total quantity over all the group of lines on the order receive the discount.

Range Price Break

Using the point price break example, if the ordered quantity is 55, then the first 10 items ordered receive a 1% discount, the next 40 receive a 2% discount, and the remaining 5 receive a 5% discount. The discount is then averaged and applied to the order line.

Promotional Goods

- Benefit items in get region can only be defined at the item level.
- The pricing engine returns additional order line for the promotional good and the calculate price flag is automatically set to N. Additional modifiers are not applied to the line unless the flag value is changed to Y or P. This prevents negative selling prices on free good items. If there are order lines where the Calculate Price flag is set to N, then additional order level modifiers are not be applied.

An example of a promotional good is:

Buy 1 PC and get a free mouse, or buy 1 PC and get free speakers.

There are two ways to set up this promotion:

1. Set up two modifiers: Modifier A is buy 1 PC and get 1 free mouse, Modifier B is buy 1 PC and get free speakers. Make the two modifiers incompatible with each other and set the precedences so that the modifier with the higher precedence will be automatically applied.
2. Set up two other item discount modifiers. Modifier A is buy 1PC and Buy 1 mouse, then get the mouse free. Modifier B is buy 1 PC and buy speakers, then get the speakers free. Make these two modifiers incompatible with each other and then set precedence. When the PC is ordered and the mouse or speakers is added to the order one of these items is free.

Note: In the Get region of the Define Modifier Details window, only price list lines that are stand-alone price list lines are displayed in the list of values. All service items and all price break lines (headers and children) are not displayed and cannot be selected from the list of values.

Coupon Issue

Two modifier lines must be set up when defining a coupon issue in the modifiers window. The price adjustment or benefit is linked to the coupon issue. When setting up the coupon issue line, a modifier number is mandatory. The pricing engine uses the number to generate a unique number series for the coupon which it passes to the calling application. The customer quotes the number when redeeming the coupon in the calling application.

Asked For Promotions

On the modifiers setup window, there is a check box on the advanced tab for asked for promotions. A customer must "ask for" the promotion name or number. The pricing engine validates order eligibility for the asked for promotion after the order is sent to the pricing engine. If the order is not eligible for the promotion, the engine will return an error message. This feature is only available for modifier list types deals and promotions.

You can also set up both automatic and manual asked for promotions. The methods in which the calling application applies asked for promotions depends on the calling application.

Recurring Modifiers

You can mark modifiers as recurring in the break type field on the modifier summary tab. The following modifier types enable recurring:

- Discount
- Surcharge
- Price break
- Promotional good (additional buy items do not recur)
- Coupon issue

An example of a recurring coupon issue is: for every 5 items ordered, get a coupon for 10% off a future order. If you order 15 items, then you receive 3 coupons.

Accruals

Both monetary and non-monetary accruals can be created through the modifier setup window. Accruals are given as a percentage, amount, or lumpsum and do not affect order line selling price. Accruals can have expiration dates attached to them and do not appear as chargeable items on invoices. Accrual accumulation is stored in the OE_Price_Adjustments table. Reference to accruals may be made by checking the Accrual flag.

Remember the following when setting up accrual discounts:

- Select the accrual check box in the discounts/charges tab.
- Benefit quantity and benefit UOM are of the benefit you wish to accrue.
- Expiration date specifies when the accrued transactions expire (optional).
- For the fields expiration period and expiration type, expiration period begins when the item begins to accrue. The pricing engine will calculate the expiration date.

> **Note:** The expiration period and expiration type fields cannot be entered if expiration date has been entered.

- Accrual conversion rate is used to specify the conversion of the benefit UOM to the primary currency. An example of a conversion rate is: if one air mile is 0.50 currency units, the accrual conversion rate is 0.50. The UI window exists for marking accruals as redeemed.

Fields reserved for future use:

- Rebate transaction
- Percent estimated accrual rate

Buckets with Monetary Accruals

Because accruals do not affect order line selling price the pricing engine does not include accruals in bucket calculations. The engine uses bucket numbers to determine the price with which to calculate accrual value. The following table illustrates this concept.

Table 5–6 Buckets with Monetary Accruals

Bucket	Modifier	Price Adjustment	Accrual Amount	Bucket Subtotal	Selling Price
List Price	n/a	n/a	n/a	n/a	$100.00
1	7% discount	$7.00	n/a	$7.00	$93.00
1	10% accrual	n/a	$10.00	n/a	$93.00
2	10% accrual	n/a	$9.30	n/a	$93.00
2	$5 discount	$5.00	n/a	$5.00	88.00

Accounting Limitations

- Accrual discounts are accounted for in the same manner as regular discounts in Oracle Advanced Pricing, Oracle Order Management, and Oracle Accounts Receivable. Oracle General Ledger account information is currently not used in Oracle Advanced Pricing modifier and accrual redemption forms.
- Oracle General Ledger account information from Oracle Order Management may not be passed to Oracle Accounts Receivable without a work-around using standard memo lines for auto invoicing. Discounts are expensed as a sales expense and accruals are deferred expense. Each implementation will require establishment of Oracle General Ledger accounts and mapping information flow of accounting for discount expenses.
- There is no interface for Oracle Accounts Receivable and Oracle Accounts Payable to research available accruals, balance, and decrementing for payments.
- Currently only net revenue is posted to the Oracle General Ledger. This posting occurs between Oracle Order Management and Oracle Accounts Receivable products.

Accrual Redemption

Accruals may be reviewed and marked as redeemed in the accrual redemption screen. This screen is an online view that reflects all accrual records from transactions. It can be used to view all redeemed and unredeemed records, or you can view using a more specific query, such as modifier number, customer, or redeemed only.

Querying any customer name or customer number returns all redeemed and unredeemed records for the customer regardless of modifier number. Querying by modifier returns specific modifiers with records for all customers who qualify for it.

Transaction type and source system are populated by any automated redemption processes. This reflects both manual redemption and those created by other transaction sources such as Oracle Accounts Receivable, Oracle Accounts Payable, or other source systems. You can manually update accrual redemption by keying in:

- Transaction reference: credit memo, check number, or write off codes
- Payment system: Oracle Accounts Receivable, Account Payable, or other system
- Redeemed date (defaults as current date, can be changed by user)

Copy Modifiers

The Copy Modifiers window and concurrent program ease modifier setup. With this window you can:

- Copy an existing modifier to a new modifier. The modifier name is specified by the user.
- Choose a different modifier description.
- Retain the modifier list effective dates or set new effective dates.
- Retain effective dates of the old modifier.
- Copy all Oracle Advanced Pricing attributes without change.
- Copy price breaks, other item discounts, coupons, promotional goods setups, and lines associated with every modifier.

> **Note:** A copied modifier is not active; the active flag for the new modifier is not checked.

6

Multi-Currency Price Lists and Agreements

This chapter contains implementation information about Multi-Currency price lists and Agreement Price List functions in Oracle Advanced Pricing. For further information, also see: *Oracle Advanced Pricing User's Guide*, Multi-Currency Conversion Lists.

For detailed information about set up, maintenance and other considerations of non-multiple currency price lists and agreement price lists, see: *Oracle Advanced Pricing User Guide*, Price Lists and Agreements.

The following topics are covered:

Implementation Decisions Single Currency versus Multiple Currency Price Lists

Advanced Pricing supports both single currency price lists and multiple currency price lists. For single currency price lists, there is one currency defined per price list. For multi-currency price lists, there is one base currency price list and an attached Currency Conversion list defining conversion factors and rules for converting prices.

It is very important to determine which price list strategy your organization wants to support. Advanced Pricing provides a profile option and concurrent program to convert existing price lists from a Single Currency Price List to Multiple Currency price lists. However, once this profile is enabled, and price lists are converted, users should not return to NON Multi-Currency price lists. Changing the profile back to No may cause undesired results if conversion criteria were used. Oracle does not support changing the setting back to No.

Single Currency Price Lists

Single currency price lists are the default setting for Advanced Pricing. These are used when your business requires that you maintain different prices for different currencies, and there is no relationship between prices defined.

Multiple Currency Price Lists

Multiple Currency price lists enable businesses that have pricing strategies based on a single price for an item in a base currency and use exchange rates or formulas to convert that price into the ordering currency. At engine run time, the pricing engine will take the currency from the order and search for a price list(s) with base or conversion currencies matching this currency. The pricing engine converts the price from the base currency and calculates the ordering currency based upon the established conversion rules.

Overview of Multi-Currency Price Lists

Multi-currency price lists use a specified base currency and the conversions for other currencies are applied to the values of the base currency price list. Additionally, multi-currency price lists allow some significant features such as Markup conversions that can be applied to the values of the base price list without any changes to the list line values. This can significantly reduce the number of price lists that must be maintained and reduce data storage.

The Currency Conversion window is used to define the conversion criteria. Seeded conversion types include: fixed, formula, user defined, spot, EMU fixed, transaction and corporate.

> **Note:** Some of these seeded conversion types require Oracle General Ledger to be installed. Users can still link to other non-Oracle stored conversion rate information by using the formula functionality

Attributes

It is possible to define multiple conversion criteria based upon certain attributes for the same base currency. For further information on attributes, see: Attribute Management chapters in the *Advanced Pricing Implementation Guide* and *Oracle Advanced Pricing User's Guide*.

Markup Values and Formulas

Users can define markup criteria per currency definition, including the base currency. This markup can be a fixed value of amount or percent, or based upon a formula. For further information on formulas, see: *Oracle Advance Pricing User's Guide*, Formulas.

Rounding

Base Round To: The Base Round To rounds the selling price after applying the modifiers.

In this field, users can enter a numeric-rounding factor that rounds the list price after the conversion and markup is applied

You cannot enter the Base Round To on the price list. Base Round To must be set in the currency conversion list as a Base Round To and will always default to the price list from the attached currency conversion list.

Conversion Round To: The Conversion Rounding Factor rounds the To Currency list price after the conversion and markup is applied.

This rounding factor could be different for the same To-Currency. This is entered in the Conversion Rounding Factor field in the currency conversion list.

Round To: Round To rounds the selling price after applying the modifiers. This rounding factor will always be the same for a Currency To.

Fresh Install using Multiple Currency Price List

Profile: QP Multi-Currency Installed

The Advanced Pricing profile QP: Multi-Currency Installed controls which type of price lists you will use. The System Administrator sets the profile. The default value for this profile is No which enables you to create and maintain many price lists defined in different base currency (non-multi-currency enabled).

You can enable multiple currency price lists by setting the value of the profile to Yes, which allows you to set up base currency price list(s) with a multiple currency conversion list attached to each price list or agreement price list.

Setting the value of the profile to Yes enables all Price List and Agreement windows with some window field and functionality changes. Once the profile is set, you must create at least one multiple currency conversion list. A currency conversion list is required when setting up a price list.

For detailed setup of Price Lists and Currency Conversion lists, see: *Advanced Pricing User's Guide*, Multi-Currency Conversion Lists. The steps to enabling multiple currency price lists are:

- Navigate to System Administer responsibility >Profile > System> QP: Multi Currency Installed > Select 'Yes'.
- Navigate to Oracle Pricing Manager responsibility >Price Lists > Multi-Currency Conversion setup > Create a base currency multi-currency conversion list
- Navigate to Oracle Pricing Manager responsibility >Price List Setup > Create a base currency master price list

> **Note:** If single currency price lists were created prior to changing QP: Multi-Currency Installed to Yes, then it is necessary to run the concurrent program *Update Price Lists with Multi-Currency Conversion Criteria*. This will convert all existing price list and agreement forms as Multi-Currency and activate the Multi-Currency Conversion Setup form.

Upgrading from Single Currency Price List to Multiple Currency Price List

If you are already using single currency price lists and want to upgrade to multiple currency price lists and agreements, the following steps are required.

Profile

The profile QP: Multi-Currency Installed controls which type of price lists you will use. Set this profile to 'Yes'. This will allow you to run the Concurrent Request program that updates all existing price lists and agreements to the multiple currency forms and functionality.

Concurrent Request

To start using the Multi-Currency feature, you have to run the concurrent program - *Update Price Lists with Multi-Currency Conversion Criteria* once. The program will create:

- A currency conversion list for each combination of Price List currency and rounding factor. This conversion will have the same Base Currency with the Base Round To from the Price list.
- Attach a currency conversion list to each price list and agreement.

This is a mandatory step since without this the pricing engine will not be able to use the current price lists as Multi-Currency price lists.

Example: There are five price lists currently set up as:

Table 6–1 Before multi-currency conversion: Price lists Currency and Round To

Currency	Round To
USD	-2
USD	-2
USD	-3
FRF	-1
CAD	NULL

The concurrent program will create four Currency Conversion Lists:

Table 6–2 After multi-currency conversion: Price lists Currency and Round To

Currency	Base Round To
USD	-2
USD	-3
FRF	-1
CAD	NULL

After running the concurrent program, all price lists and agreement will be multi-currency enabled. You can now add additional To Currencies and their conversions.

Implementation Decisions for Creating Multi-Currency Conversion Lists

Prior to using Multi-Currency price lists, the following need to be considered:

Combining Price Lists

Before creating or merging price lists and their currency conversion lists you must determine whether you can use one single price list or whether you need to create more than one.

You can use a single base currency price list, to replace price lists that are defined in various other currencies for conversions, provided that all have the following identical attributes:

- Items
- Qualifiers
- Pricing attributes for price list lines

Basis of the price list line values, of the single currency defined price lists, should equal the base currency of the single multi-currency price list, to which you attach the conversion list. Otherwise you will need to define multiple conversion criteria based upon attribute for the same To-Currency.

Example of combining single currency price lists: Qualifiers, and Items are identical in the selected single currency price lists. All of the single currency price lists have list line values that are converted values based on the USD price list. Except for CAD, all the lists apply their conversion types, equally to all items except for the CAD defined price list. CAD Price List line values equal a Fixed conversion rate = 2, EXCEPT Item C which is valued at a conversion rate of 1.5 as shown in the following table:

Table 6–3 Price lists and conversion rates

Price List = USD	Price	Price List = CAD	Price	= Conversion Rate
All Items	--	All Items	-	2
Item A	100	Item A	200	2
Item B	200	Item B	400	2
Item C	200	Item C	300	1.5

In the following table, the individual price lists for MXN, JPY, and Euro that previously were set up with the price list values equal to the conversion types have been transitioned in the setup. The CAD defined price list required an additional step to set up a conversion line for the item(s) that were not converted at the same conversion type of Fixed, with same Fixed Value.

Currency conversions will apply to the price list values on the base currency price list to which it is attached.

Table 6–4 Currency Conversion List Setup

Base Currency	USD	Conversion Type	Fixed Value	Attribute Code	Attribute Value	Start Date	End Date	Precedence
To-Currency	CAD	Fixed	2	--	--	01/01/02	112/31/02	2
To-Currency	CAD	Fixed	1.5	Item Number	Item C	01/01/02	12/31/02	1
To-Currency	MXN	Corporate	--	--	--	01/01/02	12/31/02	--
To-Currency	JPY	Corporate	--	--	--	01/01/02	12/31/02	--
To-Currency	Euro	Spot	--	--	--	01/01/02	12/31/02	--

Deactivating Price lists

You should deactivate a price list once you have merged it successfully with a multi-currency conversion list.

Rounding

There are three rounding profiles that affect rounding in Multi Currency price lists. These profiles are discussed in the Profile Options section of the Advanced Pricing Implementation Guide.

- QP: Unit Price Precision Type: Used to determine the value for the rounding factor which is defaulted on the price list.
- QP: Price Rounding: If this profile option is set to 'Enforce Currency Precision', the Base Round To cannot be updated in the currency conversion list.

- QP: Selling Price Rounding Options: This profile has optional settings to determine how the converted list price and adjustments may be rounded.

Currencies

The following must be defined in Oracle General Ledger:

- Base currency and To Currencies
- Seeded conversion types of user defined, spot, EMU fixed and corporate

Markup and Conversion Formulas

Formulas can be created and used for Markup and for Conversion Type 'Formula'. For set up information see Formulas Chapter of Advanced Pricing User Guide.

There are some limitations in using formulas in multiple currency price lists.

Base Markup Formula: The value returned by the Base Markup Formula will be used as markup for the base currency. If the Formula uses the List Price (LP) component, the formula will use the list price after applying the conversion rate. A formula with a component type as 'PLL' (Price List Line) is not allowed as a Base Conversion Formula. This is because the 'PLL' line may have a different base currency than the price list to which this currency conversion criteria is attached.

Conversion Type Formula: The value returned by the Formula selected, is the conversion rate.

A formula with a component type as PLL (Price List Line) is not allowed as a conversion formula because the PLL line may have a different base currency than the price list to which this currency conversion is attached.

A formula with a component type as MV (Modifier Value) is not allowed as a conversion formula because there is no modifier value in the currency conversion list.

Attributes

You can use different attribute values for attribute types: Product, Pricing, and Qualifier, to specify more than one conversion for the same Currency To, start date and end date.

For more information, see: *Advanced Pricing Implementation Guide* and *Advanced Pricing User's Guide*, Attribute Mapping.

Precedence

This precedence differs from attribute precedence. See Chapter 9, "Precedence and Best Price" for more information and the *Advanced Pricing User's Guide*, Multi-Currency Conversion List chapter.

This precedence, set up by the user, applies only to the currency conversion window. You must enter a precedence value in this field when setting up more than one conversion for the same Currency-To, start date, end date and having different Attribute Value. This occurs because when more than one attribute is passed from the calling application, the one with the lowest precedence value is selected by the pricing engine for conversion.

Table 6–5 Fixed and Daily conversion rates

Currency-To	Conversion Type	Attribute Code	Attribute Value	Start Date	End Date	Precedence
JPY	Fixed	Item Number	AS54888	3/1/01	3/31/01	1
JPY	Daily	All Items	All	3/1/01	3/31/01	2

In this scenario, when item AS5488 is passed from the calling application the conversion defined for Item Number= AS54888 is used by the pricing engine since it has lower precedence value = 1 as compared to the precedence for All Items = All which is 2.

Using Multiple Currency Price List with other Oracle Products

The Multiple Currency Price List feature is fully integrated with Oracle Order Management. Before using this feature, you should check with other Oracle products that integrate with Advanced Pricing to determine when they will support this feature. At the time of this writing, these products include Oracle iStore, Oracle Order Capture, Oracle Quoting and Oracle Contracts.

If Advanced Pricing is Multi-Currency enabled and you use Oracle products that do not yet support multiple currency price lists, when the calling application passes the Price List, the pricing engine matches the base currency of the price list with the order currency. If no price list is passed, the pricing engine looks for price lists whose base currency matches the order currency. The pricing engine does not look to the conversion currencies on the Multiple Currency price lists. This means that price lists still need to be defined and maintained for every single base currency used.

7

Unit of Measure

This chapter discusses implementation considerations for unit of measure (UOM). The following topics are covered:

Implementation Decisions and UOM

As you analyze and understand a client's pricing scenario, key implementation decisions must be addressed to develop a logical pricing solution. These decisions, in relation to unit of measure, are addressed in this chapter.

Key Implementation Decision: How do I adjust UOM setups to use Oracle Advanced Pricing?

- You must define conversion rates between units of measure in order to price and discount in UOMs other than the primary UOM.
- All modifier UOMs must be consistent with the UOM on the price list. Price lists and modifiers must be constructed in the same unit of measure; this is what the pricing engine expects.
- Accruals require definition of UOM and UOM class.
- You must define units of measure if you use seeded qualifiers for line volume or line weight.

Defining Unit of Measure

You must define unit of measure in order to use Oracle Advanced Pricing. UOM is used to:

- Price or discount items.
- Give non-monetary accruals as benefits. You must define a UOM class and UOM that represent your non-monetary accruals. The profile option QP: Accrual UOM Class should be set to the UOM class that you define.
- Define qualifier rules that include the seeded qualifiers line volume or line weight.

Note: Defining unit of measure is not necessary if you have already installed and setup Oracle Inventory, or if you performed this common applications setup for another Oracle product. Refer to *Oracle Inventory User's Guide, Defining Unit of Measure* for more information.

Defining Unit of Measure Conversions

To price and discount an item in a different (not primary) UOM, you must define the conversion rates between the base and other UOMs within the class. Oracle Advanced Pricing uses these rates to automatically convert transaction quantities to the primary pricing UOM. All price adjustments, benefits, and charges need to be defined in the same unit of measure as on the price list.

Note: This step is not necessary if you have already installed and setup Oracle Inventory or performed this common applications setup for another Oracle product. Refer to *Oracle Inventory User's Guide,* Defining Unit of Measure Classes, for more information.

Pricing Actions: Primary UOM/Pricing UOM

The primary UOM feature is used to price an item in different units of measure without having to explicitly define prices for each UOM. The pricing UOM is the unit of measure in which the pricing engine prices the order line.The pricing quantity is the order quantity expressed in the pricing unit of measure. Invoicing shows information based on the ordered quantity and ordered UOM.

Example:

- An item is defined with each declared as the primary UOM in a price list. You order 1 dozen. The price list does not have a price defined in dozen. The Pricing engine uses the conversion factor to calculate a pricing quantity of 12 and pricing UOM of each.
- In the Price List window, you define a price for an item/UOM combination as a price list line. In a price list, you may specify only one as primary UOM for an item.

Defaulting UOM while Creating Price Lists

When creating a new price list, the primary UOM defined in Oracle Inventory for an item will default into the price list UOM. Users can change the defaulted UOM and select the primary flag on the price list line to make this the Pricing primary UOM.

Example: Item A can have:

- Primary UOM in INV: EA
- Primary UOM in QP: CASE

Pricing Controls: Profile Options

There are three profile options critical to UOM in Oracle Advanced Pricing: QP: Accrual UOM Class, QP: Line Volume UOM Code, and QP: Line Weight UOM Code.

QP: Accrual UOM Class

QP: Accrual UOM Class specifies the UOM class used to define accrual units of measure. The modifier setup window displays all units of measure in this class when entering the benefit UOM for an accrual.

- Default value: none
- Required if your business gives non-monetary accruals as benefits
- All UOM classes are defined to Oracle Applications
- Visible and can be updated at the site and application level

QP: Line Volume UOM Code

Specifies unit of measure of the line volume qualifier. The attribute sourcing API converts the item on the request line to its primary UOM. It then uses the volume attributes of the item to derive the line volume of the item in the specified UOM.

- Default value: none
- Required if your business must define qualifier rules that include the seeded qualifier line volume
- All units of measure currently defined to Oracle
- Visible and can be updated at the site and application levels

QP: Line Weight UOM Code

Specifies the unit of measure of the line weight qualifier. The attribute sourcing API converts the item on the request line to its primary UOM, and uses the weight attributes of the item to derive the line weight of the item in the UOM specified in this profile option.

- Default value: none
- Required if your business needs to define qualifier rules which include the seeded qualifier line weight

- Specifies UOM of the line weight qualifier. The attribute sourcing API converts the item on the request line to its primary UOM, then uses the weight attributes of the item to derive the line weight of the item in the specified UOM.
- All units of measure currently defined to Oracle
- Visible and can be updated at the site and application levels

8

Multiple Organizations

This chapter discusses multiple organizations in relation to Oracle Advanced Pricing. The following topics are covered:

Multiple Organizations, Pricing Security, and Pricing Actions

Oracle Advanced Pricing does not use the multiple organization structure of Oracle Applications. However, pricing security features are provided that enable you to:

- Assign or reassign Operating Unit ownership to price lists and modifiers.
- Control which operating units can use pricing entities when pricing transactions.

See Chapter 4, "Pricing Security" for more information on security features and operating unit assignments (including Global Usage).

Using Qualifiers to Create Multiple Organizations

By using qualifiers, you can create price lists and modifiers that are operating unit specific. Operating unit is not a seeded qualifier. It is necessary to set up the attribute and sourcing rules. Once the qualifier attribute of operating unit is available for use, it can be applied to price lists and modifiers. When a call is made to the pricing engine, only those price lists and modifiers for the specified operating unit are eligible.

QP: Item Validation Organization

You must set the profile option QP: Item Validation Organization to the level in your organization hierarchy at which you set prices when using Oracle Inventory. This indicates the Oracle Manufacturing organizations that items are validated and viewed against when entering price lists or modifiers. This profile value should be set to the master organization.

Modifier Type: Item Upgrade

When you define item relationships for item upgrade type of modifiers, the item relationship must belong to the same item organization as is specified in QP: Item Validation Organization.

Cross Order Volume Loader

The concurrent program accumulates all cross order totals for an operating unit, but it does not summarize across operating units. The engine only considers orders that a customer places with this sales organization, and does not consider orders that the customer places with other operating units. When running the cross order volume

loader you may specify that the load run for one operating unit. If specified as null, it summarizes for all operating units for which there are orders (assuming there is a multi-org install). If loading for multiple operating units, each summary is within an operating unit.

Organization Specific Seeded Qualifiers and Pricing Attributes

Some of the seeded qualifiers attributes and pricing attributes in Oracle Advanced Pricing are specific to an operating unit. The following is a list of seeded qualifier and pricing attributes that are operating unit specific:

Table 8–1 Seeded qualifier and pricing attributes that are operating unit specific

Type	Context	Qualifier Attribute
Qualifier	Customer	Site use
Qualifier	Customer	Bill to
Qualifier	Order	Line type
Qualifier	Order	Ship from
Qualifier	Volume	Period 1 order amount
Qualifier	Volume	Period 2 order amount
Qualifier	Volume	Period 3 order amount
Pricing	Volume	Period 1 item quantity
Pricing	Volume	Period 2 item quantity
Pricing	Volume	Period 3 item quantity
Pricing	Volume	Period 1 item amount
Pricing	Volume	Period 2 item amount
Pricing	Volume	Period 3 item amount

Price List LOV in Oracle Order Management

Because price lists are not operating unit specific, price lists displayed in the price list LOV at the sales order header and line level are not specific to the operating unit to which OM is set. All operating units' price lists are visible. This also applies when a qualifier attribute is attached to a price list. The pricing engine validates a qualifier of an operating unit only once the sales order or line has been sent to pricing.

9

Precedence and Best Price

This chapter discusses implementation considerations surrounding incompatibility resolution using precedence and best price in Oracle Advanced Pricing. The following topics are covered:

Overview

Incompatibility occurs when the pricing engine finds more than one price or modifier to return, but engine and user-defined rules prohibit application of more than one of them. The engine resolves this incompatibility using either precedence or best price.

Phase Incompatibility Resolution

Incompatibility resolution is defined for each engine phase. There is an Incompatibility Resolve Code field on the Event Phases window. This code can be set to precedence or to best price. With the exception of Phase Sequence 0 - List Line Base Price, this code can be changed from the default value to reflect how incompatibility should be resolved for the specified phase. See the Events and Phases chapter of this manual for more information on phases.

Precedence

When incompatibility is encountered between price lists or modifier lists and the incompatibility resolve code for the phase is precedence, the engine attempts to resolve the incompatibility by ordering the qualifier attributes and item context attributes from highest to lowest priority (number 1 having the highest priority). The qualifier or item attribute with the highest priority has precedence over all other attributes, and the engine selects the price list line or modifier list line to which this attribute belongs.

Best Price

When incompatibility is encountered between modifier lists and the incompatibility resolve code for the phase is best price, the engine attempts to resolve the incompatibility by finding the modifier that gives the best price. Best price is the highest modifier value that calculates the highest discount value.

Default Precedence Numbers

The segment numbers defined in the qualifier contexts and item context pricing context in the Descriptive flexfield are defaulted onto the price list lines/modifier as precedence numbers. The precedence number is used by the pricing engine to determine specificity. The precedence number defined for each attribute determines which qualifiers or items have priority over others when the pricing engine is forced to choose between multiple prices or incompatible modifiers. Ordering of qualifier attributes and item context attributes is known as precedence.

Warning: Do not change seeded segment numbers; this affects future upgrades! Users are able to change the seeded values in Qualifier, Price List, and Modifier forms.

Matched Qualifiers for Modifiers/Price Lists

The pricing engine only evaluates matched qualifiers for a price list or modifier when ordering for precedence. A matched qualifier is described as those attributes that are evaluated as true. Because qualifiers can be defined as OR conditions, some qualifier attributes that exist on the setup of the price list or modifier might not be chosen by the engine, or in other words, are not qualified by the engine. The pricing engine only uses matched qualifiers when ordering the qualifier attributes to determine which has the highest priority.

The following table lists several of the seeded qualifier contexts and qualifier attributes with the segment number for each attribute that are on a specific modifier list. The qualifier attributes of agreement type and customer class have the same grouping number, and order type is in a different grouping number. For the engine to select the modifier, an order must have either agreement type and customer class be true or order type be true. In this example, order type is true and agreement type and customer class are false. Order type is a matched attribute. Only the value for order type is used when evaluating precedence.

Table 9–1 Seeded qualifier contexts and qualifier attributes with segment number

Grouping Number	Context Type	Context	Attribute	Segment Number	Matched?
1	Qualifier	Customer	Agreement type	240	No
1	Qualifier	Customer	Customer class	310	No
2	Qualifier	Orders	Order type	470	Yes

Precedence and Pricing Attributes

The segment numbers for the attributes defined in the pricing context (excluding item context) in the descriptive flexfield do not have significant meaning for engine evaluation in determining precedence.

Price List Incompatibility Resolution

The pricing engine is coded to calculate base price from the price list in phase sequence 0. Phase 0 is coded with the resolve incompatibility code as precedence. The user cannot change this setting. When the pricing engine determines that a pricing request is eligible for a price from more than one price list, the engine attempts to resolve the incompatibility by ordering the qualifier attributes and item attributes described in the precedence resolution section of this chapter. If two or more price list lines are matched and have the same precedence, the engine selects the price list line with the higher pricing attribute count. If the engine cannot resolve this incompatibility between price lists, it will return an error message to the calling application that it cannot apply a price and will return the names of the price lists that this incompatibility resides.

Modifier Incompatibility Resolution

The incompatibility resolution codes for phases associated with modifier processing can be changed from the seeded values to reflect the business need. The exception to this is for Phase 0, in which the code cannot be changed.

Best Price Resolution for Modifiers

Incompatible modifiers can be across modifier types, therefore when the engine resolves incompatibility by best price, the engine needs to evaluate the modifiers on a similar basis. The engine does this by calculating a common benefit percent for each modifier. The attached chart shows which column the engine uses to evaluate best price for each modifier type:

Table 9–2 Modifier types and values

Type of Modifier	Modifier Value
Discount/surcharge: percentage	List price%
Discount/surcharge: amount	Amount
Discount/surcharge: new price	List price - new price
Discount/surcharge: lumpsum	Lumpsum amount/line quantity
Price Break	Best price comparison for price break is based on the value of the matching break modifier and its list price *%.

Table 9–2 Modifier types and values

Type of Modifier	Modifier Value
Terms substitution	Estimated discount value of modifier such as Comparison Value. If not provided then best price is set to zero.
Item upgrade	Estimated discount value of modifier such as Comparison Value. If not provided then best price is set to zero.
Coupon issue	Estimated discount value of modifier such as Comparison Value on the coupon issue line. If not provided then best price is set to zero.
Other Item discount	Not included - always valued at zero
Promotional goods	Not included - always valued at zero
Freight Charges	Same as discount and surcharges.

The pricing engine does not consider modifiers with formulas for best price calculation.

Best Price Calculation Ignores Buckets

The engine ignores buckets when selecting modifiers and determining best price calculation. Best price is calculated off of the list price. For example, an item has a $100 list price. The engine determines that the item is eligible for modifiers A, B and C. Modifiers B and C are incompatible and the engine must resolve the incompatibility by best price. For modifier B, the engine calculates:

100 - (100 - 75) = 75

Modifier C is calculated as:

100 - (100 * 0.125) = 87.5

Modifier B is selected because it gives the better price.

Modifier	Incompatibility	Bucket	Application Method	Value	Calculate Best Price	Engine Selection
A	Not applicable	1	Percent	20	=	Yes
B	Level 1	2	New price	75	75	Yes
C	Level 1	2	Percent	12.5	87.5	No

If only eligible modifiers are sent to the calculation engine, these modifiers are calculated in the correct bucket. In this example, the calculation engine takes the list price, $100, subtracts the bucket 1 discount, 20, for a selling price of $80. Next the engine calculates the bucket 2 discount off of the $80. Because modifier B is a new price discount, a discount of $5 is created and the new selling price is $75.

Calculating Common Benefit Percent

The engine needs a common basis to estimate modifier types to determine which modifier yields the best price. The engine accomplishes this by calculating the common benefit percent. For some modifier types, there is no stated value that the engine can use to evaluate best price, thus the engine uses the value the user inputs in the Comparison Value column on the modifier summary line. The engine calculates the benefit percent by taking the Comparison Value as the numerator divided by the list price of the item that the modifier will be applied as the denominators. This benefit percent is compared to the other benefit percents for the modifier lines that are being evaluated. The highest benefit percent number is the modifier that yields the best price. This is the modifier that the engine will apply.

Note: Comparison Value is a user-entered field. It is the value that the engine uses to calculate the common benefit percent.

Header and Line Qualifiers for Modifiers

To determine precedence, the engine selects all qualified header level and line level qualifiers and item context for the modifier line. The engine then chooses the attribute with the lowest precedence number. This attribute is used for incompatibility resolution.

Modifier Incompatible Precedence Resolution

When the incompatibility resolve code for a phase is precedence, and more than one modifier line is eligible, the engine resolves the incompatibility as follows (for this example, assume that all qualifier attributes are matched):

Table 9–3 Modifier Incompatible Precedence Resolution

Modifier	Context Type	Context	Attribute	Segment Number	Select Precedence	Resolve Incompatibility
A	Qualifier	Customer	Agreement type	240	Yes	No
A	Qualifier	Customer	Customer class	310	No	No
A	Pricing	Item	Item number	300	No	No
B	Qualifier	Order	Order type	470	No	No
B	Pricing	Item	Item category	290	Yes	No
C	Qualifier	Order	Order amount	100	Yes	Yes
C	Pricing	Item	Item number	200	No	No

First, the engine will select the attribute with the lowest precedence number for each of the modifiers. For modifier A, agreement type is selected because it has the lowest number. For modifier B, item category is selected and for modifier C, order amount is selected.

The engine resolves incompatibility by ordering the precedence numbers for the three modifiers. Because modifier C has the lowest number, it is selected by the engine as the highest priority and is the modifier returned by the engine.

When the incompatibility resolve code for a phase is precedence and the engine cannot resolve incompatibility between two or more modifiers through precedence, the engine resolves incompatibility using best price resolution. If two or more modifiers result in the same best price, the engine randomly selects one modifier to return to the calling application.

Modifier Incompatibility Setup window

The Modifier Incompatibility Setup window enables you to query modifiers by phase sequence, phase name and incompatibility group name. Because modifiers are incompatible within a phase, the query is phase-specific; this makes is simple to identify which resolve method the engine uses to process incompatibilities. You can make changes to the incompatibility level of a modifier directly on this window. The window displays various modifier line details that provide a cross modifier view of modifiers that are incompatible with each other.

The product precedence field displays for each modifier line that can be used in identifying discounts that have a higher precedence. However, this view is

incomplete; it does not display qualifier attribute precedence. These attributes are evaluated by the engine during resolution. This window contains a direct link to display the modifier detail where you can review and revise modifier details.

Incompatibility Resolution Examples

Price List Precedence Incompatibility Resolution

The following table lists several of the seeded qualifier contexts, qualifier attributes, and the item context and item number attribute with the segment number for each attribute.

Table 9–4 Price List Precedence Incompatibility Resolution

Context Type	Context	Attribute	Segment Number	Price List
Qualifier	Customer	Agreement type	240	B
Qualifier	Customer	Customer class	310	A, B
Qualifier	Orders	Order type	470	A
Pricing	Item	Item category	290	A, B

For example, Price List A has qualifier Customer Class = VIP Customer. Price List B has qualifiers Agreement Type = Yearly and Order Type = Special. Both price lists contain prices for Item Category Z. Item X is part of Item Category Z. There is not a price list defined for Item X.

An order is placed in Oracle Order Management for Item X. When Oracle Order Management sends the pricing request without a price list name to the pricing engine, the pricing engine must search for a price list for Item X. The pricing engine will not find a price for Item X, but will find that Item X belongs to Item Category Z. Item Category Z is on both Price List A and Price List B. Because the engine can return only one price for an item, the engine must determine which price list to select.

To resolve this, the engine first evaluates the qualifier attributes and item context attribute on Price List A. On Price List A, customer class has a segment number of 310 and item category has a segment number of 290; therefore the engine selects item category as the highest precedence on Price List A. On Price List B, agreement type has a segment number of 240 and order type has a segment value of 470; therefore the engine selects agreement type with a number 240 to compare to the item context. Item category with a segment number of 290 is compared to

agreement type with a number of 240. Agreement type has a lower number and has the highest precedence on Price List B. The engine orders the precedence numbers for Price Lists A and B in a sequence of the lowest number to the highest. The engine chooses the lowest number (the number with the highest precedence).

The engine will order the segments as outlined in the following table:

Table 9–5 Segment order

Context Type	Context	Attribute	Segment Number	Price List
Qualifier	Customer	Agreement type	240	B
Pricing	Item	Item category	290	A, B

Because agreement type has a precedence of 240, the pricing engine selects the price of Item X from Price List B and returns this information to the calling application.

Note: If the calling application sends a validated price list to the engine in the pricing request and is qualified to receive price from this price list, the engine does not perform precedence resolution; it returns the price from the designated price list. Using the example, if Oracle Order Management sends the pricing request to use Price List A and the eligibility rules are met, the engine returns the price from Price List A to Oracle Order Management and does not need to perform incompatibility processing.

Note: The default precedence numbers can be changed by the user on the price list window at the time of setup.

Modifier: Precedence Incompatibility Resolution

In the following example, the engine selects the following modifiers based on precedence resolution:

For this example, assume that all qualifier attributes are qualified.

For this list line adjustments phase:

- The Preferred Discount is applied because there are no other modifiers in this incompatibility level.
- The XYZ Brand Discount and the Summer Promotion are in the same phase and same incompatibility group. All Items has the lowest precedence number for Summer Promotion - 315. The XYZ Brand Discount has an item category precedence of 290. The XYZ Brand Discount is selected over the Summer Promotion because the precedence for item category at 290 has the highest priority.

For line charges phase:

- The Repack Charge is applied because there are no other modifiers in this incompatibility level.

For header level adjustments phase:

- The New Site Discount is exclusive and therefore the only modifier that is applied in this phase.

For header level charges phase:

- The Handling Charge is applied because there are no other modifiers in this incompatibility level.

Table 9–6 Modifier: Precedence Incompatibility Resolution

Pricing Phase	Modifier	Incompatibility Group	Qualifier/Product Attributes	Precedence	Engine Selection
List line adjustments	Preferred discount	Level 1	Customer class	310	Yes
			Item number	220	
List line adjustments	Summer promotion	Level 2	Sales channel	320	No
			All items	315	
List line adjustments	XYZ brand discount	Level 2	Item category	290	Yes
Line charges	Repack charge	Level 1	Item number	220	Yes

Table 9–6 Modifier: Precedence Incompatibility Resolution

Pricing Phase	Modifier	Incompatibility Group	Qualifier/Product Attributes	Precedence	Engine Selection
Header level adjustments	New Site discount	Exclusive	Site use	270	Yes
			Item all	315	
Header level adjustments	Order amount discount	Level 1	Customer name	260	No
			Item number	220	
Header level adjustments	Independence Day promotion	Level 1	Sales channel	320	No
			All items	315	
Header level charges	Handling charge	Level 1	None	None	Yes

Modifier: Best Price Incompatibility Resolution

The pricing engine determines that Modifier A and Modifier B are incompatible. The incompatibility resolve code for the phase is set to best price. Modifier A is a 10% discount and Modifier B is an other item discount with a Comparison Value of 200. The list price for the item that the modifier will be applied to is $1000. The engine calculates the common benefit percent to determine which modifier has the highest number.

Table 9–7 Modifier: Best Price Incompatibility Resolution

Modifier	Value	Calculation	Common Benefit Percent	Highest Number
A	10%	None	10	No
B	200 Comparison Value	200 Comparison Value/1000 list price	20	Yes

Modifier B has the highest common benefit percent and is applied as the best price discount.

10

Attribute Management

This chapter discusses implementation considerations for attribute mapping. The following topics are covered:

Overview of Attribute Management

Pricing objects in Oracle Advanced Pricing enable users to define pricing actions and pricing rules for a given business process. Through price lists, modifiers, and formulas, these pricing actions provide the ability to define prices, price adjustments, and other benefits. By defining pricing rules such as qualifiers and pricing attributes, these pricing rules are used to drive your pricing actions. The attributes can be used as elements to express your product or customer hierarchy.

A *product hierarchy element* is a level at which pricing characteristics can be defined. Examples of product hierarchy are item number, category, or brand. Customer hierarchy is an example of a qualifier hierarchy, such as customer group.

Through Attribute Management, the Pricing Engine receives all the values of the attributes defined in the qualifier and pricing attributes in order to determine which price lists and modifiers the transaction is qualified for. The data sources for the qualifiers and pricing attributes can be within Oracle Applications or from outside Oracle Applications. Users can use the feature of Attribute Mapping to easily extend Pricing to tap into data from a wide variety of non-standard sources to drive their pricing actions. In the attribute management windows, users can create new attributes, update existing attribute properties, or disable attributes.

The calling application first uses the attribute mapping API to get all qualifier and pricing attributes associated with the transaction. Then the calling application makes a call to the pricing engine. The pricing engine evaluates these values to determine which price lists and modifiers are eligible for the transaction. The data sources for the qualifiers and pricing attributes can be within Oracle Applications or from outside Oracle Applications.

The Attribute Management feature enables you to create new contexts and attributes, update existing context/attribute properties, or disable existing contexts. Creating new context/attributes extends Oracle Advanced Pricing's ability to provide user defined data sources to drive pricing actions. The three methods to source data for an attribute are:

- User Entered (attribute value is entered by user)
- Custom Sourced (custom code is used to derive an attribute)
- Attribute Mapping (Pricing Engine derives information from other Oracle Applications and non-Oracle data sources.

The following sections discuss the steps needed to create attributes using attribute mapping.

Terminology for Attribute Management

Pricing Transaction Entity

A new entity has been created called a Pricing Transaction Entity. A Pricing Transaction Entity (PTE) is an ordering structure that has associated Request Types and Source Systems. Different applications may have different request structures when they make requests to the pricing engine.

Source System

The Source System is the application that captures the pricing setup data.

Request Type

The Request Type identifies the type of transaction that is being priced. Different applications make requests to the pricing engine. Request types of these applications may be different. Some applications may share their request types.

For example: iStore and Order Capture share the same request type. On the other hand, Order Management and iStore have different request structures.

Pricing Transaction Entity

A Pricing Transaction Entity consists of a group of applications that point to the same setup data and attributes, and includes Request Types and Source Systems. The Source System is the application that captures the pricing setup data. The Request Type identifies the type of transaction that is being priced.

Different applications may have different request types when they make requests to the pricing engine. For example, Istore and Order Management can be part of one Pricing Transaction Entity because they are both Order Fulfillment systems and share the same setup data. Conversely, Oracle Transportation resides in a different Pricing Transaction Entity and has its own attribute and setup information.

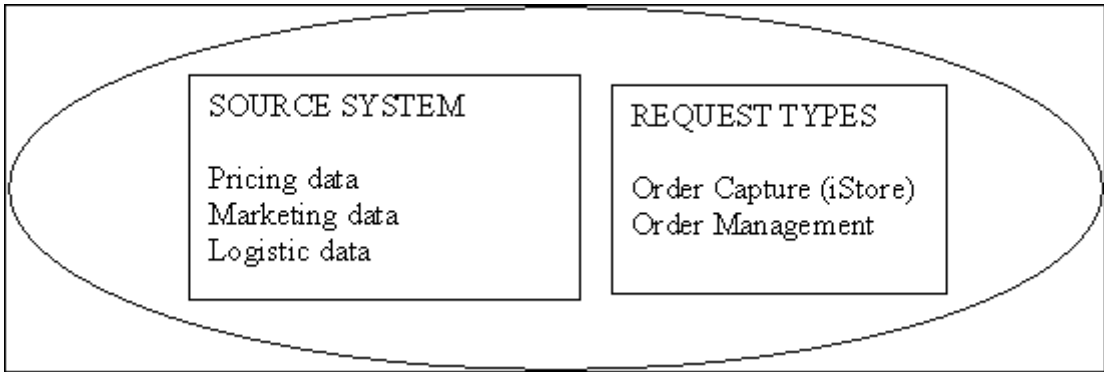
The Pricing Transaction Entity narrows the data the search engine reviews. The search engine examines only the setup data generated by the source systems defined for that Pricing Transaction Entity. All applications belonging to the same pricing transaction entity have the same set of attributes available to them. A same PTE ensures that the applications sharing the same data always give the same price for an item irrespective of the request type.

Examples of a Pricing Transaction Entity

In the example below, the Pricing Transaction Entity is Order Fulfillment. Within this PTE, there are several source systems. Istore, Order Capture, and Order Management will be evaluating pricing data defined in all these source systems.

The following graphic displays the Pricing Transaction Entity - Order Fulfillment:

Figure 10–1 Pricing Transaction Entity - Order Fulfillment

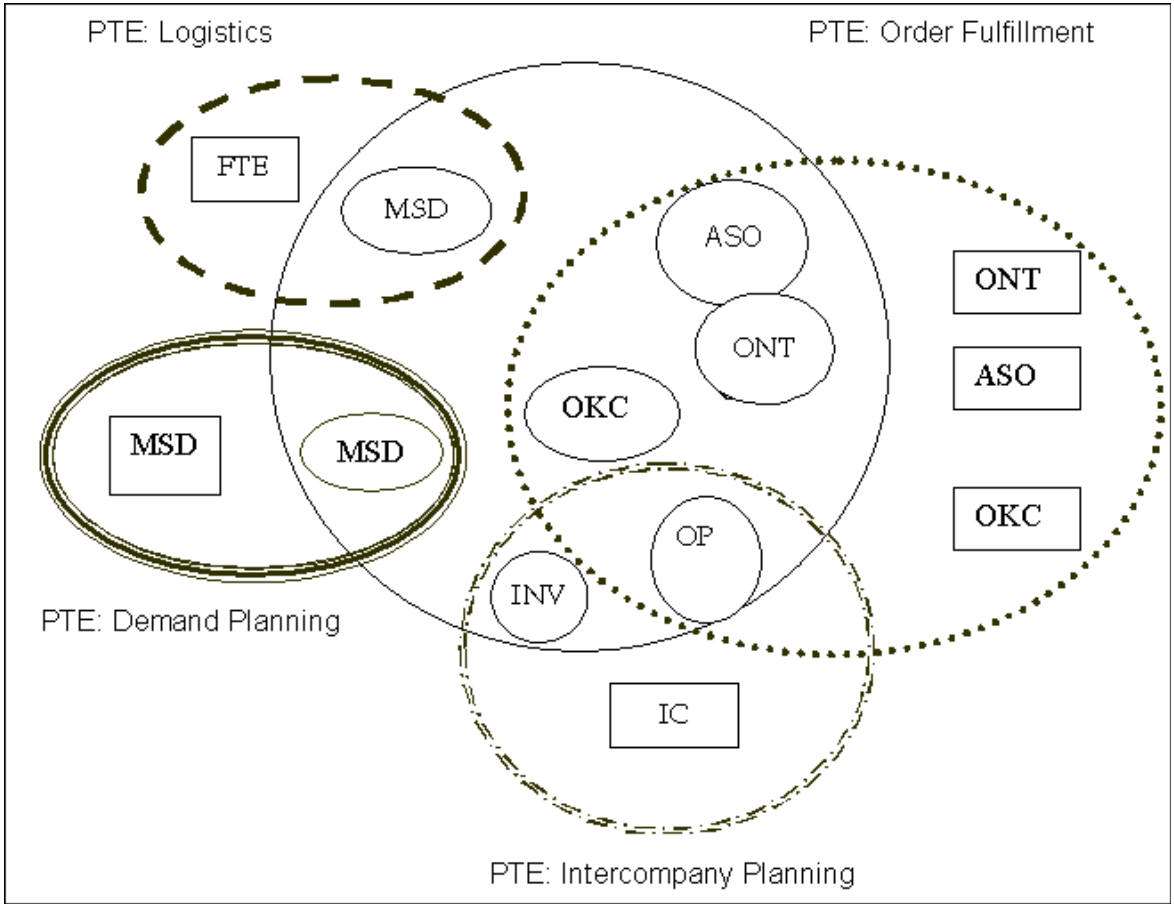


Seeded PTE's in Advanced Pricing

There are four seeded PTEs in Oracle Advanced Pricing.:

- Order Fulfillment
- Intercompany Transaction
- Logistics
- Demand Planning

Figure 10–2 Four seeded PTEs in Oracle Advanced Pricing



The following tables display the Request Types and Source Systems seeded for each PTE:

Order Fulfillment

Table 10–1 Order Fulfillment: Request types and source systems

Request Types	Source Systems
ONT	ASO

Table 10–1 Order Fulfillment: Request types and source systems

Request Types	Source Systems
ASO	AMS
OKC	QP
--	OKC

Intercompany Transaction

Table 10–2 Intercompany Transaction: Request types and source systems

Request Types	Source Systems
IC	INV
--	QP

Logistics

Table 10–3 Logistics: Request types and source systems

Request Types	Source Systems
FTE	FTE

Demand Planning

Table 10–4 Demand Planning: Request types and source systems

Request Types	Source Systems
MSD	QP

When to Define a New Pricing Transaction Entity

When a new request type is created (for example, a new ordering structure is developed), the business must decide if the request type will use the same set of source systems in the existing PTE. A new PTE needs to be created only if the new request type uses a different ordering structure and a different set of source systems. Request types have to be unique across PTEs.

In the example below, the Pricing Transaction Entity is Order Fulfillment. Within this PTE, there are several source systems. Istore, Order Capture, and Order Management will be evaluating pricing data defined in all these source systems.

Tasks for Extension For A New Attribute: End to End Flow for Implementation

1. Setting up a Pricing Transaction Entity Verify the request types and source systems for a given PTE. Otherwise create a new pricing Transaction Entity in QP Lookups.
2. Create a new Context.
3. Define the attribute for that context to be used for Pricing Setup.
4. Link the attribute to a Pricing Transaction Entity (PTE).
5. Define properties of the attribute for the given pricing transaction entity.
6. For mapped attributes, define Attribute Mapping Rules.
7. Use the attribute in a valid pricing setup.
8. Running the Build Attribute Mapping Rules Program.
9. Enter an order.
10. Verify the mapped and user enter attributes have been correctly passed to the pricing engine from the Pricing Engine Request Viewer.

Overview of How Attributes Mapping works at Run time

1. Calling application calls QP_Attr_Mapping_PUB.Build_Contexts. This routine executes the static generated attribute mapping routines and returns the attributes to the calling application.
2. Calling application then appends the user entered attributes and asked for qualifiers to this request.
3. Calling application then calls the pricing engine with these mapped attribute values.
4. Pricing engine also appends a few internally mapped attributes to this request.
5. Pricing engine processes the request and returns the results to the calling application

Defining Pricing Transaction Entity

The entity Pricing Transaction Entity will help to slice the data search engine needs to look at. Search engine will look at only the setup data generated by all the source systems defined for that Pricing Transaction Entity. It also makes contexts of one Pricing Transaction Entity unavailable to other Pricing Applications families.

If you need to define a new Pricing Transaction Entity, follow the steps below. Otherwise, please go to the section on Pricing Transaction Entity Associations.

To define a new Pricing Transaction Entity:

1. Navigate to the Pricing Lookups window.
2. Query on QP_PTE_TYPE as lookup type
3. Enter a code (short name) for a Pricing Transaction Entity.
4. Enter Meaning.
5. Enter a Description.
6. Enter optional fields.
7. Save the record.

Pricing Transaction Entity Association

In this window, you can define the request types and source system for a given Pricing Transaction Entity.

To add a new source system/request type to the Pricing Transaction Entity:

1. Navigate to the Pricing Transaction Entity - Source System and Request Types window.
2. Select the Source systems/Request Types tab.
3. Enter the required information.
4. Select the Source Systems tab.
5. Select a source system application from LOV
6. Description is display only.
7. Select the Enabled box to activate the Source System.
8. Select the Request Types tab.
9. Enter a code (short name) for a request type from.

10. Enter a description.
11. Enter a header level global structure name
12. Enter a line level global structure name
13. Enter a header level view name.
14. Enter a line level view name.
15. Select the Enabled box to activate the request type.
16. Save the record.

Exceptions: A Request type cannot be created for more than one Pricing Transaction Entity.

A Request type cannot be deleted, if it has a mapping rule defined for it. A Source System cannot be deleted, if it is used by a setup for a given Pricing Transaction Entity.

Source Systems tab

Name: Name of a Pricing Transaction Entity.

Description: Description of a Pricing Transaction Entity.

Source System: It is an application generating setup data. For example istore, Oracle pricing and AMS generate modifiers. Hence these applications could be source system.

Code: A short name for the source system application.

Description: Description for a source system.

Request type Tab

Code: A short name of a request type.

Description: Description of a request type.

Header/Line structure/view: A request type may have a global record structure or a view defined to map the data.

The user will either enter global structures for header and line or view names for header and line. All of these four fields are optional.

If data is entered in any of these fields, based on this data a list of values (LOV) will be provided on ValueString field of the Attribute Mapping setup window. If no

data is entered in any of these fields, no LOV will be provided on Value String field of the Attribute Mapping setup window.

Header Global Structure: The global Structure for the Header.

Line Global Structure: The global Structure for the Line.

Header view name: View name for the header

Line view name: View name for the line.

Global Structures

The table below shows examples of the global structures at the order level and line level for the different request types of the Order Fulfillment Pricing Transaction Entity.

Table 10–5 Global structures

PTE	Request Type	Order Level Global Structure	Line Level Global Structure
Order Fulfillment	Order Capture	ASO_PRICING_INT.G_ HEADER_REC	ASO_PRICING_INT.G_ LINE_REC
Order Fulfillment	Oracle Contracts Core	OKC_PRICE_PUB.G_ CONTRACT_INFO	OKC_PRICE_PUB.G_ CONTRACT_INFO
Order Fulfillment	Order Management Order	OE_ORDER_PUB.G_HDR	OE_ORDER_PUB.G_LINE
Intercompany Transaction	Intercompany Invoicing	INV_IC_ORDER_PUB.G_ HDR	INV_IC_ORDER_PUB.G_ LINE

Creating Contexts and Attributes

For information on creating contexts and attributes, see: *Oracle Advanced Pricing User's Guide*.

Note: You cannot add new contexts of Product Context type. However, you can add attributes to the existing Product context ITEM.

Deleting Contexts and Attributes

User cannot delete a context, if it has one or more attributes.

Attributes used in Pricing Setups windows cannot be deleted.

Warning for Pricing Attributes and Flexfields tables

A context of type Pricing Attribute will also be created as Flexfield Pricing Contexts in the Flexfield, once it is created using the Contexts and Attributes window. All the above definitions of contexts and attributes are stored in QP tables. However, the same data will be stored in flexfield tables as well, if the context type is Pricing Attribute. This occurs because Order Management (OM) uses flexfield structure for pricing attributes. Hence for pricing attributes to display on OM's windows, these definitions need to be in the flexfield tables as well.

However, Pricing Attribute setup must always be done in the Context and Attributes setup menu. If user does a pricing attribute setup in flexfield tables, the setup data will not be available in the QP tables as data is not copied from flexfield tables to QP tables. So even though the pricing attribute displays in OM windows, it will never be picked up by the engine.

A context and attributes of type Pricing Attribute will get updated and deleted in the flexfield, if updated and deleted in the Context and Attributes window if it meets the deletion criteria.

To create a context or attribute of Pricing Attribute type in Flexfield tables:

1. Context of type Pricing Attribute will also be created as Flexfield Pricing Contexts in the Flexfield, once it is created using the Contexts and Attributes window.
2. Context of type Pricing Attribute will also get registered automatically, if the mapped column is greater than PRICING_ATTRIBUTE30.

3. Once the context of type Pricing Attribute is created and registered, the system will automatically compile the flexfield, in the background. The user can follow the stage of compilation, by viewing his concurrent requests.
4. Descriptive flexfields in Advanced Pricing for Pricing Context gets mapped to Pricing Context and Product Context (context=ITEM only). All qualifier context gets mapped Qualifier Context.

Attribute Linking and Mapping

The following steps describe the process for linking attributes to a Pricing Transaction Entity.

1. Navigate to the Pricing Transaction Entity - Linking Attributes window.
2. Find the Pricing Transaction Entity by choosing View > Find or select the Search icon. The Pricing Transaction Entity must have at least one Request Type, in case of Advanced Pricing.
3. Alternatively, use the LOV to select a Pricing Transaction Entity.
4. Select the Context Type.

   The system queries up all the contexts for the context type and displays them in the Contexts region.
5. Select Show All Contexts to view all the contexts in the Contexts region. If the Show All Contexts box is cleared, only the contexts that have been linked to the Pricing Transaction Entity display in the Contexts region.
6. Click the Context button to revisit the Context Setup window. You may add or modify the selected context or open up the Context window in a new mode.
7. Click Link Attributes to display the Link Attributes window.
8. Place the cursor on the blank record and select Code from the LOV. You can change Level, Attribute Mapping Method LOV Enabled, Use in Limits and Attribute Mapping Enabled boxes. The Attribute Mapping Status box is display only.
9. Add more attributes belonging to same context. Every time you add a new attribute, the Code LOV will get decremented by the Attribute that has already been added. The maximum rows that can be added will equal the number of attributes, for that context.
10. Save the record.

    You have linked the attributes to the Pricing Transaction Entity.

Attribute Mapping Methods

There are three different attribute mapping methods to define how an attribute is mapped:

USER ENTERED: The attribute value is entered by the user; therefore, attribute mapping is not required.

If the Attribute Mapping Method is USER ENTERED and the context type is Pricing, you must complete the following steps to view this attribute in the Order Management Sales Order pad:

a. Navigate to the Flexfields window.

b. Select the Freeze Flexfield Definition box.

c. Click Compile to compile the flexfield.

See *Oracle Advanced Pricing User's Guide* for more information on creating contexts and attributes, and setting up a user-defined attribute.

> **Note:** The attributes ITEM AMOUNT and ITEM QUANTITY are user-entered but are sourced internally by the pricing engine.

CUSTOM SOURCED: A code must be provided by the user to source an attribute. Attribute mapping is not required as the attribute will be sourced by Get custom attributes. For more information, see: Using Custom Sourced Attributes on page 10-19.

Confirm that the profile option QP: Custom Sourced is set to Yes. The Build Contexts program builds the contexts from the dynamic package generated by the Build Attribute Mapping Rules program and from the custom package created by the customers.

ATTRIBUTE MAPPING: Attribute mapping is required and an attribute mapping rule needs to be defined. To create a mapped attribute, complete the following in the Link Attributes window:

Attribute Level: Select the Attribute Level: BOTH, LINE, ORDER.

For example, for an attribute Agreement Id, if the level is ORDER it means that only the Agreement Id at the header level will be attribute mapped. The Agreement Id on order lines will not be attribute mapped. This level is also used to selectively display pricing attributes/qualifiers in the Pricing Setup list of values.

Attribute Mapping Method: Select ATTRIBUTE MAPPING. Note that the Attribute Mapping button becomes enabled. The button is grayed out if the Attribute Mapping Method is USER ENTERED or CUSTOM SOURCED.

LOV Enabled: Select this box to display the attribute in the LOVs of modifier, qualifier, price list, and formula windows. If not selected, the attributes will not be available in the modifier, qualifier, formula, and price list setup windows. However existing modifiers defined using that attribute, will behave the same way.

If LOV Enabled is selected and the Attribute Mapping Enabled box is cleared, then even though the attribute is available for setup, a mapping rule is not created. So if the attribute is used in the qualifier, modifier, price list, and formula setup windows, an error will be raised.

The LOV Enabled box is selected for all seeded attributes to make them available in the LOVs from price list, qualifier, modifier, formula and other setup windows. This box will be untouched whenever data upgrade scripts are run.

Use in Limits: Select this box to display the attribute in the Attribute LOVs in Limits setup windows (including the Other Attributes window).

Attribute Mapping Enabled: Select this box to enable the attribute to be used in Attribute Mapping. If selected, the concurrent program Build_context API will generate the code for this attribute.

If not selected, the attribute will not be mapped. This box is cleared for all seeded attributes to avoid the mapping of unwanted attributes. As a part of setup, you need to select this flag for the attributes to be mapped.

Attribute Mapping Status: This box is selected (or cleared) automatically by the concurrent program which generates the Build_Contexts API for mapped attributes. The Attribute Mapping Status box is cleared:

- For new attributes until the concurrent program is run after defining the attribute mapping for it.
- When an attribute mapping rule is modified.
- If the Attribute Mapping Status box is cleared and 1) Attribute Mapping Method is ATTRIBUTE MAPPING and 2) Attribute Mapping Enabled is selected, then the concurrent program must be run to regenerate the Build_ Contexts api. The Attribute Mapping Status box is selected when the concurrent program is run successfully. The field is non-navigable.

Click Attribute Mapping to display the Attribute Mapping window. In the window, define the mapping rules for each request type. The Attribute level determines the mapping rules that need to be created.

Attribute Level Restrictions On Pricing Setup windows

Qualifiers Setup: The list of values for the qualifier attribute on the Qualifier Group setup window will show all levels of qualifier attributes. However, a qualifier group cannot have one qualifier attribute with the level ORDER and another qualifier attribute with the level LINE within the same Qualifier Grouping No.. All the qualifiers within a Qualifier Grouping No must have the attributes with the level either as ORDER or BOTH or as LINE or BOTH.

Price List Setup: The list of values for qualifier attributes in the Qualifiers tab will show qualifier attributes with the attribute levels of LINE or BOTH. An order line may qualify for a price list based on a qualifier attribute of the order line or both the order header and order line.

The list of value for Pricing Attributes will show the pricing attributes with the attribute level of Line.

Formula Setup and Factor List Setup: Since a formula can be applied either to the order or line level, it is not possible at the definition time to restrict the attributes appearing in the list of values, based on the attribute level. Hence there will be no level based restrictions in the list of values.

Modifier Setup: For qualifiers, the list of values for qualifier attributes in the List Qualifiers window will show qualifier attributes with LINE, ORDER, or BOTH.

For modifiers with modifier level code=Order, the list of values for line qualifiers will show qualifier attributes with attribute levels of ORDER or BOTH.

For modifiers with modifier level code=Line or Group of Lines, the list of values for line qualifiers will show qualifier attributes with attribute levels of LINE or BOTH.

Pricing Attributes: The list of values for pricing attributes will show pricing attributes with attribute level of LINE.

Attribute Linking for a Mapped Attribute: Defining a Mapping Rule

Below are the steps for setting up an attribute that has an associated mapping rule:

Set up a new mapping rule for qualifier/pricing attribute

1. Navigate to the Pricing Transaction Entity - Attribute Linking window.
2. Find the Pricing Transaction Entity.
3. In the Context Type Field Select the Context Type from LOV.
4. Click Link Attributes to open the Link Attributes window.
5. Find the attribute to which you want to define mapping rule.
6. Click Attribute Mapping to display the Attribute Mapping window.

   Context, Attribute are display only fields.
7. Select request type.
8. Enter information in Header Level or Line Level region, or both the regions, depending on the Level of the attribute in the previous window.
9. The Global structure/view name is already displayed for that Request type chosen.
10. Select the user source type from LOV.
11. Enter a User Value String.
12. Save the record.

Attribute Mapping window

After finishing attributes setup, you can start the set up activities of creating modifiers and price lists, and attaching qualifiers or product hierarchy elements. However, the pricing engine will not know where to find/derive the values of these elements at the run time.

Hence Attributes Mapping is a mandatory step before any pricing element is used by the pricing engine. The pricing engine also provides flexibility that a transaction area can define the attributes mapping specific for its use.

For example, Customer Region can be derived from a OE_ORDER_PUB.G_HDR.sold_to_org_id for Order Management. However this same element may have to be derived from ASO_PRICING_INT.G_HEADER_REC.cust_account_id for Order Capture.

This window is used to define attribute mapping rules and is available when you click the Attribute Mapping button in the Assign Attributes window. Users cannot navigate to this window, if the Attribute Mapping Method is not Attribute Mapping. Users will be unable to navigate to this window if no Request types are defined for the Pricing Transaction Entity.

Request types region (Attribute Mapping window)

This region displays all the enabled Request types defined for the given PTE.

User needs to define attribute mapping for all the request types listed in this region.

- Application Name: This column indicates the Application that created the mapping rule. User is expected to enter the Application, before creating a mapping rule. In case user does not enter an Application name, the system defaults Oracle Pricing as the creator of the mapping rule.
- Header level region: User can enter data here if the Attribute Mapping level (defined in the Assign Attributes window) of the attribute is Order or Both. This region will be grayed out if the level is Line.
- Global structure/View Name: This is a display only field. It will show the value defined in Request type tab of the Pricing Transaction entity setup window.
- User Source type: The possible values of source type are
  - Application Profile: Profile option: Select the profile option from where you want to get the default value for this attribute. A LOV will provide a list of valid profile options such as OE: Item Validation Organization.
  - PL/SQL API: Attribute can be sourced directly from a global structure defined for the given source system. The following flexfield window will pop up. For OM all the record structures are defined in the package OE_ ORDER_PUB. For example , if you want to use payment term id as the source value for the new segment that you have defined , enter G_ LINE.payment_term_id in the function name. You have two record structures available. OE_ORDER_PUB.G_LINE contains all the possible values of a sales order line. OE_ORDER_PUB.G_HDR contains Fields from Order headers. Structure of the Line_rec_type and header_rec_type can be obtained from the Manufacturing Open Interface Manual. (For IStore/OC the equivalent global structures are defined in the package ASO_PRICING_ INT.
  - PL/SQL API Multi-Record: You can write a custom API (must be a function) that returns multiple values. The output of your function can only be a table of VARCHAR2s: for example, to get the inventory categories for

an item. The flexfield window above displays. For more information, see seeded sourcing for Item Categories or customer class as an example of multi-Value pl/sql API.

Package: Enter the PL/SQL package name.

Function: Enter the function name.

Constant Value: Constant: Enter a constant value that will always be mapped to this attribute for the given condition.

System Variable: Enter the system variable that will be mapped to the attribute for the given condition such as SYSDATE.

- Seeded Source type: It will have value if the record is seeded. This field is protected against update. If user wants to modify seeded mapping, user can enter data into corresponding user field (user source type).
- Value String (user and seeded): It is a value of the source type. Seeded value string will have seeded value. This field is protected against update. If user wants to modify seeded mapping, user can enter data into corresponding user field (user value string).
- Seeded: This is a non-navigable field indicating, whether the mapping rule is seeded or not.

Line Level region (Attribute Mapping window)

User can enter data here if the level (defined in Assign Attributes window) of the attribute is LINE or BOTH. This region will be grayed out if the Attribute Level in the previous window is ORDER.

The remaining fields are the same as defined for the Header level region.

Using Custom Sourced Attributes

Attributes can also be passed to the pricing engine directly, without the need for an attribute mapping rule. In such cases, the Attribute Manager API calls a custom API, QP_CUSTOM_SOURCE, where the user has manually defined the attributes being passed and coded the sourcing of their values.

The user code is written in the package procedure QP_CUSTOM_SOURCE.Get_ Custom_Attribute_Values. The Attribute Manager API program (Build_Contexts), calls this procedure to pick up custom-sourced attributes if the profile option QP_ CUSTOM_SOURCED is set to Y. The input parameters to QP_CUSTOM_SOURCE are Request Type code and Pricing Type. Typical values of Request Type Codes that

can be passed are ONT, ASO, OKC,IC FTE or MSD. By using the Request_Type_ Code, the user can control how the attributes are sourced based on the PTE of the calling application.

The Pricing Type can be H (Header) or L (Line) which defines the level of the attribute mapping rule. These attributes and their values are passed to the pricing engine in the same manner as the attributes sourced through attribute mapping rules.

Example

The follow example explains how a customer may code the body of QP_CUSTOM_ SOURCE for a particular case.

In this case, two segment mapping columns, 'QUALIFIER_ATTRIBUTE31' and 'PRICING_ATTRIBUTE31' that belong to contexts CUST_SOURCE_QUAL_CON and CUST_SOURCE_PRIC_CON respectively and linked to PTE 'Order Fulfillment', will have Custom Sourced values as 10 for ORDER as well as LINE Attribute Mapping levels. The user must ensure that the Attribute Mapping method for both these PTE-Attribute links is CUSTOM SOURCED in the 'Attribute Linking and Mapping' setup window.

```
CREATE or REPLACE PACKAGE body QP_CUSTOM_SOURCE AS
/*Customizable Public Procedure*/
PROCEDURE Get_Custom_Attribute_Values
(   p_req_type_code      IN VARCHAR2
    ,p_pricing_type_code IN VARCHAR2
    ,x_qual_ctxts_result_tbl  OUT QP_ATTR_MAPPING_PUB.CONTEXTS_RESULT_TBL_TYPE
    ,x_price_ctxts_result_tbl OUT QP_ATTR_MAPPING_PUB.CONTEXTS_RESULT_TBL_TYPE
)   is
Begin
    Ifp_req_type_code='ONT'andp_pricing_type_codein('L','H')then
        x_qual_ctxts_result_tbl(1).context_name := 'CUST_SOURCE_QUAL_CON';
        x_qual_ctxts_result_tbl(1).attribute_name := 'QUALIFIER_ATTRIBUTE31';
        x_qual_ctxts_result_tbl(1).attribute_value := '10';
        x_price_ctxts_result_tbl(1).context_name := 'CUST_SOURCE_PRIC_CON';
        x_price_ctxts_result_tbl(1).attribute_name := 'PRICING_ATTRIBUTE31';
        x_price_ctxts_result_tbl(1).attribute_value := '10';
    end if;
end Get_Custom_Attribute_Values;
END QP_CUSTOM_SOURCE;
/
```

Building Attribute Mapping Rules

The program Build Attribute Mapping Rules dynamically generates the package QP_BUILD_SOURCING_PVT, which contains attribute mapping calls to build attribute mapping rules for Attributes. The package QP_BUILD_SOURCING_PVT is generated when the concurrent program is run. This package contains only the rules of used attributes (Qualifiers/Pricing Attributes used in any pricing setup) that can be mapped at run-time. This means anytime you use a new attribute (one that has not been used by any other modifier, price list, formula), you will need to run the program Build Attribute Mapping Rules accessible in the Oracle Pricing Manager Responsibility or from the Tools Menu of the Attribute Linking and Mapping windows. Also anytime you change any attribute mapping rules you will need to run this program.

The. program generates the attribute mapping rules for all the attributes that have the Attribute Mapping Enabled box selected. When the Build Attribute Mapping Rules program runs successfully, the Attribute Mapping Status is checked for those attributes for which an attribute mapping rule is generated.

The Build Attribute Mapping Rules reports back success or failure to the user. If the Attribute Mapping Status box changes to selected status for the attributes that were successfully mapped then it was successful. The window automatically re-queries the database if the Build Attribute Mapping Rules process is successful and moves focus to the Attribute Mapping Status box.

You will not have to run the Build Attribute Mapping rules program for User entered or custom sourced attributes.

This task flow should be used if an attribute has Attribute Mapping Enabled selected and Attribute Mapping Method is attribute mapping and the Attribute Mapping Status box is cleared.

This task flow helps by doing the following:

1. Generates a source code for a new qualifier/pricing attribute for which attribute mapping is defined and generates a source code for attributes for which attribute mapping rules are modified. Since Build_context program changes the status of database objects, it is recommended this program be run during off peak hours.

Make sure this attribute has been used in a valid pricing setup such as modifiers or price list. Set up a modifier and attach the newly created qualifier/pricing attribute.

For example, for this qualifier to be mapped by OM, you need to regenerate the attribute mapping package by running the program Build Attribute Mapping Rules

2. Check if the program has completed with a success

   This program needs to be run when a qualifier is used for the first time to set up a modifier or a price list. For example , if you are using Customer Class as a qualifier for the first time in your install , you need to run this program.

3. Navigate to the Pricing Transaction Entity - Attribute Linking window.

4. Find the Pricing Transaction Entity.

5. In the Context Type Field Select the Context Type from LOV.

6. Navigate to Tools and select Build Attribute Mapping Rules to run the Build_ context api. which will generate a mapping rule code only for those attributes that have Attribute Mapping Enabled box selected.

7. Check if the program has completed successfully. If it has, the Attribute Mapping Status box is selected for the attributes for which a mapping rule code is generated.

   Note: When the concurrent program Build Attribute Mapping Rules is run, the Used In Setup box is updated to reflect the status of the attribute usage. When the attribute is removed, the Used In Setup box will remain selected.

8. Enter the order

9. Verify through the Pricing Engine Request Viewer that the attribute has been correctly sent to the pricing engine. You can also verify if the pricing engine used the correct attributes while pricing. For more information, see: *Pricing Engine Request Viewer*.

   Note: Before running Build Attribute Mapping Rules program, select the profile QP: Build Attributes Mapping Options. When set to Yes, mapping rules can be generated for attributes used in active pricing setup.

   When set to No, mapping rules are generated for all the attributes. The Build Attribute Mapping Rules program will map the attributes that are used only in the active pricing setup if the profile value QP: Check For Active Flag is set to Yes. If the profile value QP: Check For Active Flag is set to No, this program will map the attributes that are used in both the active and inactive setup.

Restore Seeded Data Using the Restore Defaults button

You can restore the default seeded settings in the following windows:

- Context Setup
- Link Attributes

Context Setup

For *contexts*, the Restore Default button restores any changed values to the original seeded values of a context. So if a seeded context has been changed, the values can be restored to their original condition. The Name and Description fields display the seeded values rather than the user entered values.

In case of seeded context , the Name and Description fields will show their seeded values. However if these seeded values are changed, these fields show the user entered values and the seeded values will be preserved in seeded name, seeded description fields respectively. These seeded fields will always be hidden.

For *attributes*, the Restore Default button restores any changed values to the original seeded values of an attribute. So if a seeded attribute has been changed, the values can be restored to their original condition. Precedence, name, valueset, datatype window fields display the seeded values in place of user entered values.

For seeded attributes, the precedence, name, valueset, and datatype fields will show their seeded values. However, if user changes the seeded values, these fields will show the user entered values and the seeded values will be preserved in separate fields named as seeded precedence, seeded name, seeded valueset , seeded datatype. These seeded fields will always be hidden.

Note: Please note that this button will replace values in all the fields with seeded values.

Link Attributes

For seeded attribute, the Attribute Mapping Method column will have its seeded value. However if the user changes the attribute mapping method, this field will show the user entered value and the seeded value will be preserved in a separate field named as seeded Attribute Mapping Method. These seeded Attribute Mapping Method field will always be hidden.

The Restore Default button restores the original seeded attribute mapping method of an attribute. If the user overwrites a seeded attribute and wishes to go back to seeded values again later, user can do so by clicking this button.

This button will be grayed out for non-seeded attributes. This button will continue to remain gray for seeded attributes, if the user-entered Attribute Mapping method remains same as seeded Attribute Mapping method.

Note: The attribute Total Item Quantity is only seeded for Oracle Transportation (FTE). However, you can manually link an attribute to the Order Fulfillment Pricing Transaction Entity, and create a corresponding sourcing rule for attributes such as Total Item Quantity.

Attribute Mapping

For all seeded Attribute Mapping rules, the Restore Defaults button restores the seeded Source type and the seeded Value string as User source type and User value string respectively. If the seeded Source type and the seeded Value string are same as user Source type and user Value string, the Restore Defaults button will be grayed out.

Troubleshooting While Setting Up Attributes

Scenario 1

This warning message will be displayed on setup windows when user tries to use in the setup, any newly added attribute having Attribute Mapping Method = ATTRIBUTE MAPPING whose Attribute Mapping Enabled flag is not Y.

Message: This Attribute (with Context: &CONTEXT and Attribute: &ATTRIBUTE) will not be mapped since it is not Attribute Mapping Enabled.

Scenario 2

This warning message will be displayed on setup windows when user tries to use in the setup, any newly added attribute having Attribute Mapping Method =ATTRIBUTE MAPPING whose Attribute Mapping Status flag is not Y.

Message: Warning: This Attribute (with Context: &CONTEXT and Attribute:&ATTRIBUTE) has not been mapped yet. Please Build Attribute Mapping Rules.

Scenario 3

The Attribute Mapping level for this Attribute was defined as BOTH. In such cases, the user is expected to enter an Attribute Mapping rule, one each for the Header and line level. If the user enters only one level, pricing inconsistencies will occur.

Message: This attribute must have an attribute mapping rule both at Header as well as Line levels.

Scenario 4

For a given PTE, the user must enter Attribute Mapping rules for all the request types that belong to that PTE. In case user does not enter for all of them, different Request types for the same PTE may not be able to fetch the same price.

Message: You must map this Attribute for all the Request types.

Checklist for Building Attribute Mapping Rules

Sometimes users believe they have successfully mapped an attribute when they get the message Attribute Mapping Rule Generation is Successful, but find that the new Attribute Mapping box for that attribute remains cleared when it should be selected. This may occur when they create a new attribute, link it to a Pricing Transaction Entity successfully and then map it using the Build Attribute Mapping Rules from the Tools menu.

To ensure a successful mapping and to avoid the situation described above, users must ensure the following conditions are met:

- The Attribute Mapping Enabled box must be selected.
- The Attribute Mapping Method must be ATTRIBUTE MAPPING. Attributes with mapping method USER ENTERED and CUSTOM SOURCED are never meant to be created by the Build Attribute Mapping Rules program.
- If the attribute is newly created, you must ensure that it is attached to at least one valid Pricing Setup such as a Modifier, Price list or Limit.
- Ensure that the PTE-Attribute link that was created has at least one attribute mapping rule.

Troubleshooting in Pricing Setup windows related to Attribute Management

Scenario 1

User tries to attach a combination of qualifier with the attributes that have Attribute Level of LINE and ORDER within the same Qualifier Grouping No of a Qualifier Rule on the Qualifier Rules window.

Message: Qualifier Attribute with an attribute level of LINE cannot be present in combination with a Qualifier Attribute with attribute level of ORDER, within a Qualifier Grouping No of a Qualifier Rule.

Scenario 2

This error message will be displayed on the Qualifiers tab of the Modifiers window when a user tries to attach a combination of qualifier attributes that have attribute level of LINE and ORDER within the same Qualifier Grouping No.

Message: There is a mix of LINE and ORDER qualifier attribute level for list line id &LIST_LINE_ID and qualifier grouping no &QUALIFIER_GRP_NO. Ensure that all the qualifier attributes should be either of level LINE/BOTH or ORDER/BOTH for a given list line id and qualifier grouping no.

Troubleshooting during Pricing Setup

While defining Qualifier or Pricing Attributes in the Pricing Setup windows, only those Attributes with LOV Enabled box selected display in the LOV's on these windows. If basic pricing is installed, only those attributes that have been set up with the AVAILABLE_IN_BASIC box selected display in the LOV's on the various Pricing Components setup windows. However, all attributes that are setup are available to Advanced Pricing users, subject to meeting the other attribute level conditions.

It is important to set the profile QP: Pricing Transaction Entity to the right value before creating or querying any setup data in the pricing application. It is not recommended to change this profile often as you will see the other context-attributes combinations for a different PTE when querying on the pricing setup windows. If this happens, you will see the internal ID code.

Troubleshooting During Integration or Runtime

Viewing Attributes

Question 1: Why do I not see the Pricing Contexts and Attributes defined by me in the Pricing setup windows?

Answer 1: The following conditions must be met;

- The contexts and attributes defined by you using the Attributes Manager windows are assigned to the right Pricing Transaction Entity Code.
- The contexts and attributes are enabled.
- System Profile Option QP: Pricing Transaction Entity Code points to the correct Pricing Transaction Entity Code.
- The Attribute levels are correct. Only Attributes with certain levels may be visible in certain windows.

Question 2: What happens if I do not set the Profile QP: Pricing Transaction Entity correctly?

Answer 2: This profile indicates the current Pricing Transaction Entity in use. Only those contexts and attributes assigned to the current Pricing Transaction Entity will be available in the LOV's on the setup windows.

Querying up setup data displays the description to be shown only for those contexts and attributes that are assigned to the current Pricing Transaction Entity.

Build Attribute Mapping Rules

Question 3: While in Attribute Linking and Mapping window, I run Build Attribute Mapping rules from the Tools bar. The return message confirms that the Build Attribute Mapping program ran successfully. But the Attribute Mapping status box of the attribute that I just linked is still unchecked.

Answer 3: You may not have completed one or more of the following pre-requisites:

- The Attribute Mapping Enabled button was not checked.
- The Attribute Mapping Method was not ATTRIBUTE MAPPING. Attribute Mapping Methods other than ATTRIBUTE MAPPING do no need to be

mapped. If the attribute that you linked was a new one, it must be used in at least one valid pricing setup.

Question 4: The concurrent program Build Attribute Mapping Rules failed with error.

Answer 4: Execute the following statement and examine the output:

a. Select text from dba_errors where name = QP_BUILD_SOURCING_PVT_TMP.

b. Check which attribute mapping API is causing this error.

Question 5: The following error occurs while running Build Attribute Mapping Rules concurrent program: ORA-04021: timeout occurred while waiting.

Answer 5: This occurs when someone makes a pricing call while the concurrent program is running. Do not run the Build Attribute Mapping Rules concurrent program when active users are calling pricing engine.

Attribute Manger Upgrade Business rules

Question 6: In my Attribute Manager setup, I cannot see one or more of the seeded attributes described in the *Oracle Advanced Pricing User's Guide* after the Upgrade program or the Loader Upload program ran successfully.

Answer 6: When the Attribute Manager Loader or the Attribute Manager Upgrade program finds an existing attribute that uses the same segment mapping column as a seeded attribute (in the case of Attribute Manager Loader program) or a user attribute created in the flexfield (in the case of Attribute Manager Upgrade program), it is resolved as follows:

- Loader program: New seeded attributes, whose mapping columns have already been used by a different attribute belonging to the same context, will not get uploaded by the Loader. Existing seeded attributes' attributes will not get updated if the mapping column to be updated was different and used by a different attribute.
- Upgrade program: Existing attributes in the flexfields whose mapping columns have already been used by a different attribute for the same context in the new Attribute Manager setup will always get upgraded by the Upgrade Program. The attribute that did not show up in the new Attribute Manager setup will most likely be a new seeded attribute from a loader upload. This different

attribute with the same mapping column, which was already present in the new Attribute Manager setup, will be deleted from the system and the user will be requested to enter this attribute manually with a different mapping column.

In either case, a line displays in the QP_UPGRADE_ERRORS table to advise you to manually add or update the attribute that was deleted or not updated. See the *Oracle Advanced Pricing User's Guide* for more information on adding or updating the attribute, its PTE-links and the Attribute Mapping rules (wherever necessary) using a different, unused segment mapping column.

Note: This problem may arise if you created your own qualifier attributes and mapped them to QUALIFIER_ATTRIBUTE1 to QUALIFIER_ATTRIBUTE30, not realizing that these mapping columns are reserved for seeded attributes. See *Oracle Advanced Pricing User's Guide* for more information on creating context and attributes.

Entering Orders

Question 7: While entering the order line in sales order PAD receives the following error:

```
FND_AS_UNEXPECTED_ERROR (PKG_NAME=oe_order_adj_pvt) (PROCEDURE_NAME=oe_line_
adj.calculate_adjustments) (ERROR_TEXT=calculate_adjustments#130 ORA-06508:
PL/SQL: could not find program unit being called).
```

Answer 7: Check dba_errors for this package in or to determine which attribute mapping API is causing the error. If this is a custom API then correct the API. If this is the seeded API then determine whether a correction patch is available.

Question 8: The context of an attribute that was successfully mapped did not show up in the Order management Sales Pad Pricing Context list of values.

Answer 8: One of the following solutions may resolve the issue:

- The only contexts that will show up in the OM Sales Pad window are the ones that have at least one attribute that is USER ENTERED. If the context has all its attributes as mapping method ATTRIBUTE MAPPING, this context will not show up Pricing Context List of values.
- You must create all new attributes using the new Attribute Manager Context and Attributes window. Using this method, all attributes of type Pricing

Attribute will get created in the OM Flexfield and the flexfield will get registered (if required); its definitions will freeze and then get compiled automatically. Remember, the converse is not true. An attribute created using the Flexfield windows will not get created in the Attribute Manager tables. Columns updated in Flexfield window is also not supported in Pricing.

- If the attribute that you just created was of the type Pricing Attribute, the system creates the same attribute in the OM flexfield tables and then compiles the flexfield. Check the Concurrent manager requests to see if the request was completed as Normal.
- Attributes having Column Mapped values PRICING_ATTRIBUTE31..100 must get registered. Although this is done automatically by the system, it is advised that you confirm that this has occurred.

Problems and Solutions

Table 10–6 Problems and solutions related to attribute management

Probable Cause	How to Debug
QP_Attr_Mapping_PUB.Build_Contexts package is invalid due to incorrect mapping of data attributes mapping.	Check dba_errors for this package in or to determine which attribute mapping API is causing the error. If this is a custom API then correct the API. If this is the seeded API then determine whether a correction patch is available.
Concurrent Program Build Attribute Mapping Rules failed with error.	Execute following statement and examine the output: select text from dba_errors where name =QP_BUILD_ SOURCING_PVT. Verify that custom sourcing causes this error.
Getting error while running Build Attribute Mapping Rules concurrent program ORA-06502: PL/SQL: numeric or value error: character string buffer too small ORA-06512: at APPS.QP_ATTR_ MAPPING_PUB, line 1445 ORA-20000: ORA-04021: timeout occurred while waiting.	This occurs when someone makes a pricing call while the concurrent program is running. Do not run the Build Attribute Mapping Rules concurrent program when active users are calling pricing engine.
While entering the order line in sales order PAD receives an error: END_AS_UNEXPECTED_ ERROR (PKG_NAME=oe_order_adj_pvt) (PROCEDURE_NAME=oe_line_adj.calculate_ adjustments) (ERROR_TEXT=calculate_ adjustments#130 ORA-06508: PL/SQL: could not find program unit being called).	Execute following statement and examine the output: select text from dba_errors where name = QP_BUILD_ SOURCING_PVT; Determine whether any custom sourcing causes the errors. If the seeded sourcing rule causes this error, determine whether there is a patch available to correct the seeded rule. If the error is "Encountered the symbol _ when expecting…" then determine the relevant patch to be applied.

Upgrading Considerations

When upgrading from releases before 11.5.7/Patch G, the upgrade program will do the following:

1. Upgrade Contexts and Attributes
2. Upgrade source systems and request types
3. Create Pricing Transaction Entities and attribute links
4. Upgrade attribute mapping rules
5. Assign Pricing Transaction Entities to existing modifiers

Table 10–7 Upgrade considerations

Pre H Release	Post H Release
Request Types and Source Systems common for all applications	Request Types and Source Systems grouped by Pricing Transaction Entity
Context and Attributes created and maintained in Flexfields	Context and Attributes created and maintained in QP tables
All attributes available to all applications for Pricing Setup	Attributes available to applications based on PTE. Attributes are now LOV Enabled and Limits Enabled
Attributes have one sourcing rule used by all request types	Attributes may have one or more attribute mapping rules for different request types
Only one ordering structure i.e. Order Fulfillment.	At least 4 seeded PTEs, with provision for expanding.

Table Upgrade

Table 10–8 Table upgrade

Contexts	fnd_descr_flex_contexts	qp_prc_contexts_b
Context	fnd_descr_flex_contexts_tl	qp_prc_contexts_tl
Attributes	fnd_descr_flex_column_usages	qp_segments_b
Attributes	fnd_descr_flex_col_usage_tl	qp_segments_tl
Source System and Request Types	qp_price_req_sources	qp_pte_source_systems
Source System and Request Types	qp_price_req_sources	qp_pte_request_types_b/tl

Table 10–8 Table upgrade

Contexts	fnd_descr_flex_contexts	qp_prc_contexts_b
Attribute Linking	oe_def_attr_condns	qp_pte_segments
Attribute Linking	ak_object_attributes	qp_pte_segments
Attribute Linking	oe_def_condn_elems	qp_pte_segments
Attribute Mapping Rules	oOe_def_attr_def_rules	qp_attribute_sourcing

Upgrading Context and Attributes

This section consists of upgrading Context and Attributes from the current Flexfield Structure to a set of new normalized QP tables.

Upgrading Contexts

In the current system, all the contexts are stored in qualifier contexts and pricing contexts flexfields. The upgrade program will copy all the contexts from these flexfields to the new QP tables. All the Qualifier Contexts flexfield qualifiers will be copied as contexts as qualifier contexts.

All the Pricing Attribute flexfield qualifiers except ITEM will be copied as pricing attribute contexts. ITEM Qualifier will be copied as Product context. All contexts created in the flexfields by user 1 will be treated as seeded data. All other information, like name and description in various languages, enabled flag are available in the flexfields.

Upgrading Attributes

For every context that is created in the new system from the flexfields, attributes are available in the flexfield usage tables. These attributes will be copied under the corresponding context in the new system. Some attributes in the flexfield usage tables do not have Value sets, as some attributes use Key flexfields for their valid values. All other information like names, mapping columns, precedence are available in flexfield usage tables.

Upgrading Source System and Request Types

Currently, all the Source Systems and Request Types are mapped as many-to-many relationships:

Table 10–9 Request types and source systems

Request Types	Source Systems
ONT	ASO
ASO	ASO
ASO	QP
IC	INV
IC	QP
OKC	OKC
OKC	QP

More seeded Request types are provided to include additional relationships between Request Types. The Request Types and Source Systems included in the seeded data for Attribute Management is outlined in the following table:

Table 10–10 Request Types and Source Systems included in the seeded data for Attribute Management

Request Types	Source Systems
ONT (Order Management Order)	ASO (Oracle Order Capture)
ONT	AMS (Oracle Marketing)
ONT	QP (Oracle Pricing)
ASO (Order Capture)	ASO
ASO	QP
IC (Inter Company Invoicing)	INV (Oracle Inventory)
IC	QP
OKC (Oracle Contracts Core)	OKC (Oracle Contracts Core)
OKC	QP
MSD (Demand Planning)	QP
FTE (Logistics Exchange Load Shipment)	FTE (Oracle Transportation)

Attribute Management has a new entity called Pricing Transaction Entity which can be described as an Ordering Structure that has associated Request Types and Source Systems. When upgrading, a new set of Pricing Transaction Entities are created in the target system. The following table displays the Pricing Transaction Entities that will always be created, as a part of upgrade. More Pricing Transaction Entities may be created depending on the Customer's own Request Types and Source Systems, different from the seeded data.

New Transaction Entities

Table 10–11 New transaction entities

Code	Name
ORDFUL	Order Fulfillment
INTCOM	Intercompany Transaction
LOGSTX	Logistics
DEMAND	Demand Planning

Mapping of Seeded Request Types and Source Systems

Secondly, there is a pre-determined process to associate Request Types and Source Systems to these Pricing Transaction Entities. These Request Types with their associations will act as seeded data. Please refer to the earlier section on Seeded Pricing Transaction Entity, Source Systems, and Request Types.

Whenever a Request type is created in the target system by the Upgrade program, the default Global Structure associated with them will be as shown below:

Table 10–12 Default global structures associated with request types

Request Types	Global Structure Name: Header Level	Global Structure Name: Line Level
ASO	ASO_PRICING_INT.G_HEADER_REC	ASO_PRICING_INT.G_LINE_REC
OKC	OKC_PRICE_PUB.G_CONTRACT_INFO	OKC_PRICE_PUB.G_CONTRACT_INFO
IC	INV_IC_ORDER_PUB.G_HDR	INV_IC_ORDER_PUB.G_LINE
ONT	OE_ORDER_PUB.G_HDR	OE_ORDER_PUB.G_LINE
FTE	None	None
MSD	None	None

Creating PTE and Attribute links

Currently, all the attributes that are available in the existing flexfield may already be attribute mapped or otherwise. In the new data model, each attribute must be linked to one or more PTEs, before it can have Attribute Mapping rules:

1) Attributes that are Attribute Mapping enabled or already mapped

Such attributes will be associated with a Source System in the current System as per defaulting rules. At this point, the Upgrade program has already created all the PTE-Request Types-Source Systems associations (with data) in the target system. The Upgrade program will find out the PTE(s), that this Source System is associated with and will create as many PTE(s) and attribute links in the new PTE-Attribute mapping table.

As an example, say Segment-1 is associated with QP Source System, as per the Defaulting Condition Templates. The upgrade program will create 3 links for this attribute, represented as 3 rows in the new PTE-Attribute mapping table as shown below.

Table 10–13 PTE Attribute Mapping Table

Pricing Transaction Entity	Attribute
1 Unassigned-2	Segment-1
2 Order Fulfillment	Segment-1
3 Intercompany Transaction	Segment-1

The three rows get created since Source System QP is attached to PTEs Unseeded-2, Order Fulfillment and Intercompany Transaction.

2) Attributes that are not Attribute mapped

Some attributes may not have a Attribute Mapping rule. These are the attributes for which there are no transactions in Price Lists, Modifiers, and Formulas. For such attributes, they will be linked to all the PTEs that we created so far (including the seeded ones) in the PTE Attribute Mapping Table.

Upgrade Attribute Mapping Rules

When upgrading, all the attributes that have default Attribute Mapping rules defined in the current system will have the same Attribute Mapping rules redefined in the new model. Every attribute present in the PTE Attribute Mapping table may have an Order level mapping, a Line level mapping or both, depending on how the attributes were defined in the current system: for example, Header only, Line only or Header and Line.

Note: If the Attribute Manager Loader or the Attribute Manager Upgrade program encounters an existing attribute in one of the following situations, it will skip the upload/upgrade of that attribute:

- At customer location that uses the same segment mapping column as used by a seeded attribute (in case of Attribute manager Loader program) or
- A user attribute created in the flexfield (in case of Attribute Manager Upgrade program).

However, unlike the current system, every attribute that will have a Attribute Mapping rule will ideally have as many sets of Attribute Mapping rules as the number of Request types that are associated with the PTE for that attribute in the PTE Attribute Mapping table. At this point, the upgrade program may not create all the Attribute Mapping rules for all the Request types. Depending on the attribute, given the application that created it, the Upgrade program will create Attribute Mapping rules for the Request Types as shown in the following table:

Table 10–14 Request Types that will be Attribute Mapped

Source Systems	Request Types
QP	ONT
OKC	OKC
ASO	ASO
AMS	ONT, ASO

Only the seeded PTE-Attribute combinations from the PTE Attribute Mapping Table that were created for use in basic pricing may have default Attribute

Mapping rules. Also, all the attributes that have defaulting rules and belong to Source System FTE, will not have Attribute Mapping rules.

Assigning PTE to existing Modifiers

With attribute mapping, all Modifier lists, Price lists, Formulas, will be identified by the Pricing Transaction Entity that created them (in addition to the Source System, which was used so far). In the new system, a new column PTE_CODE has been added to QP_LIST_HEADERS_B table, which will store the Pricing Transaction Entity. The Upgrade program will update the PTE_CODE for every modifier list/price list/formula based on the following logic.

At this point, the upgrade program has already created all the PTE-Request Types-Source Systems associations (with data) in the target system. The Upgrade program will find out the PTE(s), for every Modifier, from its Source System. If the Source System is associated to one PTE only, the PTE_CODE will be updated with that PTE. If the Source System is associated with more than one PTE, Order Fulfillment will be chosen over other PTEs.

Sample Code

```
-- Please note, the cursor cs_item_cost may need to be modified
-- before use in your environment.


CREATE OR REPLACE
PACKAGE MY_CUSTOM_SOURCING AUTHID CURRENT_USER AS
-- please see package body for version, history and notes.

-- Package globals...
G_Organization_id NUMBER := FND_PROFILE.VALUE('QP_ORGANIZATION_ID');

-- Cached in and out values for each sourcing routine.

TYPE Item_Info_Rec_Type IS RECORD
( inventory_item_id  VARCHAR2(240)
, item_cost NUMBER
);


G_Item_Info   Item_Info_Rec_Type;

FUNCTION Get_Item_Cost (p_item_id IN NUMBER) RETURN VARCHAR2;
```

```
PRAGMA RESTRICT_REFERENCES (Get_Item_Cost,WNDS);

END MY_CUSTOM_SOURCING;
/
-------------------------------------------------

CREATE OR REPLACE
PACKAGE BODY MY_CUSTOM_SOURCING AS

/* Package Body version 1.01 - 30th January 2001 */

-- This is the package body for Simon's example of
-- Advanced pricing 11i's custom sourcing routines used for
-- retrieving additional information from the apps tables into pricing
-- data structures.
-- Additions should only be created where a solution using
-- standard functionality is not feasible.
-------------------------------------------------------------
-- To marginally improve performance on successive pricing calls,
-- we cache the most recent details from the sourcing functions
-- so that the cursors are not run every time if the requests
-- are the same. These cached values are stored in package-wide
-- variables.
-------------------------------------------------------------

FUNCTION Get_Item_Cost (p_item_id IN NUMBER) RETURN VARCHAR2 IS
-- Function to retrieve an item cost from cst_item_costs
-- note, this returns null is a cost is not found
-- so that the calling app can handle this.

-- Note, Use your own cost type id from cst_cost_types.
-- I've used 1 just for testing.

CURSOR cs_item_cost(cp_item_id IN NUMBER) IS
  SELECT    cic.item_cost
    FROM    cst_item_costs cic
    WHERE   cic.inventory_item_id = cp_item_id
    AND     cic.cost_type_id      = 1
    AND     cic.organization_id   = G_organization_id;

 v_cost cst_item_costs.item_cost%TYPE := NULL;

BEGIN
  IF p_item_id = G_Item_Info.inventory_item_id THEN
```

```
  -- if the requested item is already cached then do nothing yet.
  NULL;
ELSE
```

11

Get Custom Price

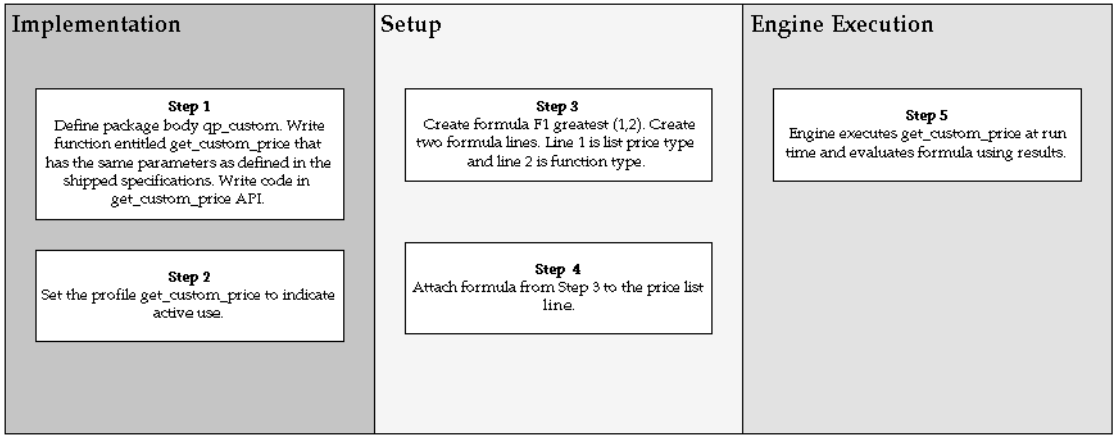
This chapter discusses implementation considerations for get_custom_price. The following topics are covered:

Overview of Get Custom Price Implementation

The actual value of a modifier or price is accomplished using get custom price. An example of get custom price is: my price is 5% less than the competitors price (your price is 95% of competitor's price), where the competitor's price is drawn from another custom database or a PL/SQL call.

An overview of the tasks involved in the implementation of get_custom_price is depicted in the following image.

Figure 11–1 Overview of get_custom_price implementation steps



Implementing Get_Custom_Price

Step 1: Write extension code in qp_custom.get_custom_price

If you use get_custom_price, then you must create the package body for qp_custom and create a function get_custom_price.

The pricing engine calls the API qp_custom.get_custom_price with the following set of parameters:

- P_price_formula_id: IN NUMBER
- P_list_price: IN NUMBER
- P_price_effective_date: IN DATE

- P_req_line_attrs_tbl: IN QP_FORMULA_PRICE_CALC_PVT.REQ_LINE_ATTRS_TBL)

The formula ID identifies the formula from which the API is called. P_req_line_attrs_tbl provides information such as product, pricing attributes and qualifiers. P_req_line_attrs_tbl contains the following fields:

- Line_index: Line index of the price request line
- Attribute_type: Qualifier, product, pricing
- Context: Context name (for example, Item)
- Attribute: Attribute name (for example, Pricing_attribute1)
- Value_from: Attribute value (for example, 149)

A DML (Insert/Update/Delete) operation is not supported in the get_custom_price routine. The pricing engine does not guarantee upgrade possibility if any engine variables are referred in the get_custom_price function.

For example:

```
CREATE or REPLACE PACKAGE BODY QP_CUSTOM AS
FUNCTION Get_Custom_Price (p_price_formula_id  IN NUMBER,
       p_list_price   IN NUMBER,
       p_price_effective_date IN DATE,
 p_req_line_attrs_tbl IN QP_FORMULA_PRICE_CALC_PVT.REQ_LINE_ATTRS_TBL)
RETURN NUMBER
is
BEGIN
if p_price_formula_id = 7538 then
 return 14.01;
end if;
end get_custom_price;
END QP_CUSTOM;
```

Step 2: Set the value of profile QP: Get Custom Price Customized

The engine calls the get_custom_price function only if profile QP: Get Custom Price Customized is set to Y. If you set up a formula but not a profile, a runtime error displays. Set the profile at site level.

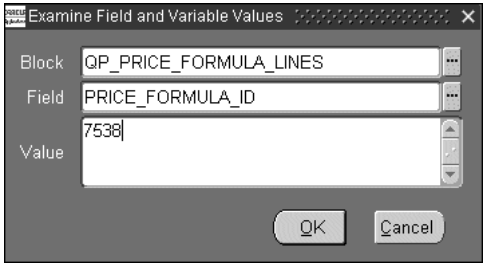
Step 3: Create a formula to use the get_custom_price function

The follow image depicts the Pricing Formulas window:

Figure 11–2 Pricing Formulas window

Remember to note the formula_id by using Help, Examine. Use this formula ID in the get_custom_price function to identify the formula. The following image depicts the Examine Field and Variable Values window that pops up from the Pricing Formulas window in Oracle Advanced Pricing.

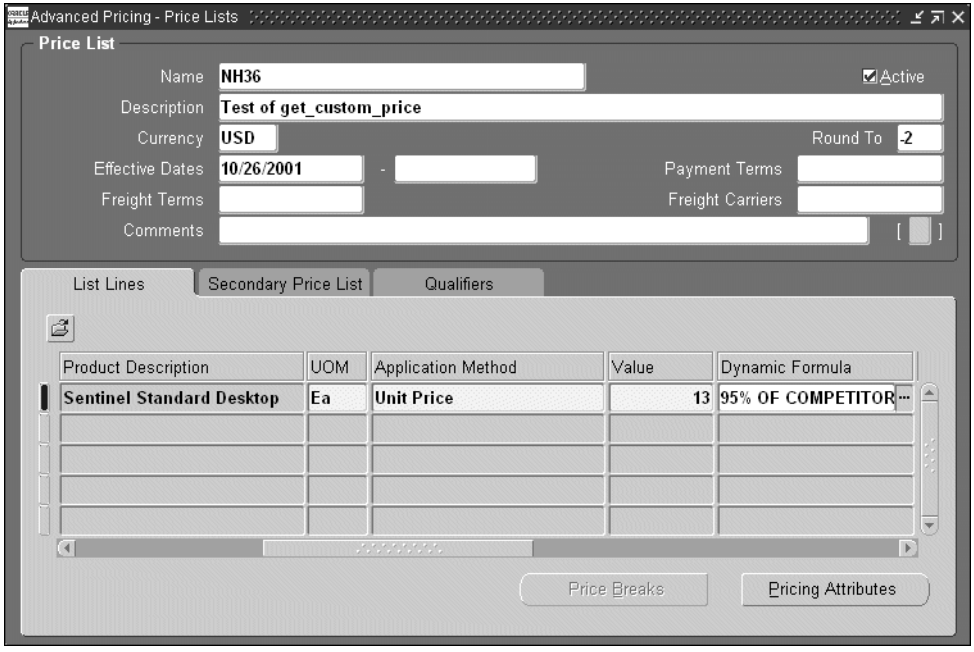
Figure 11–3 Examine Field and Variable Values window



Step 4: Attach this formula to the price list line.

The following image depicts the Price Lists window in Oracle Advanced Pricing.

Figure 11–4 Price Lists window



Get_Custom_Price_Customized

The following is a sample code that depicts how the body of the Get_Custom_Price function is coded in the file QPXCUSTB.pls. The user must use the function specification of Get_Custom_Price as well as any type definitions from QPXCUSTS.pls.

The parameters to Get_Custom_Price are always fixed and cannot be customized. The user can use the input parameters passed by the pricing engine in their custom code. The function returns a number. The user can code the function to return the desired value which must be a number. The return value is used in the evaluation of the formula.

For example, consider a formula with the expression 1*2 where 1 and 2 are step-numbers. Each step-number corresponds to a formula line. Each formula line has a type.

Step 1 corresponds to a formula line of type numeric constant, with a component of 200, and Step 2 corresponds to a formula line of type function. The value returned by the QP_CUSTOM.Get_Custom_Price function is used as the value for this step.

To evaluate the formula, the pricing engine first obtains the value of each step and substitutes the step with its value in the expression. Step 1 is substituted by the value which is 200. Step 2 is substituted with the value returned by Get_Custom_Price which must be customized by the user (the profile option mentioned previously must also be set to Yes to use this Get_Custom_Price functionality).

If Get_Custom_Price is customized as:

Package Body Qp_custom

The Get_Custom_Price function name and parameters cannot be customized but the body can be been customized. The parameters are:

1. p_price_formula_id: Primary key of formula that uses Get_Custom_Price function.
2. p_list_price: List price of the price list line to which the formula using Get_Custom_Price is attached. May have null value.
3. p_price_effective_date: Date of formula evaluation by the pricing engine.
4. p_req_line_attrs_tbl: A PL/SQL table of records containing context, attribute, attribute value records for product and pricing attributes and a column indicating type (product attribute or pricing attribute). The engine passes the pricing attributes and product attributes of the current line to which the formula is attached.

The parameters are passed to the function by the pricing engine and can be used in the function body.

```
FUNCTION Get_Custom_Price (p_price_formula_id IN NUMBER,
    p_list_price IN NUMBER,
    p_price_effective_date IN DATE,
    p_req_line_attrs_tbl IN QP_FORMULA_PRICE_CALC_PVT.REQ_LINE_ATTRS_TBL)
RETURN NUMBER IS
v_requested_item VARCHAR2(240);
v_weight NUMBER;
BEGIN
    IF
    p_price_formula_id = 1726 -- Assume this is the internal Id/primary key for
    the sample Formula 1*2
    THEN
        Loop through the PL/SQL table of records passed by the Engine as an
        input parameter and containing Pricing Attributes and Product Attributes
        of the Price List Line or Modifier Line to which the current formula is
        attached.
FOR i IN 1.p_req_line_attrs_tbl.count LOOP
    IF p_req_line_attrs_tbl(i).attribute_type = PRODUCT
    AND
    p_req_line_attrs_tbl(i).context = ITEM
    AND
    p_req_line_attrs_tbl(i).attribute = PRICING_ATTRIBUTE1
THEN
    For this combination of Product Context and Attribute, the Attribute Value
    is the Inventory Item Id v_requested_item:= p_req_line_attrs_tbl(i).value;
 END IF;
    IF p_req_line_attrs_tbl(i).attribute_type = PRICING
    AND
    p_req_line_attrs_tbl(i).context = MIXED
    AND
    p_req_line_attrs_tbl(i).attribute = PRICING_ATTRIBUTE4
 THEN
    For this combination of Pricing Context and Attribute, let's say, the
    Attribute Value is the Weight of the item to which the formula is attached.
    v_weight:= p_req_line_attrs_tbl(i).value;
    END IF;
END LOOP; For Loop
RETURN v_weight;
    EXCEPTION
        WHEN OTHERS THEN
            RETURN NULL;
END Get_Custom_Price;
```

```
END QP_CUSTOM;
```

If v_weight has a value 1.2 then Get_Custom_Price returns a value of 1.2. The pricing engine evaluates the formula as 200*1.2 = 240.

12

Events and Phases

This chapter discusses implementation considerations for phases and events. The following topics are covered:

Overview

Pricing events and phases enable you to configure Oracle Advanced Pricing so transaction pricing occurs when it is required by the application process flow. Pricing events and phases also enable you to define which pricing data is considered for application to a request at the pricing point in your transaction process flow. You can separate the pricing of your transaction, rather than pricing a whole transaction at once. Events and phases allow for the implementation of the following types of pricing business rules:

- Freight and special charges are calculated at time of shipping.
- Cross order volume discounts apply at end-of-day (once total order volumes have been derived) in a high volume batch environment.
- Coupons are awarded only after all items in the shopping cart are priced and the user proceeds to final checkout.

Note: The Price a Logistics Load event is only used by the Oracle Transportation application for pricing their transactions.

What are Pricing Events?

A pricing event is a point in the transaction life cycle when you want to price the transaction (or certain transaction lines), or when you want to apply price adjustments, benefits, or charges to the whole transaction or specific transaction lines.

Note: The calling application must pass the events. The pricing engine searches the set of data belonging to the phases for the events passed. Multiple events can be concatenated in a single pricing engine call.

The following table outlines the action for each of the seeded events in Oracle Advanced Pricing:

Table 12–1 Seeded events in Oracle Advanced Pricing

Event	Action	Function Which Calls the Pricing Engine
Price	Fetch list price	User enters item quantity and unit of measure
Line	Enter order line	User exits the order line
Order	Save order line	User saves order
Book	Book order	Order is booked
Ship	Enter shipments	Order is ship confirmed
Reprice Line	Reprice line	Order is shipped, prior to invoice
Batch	Batch processing	Order Import

Note: INV: Batch Processing for Intercompany Transfer Pricing event is only used by the Oracle Inventory application for pricing their transactions.

What are Pricing Phases?

A pricing phase controls which list types (prices and modifiers) are considered by the search engine and the sequence in which they are applied to a pricing request. The attributes of a pricing phase enable you to control which modifiers are placed in a phase. When you assign a modifier to a pricing phase, the Modifier Setup window matches the attributes of the modifier with the attributes of the available pricing phases to validate which pricing phase or phases a modifier can be placed in. A modifier can only be assigned to one phase.

The following table summarizes the seeded pricing phases in Oracle Advanced Pricing:

Table 12–2 Seeded pricing phases in Oracle Advanced Pricing

Phase Sequence	Name	Level	List Type	Incompatibility Resolve Code	Freeze Override
0	List line base price	Line	PRL	Precedence	N/a
10	List line adjustment	Line	N/a	Precedence	N/a
30	All lines adjustment	N/a	N/a	Precedence	N/a
40	Header level adjustment	Order	N/a	Precedence	N/a
50	Line charges	Line	Charges	Precedence	Yes
60	Line charges - manual	Line	Charges	Precedence	Yes
70	Header level charges	Order	Charges	Precedence	Yes

Assigning Pricing Phases

Key Implementation Decision: What phases and events meet my business requirements?

Phases and events are very specific methods of Oracle Advanced Pricing. Because events separate the order cycle into points attachable to pricing actions, you can exercise control over what pricing actions are taken and when this occurs. You must decide to which phases your discounts and promotions (modifiers) belong.

You can map a pricing event to several pricing phases, and a pricing phase can be assigned to more than one pricing event. This helps to define which pricing phases are to be processed and in which pricing event.

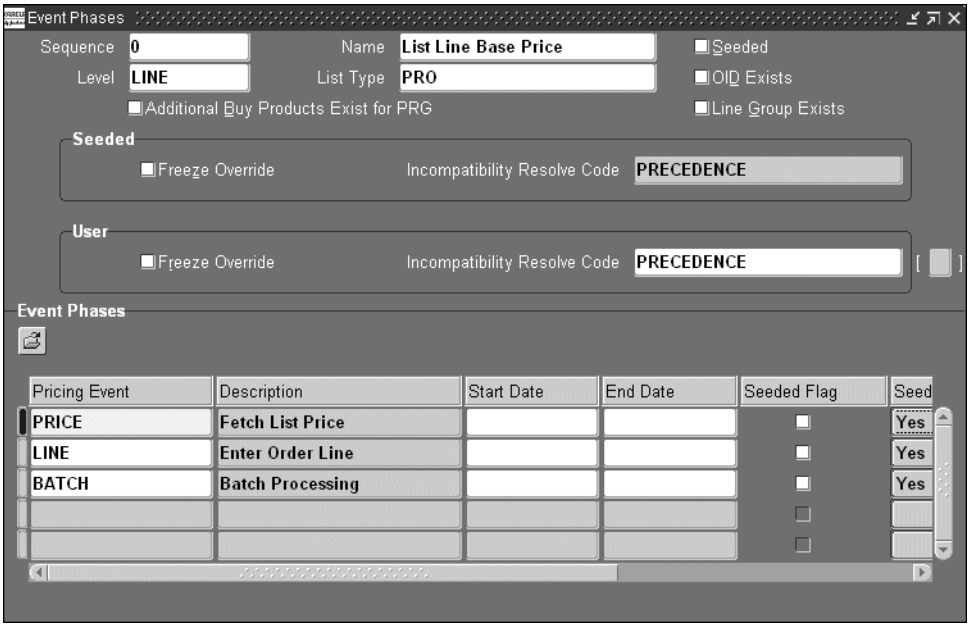
> **Note:** Do not add any modifier phase to the pricing event. Do not assign group of lines and other item discount modifier phase to a line level event; the pricing engine may not have all the necessary order lines. If you add any phase to line or order event, those phases must be added to the batch event.

Key Implementation Decision: Have I defined my modifier groupings so that the seeded phases are sufficient for my pricing actions?

You can accomplish this through the Event Phases window in the Setup menu. Remember the following:

- Place line level discounts in line adjustments phase.
- Place modifiers that span multiple lines in the all lines phase.
- Place modifiers that apply to shipping in ship event phase.

Figure 12–1 Event Phase window



The following describe the fields in the Event Phases window:

Sequence: This field is required. Its numerical value is equivalent to the phase number. The pricing engine uses this number to determine the execution order of the phases when there are multiple phases in an event.

Name: This field is required and appears to Modifier Setup window users when assigning a modifier to a phase. The field name should describe timing and contents of a phase.

Level: This field is optional. To restrict the modifiers in a particular phase to modifiers of a particular modifier level, enter the level in this field.

List Type: This field is optional. To restrict the modifiers in a phase to modifiers on a modifier list, enter the list type in this field.

Seeded and User: Two sections appear below the Sequence, Name, Level, and List Type fields: one entitled Seeded and the other entitled User. These two sections

are used to differentiate the seeded values from the user-entered values. In the User section, you can update the freeze override flag and incompatibility resolve code.

OID Exists: This cannot be entered by the user. This box is checked if there are any modifiers of the type other item discount for a particular phase.

Line Group Exists: This cannot be entered by the user. This box is checked if there are modifiers in the level group of lines for a particular phase.

Additional Buy Products Exist: This cannot be entered by the user. This box is checked if you defined modifiers for promotional goods or other item discounts and you will define additional buy products.

Freeze Override Flag: This flag provides additional control over freezing lines on your transaction. If calculate price sent from the calling application on the request line is set to P, the pricing engine looks at this Freeze Override Flag on the phase. If it is checked, the pricing engine applies eligible modifiers in this phase to the request line. If it is not checked, the modifiers in this phase are not considered for application to the request line. This flag can also be updated on seeded pricing phases.

Incompatibility Resolve Code: This field is required. Incompatibility resolve code refers to the method used to determine which modifier is selected when multiple modifiers in the same exclusivity or incompatibility group are eligible to be applied to the same pricing request line. The incompatibility resolve codes are as follows:

- Best price: The modifier which gives the lowest price (most advantageous) to a customer on the given pricing request line is applied.
- Precedence: The modifier with the lowest (most specific) precedence on the given pricing request line is applied

The incompatibility resolve code value can be updated on seeded pricing phases.

Pricing Event: This field assigns the pricing event you are linking to a phase. Multiple pricing events may be entered for a single phase. Valid values are (pricing event lookup):

- Batch
- Book
- Order
- Line

- Price
- Ship

Start Date: This field is optional. Enter the date for the event to start using the pricing phase.

End Date: This field is optional. Enter the date for the event to stop using the pricing phase.

Search Flag: This flag is used by the pricing engine to decide whether the engine should perform a search for price or modifier lists in addition to any that the calling application has requested.

The search flag should be set to No when the calling application has already identified which price and modifier lists it will use to price the request. The pricing engine then uses the lists that are passed; it does not attempt to find any other lists. For example, a number of discounts have been negotiated and recorded on a customer's service contract. When applying discounts to the service order, the application already knows which discount list should be used to price the order.

The search flag should only be set to No in the circumstances just described; in all other cases the search flag should be set to Yes. If you set the flag to No and the calling application does not pass all the required pricing information, the prices and modifiers may not be applied to the transaction.

Price List Search Based on Search Flag (Extended Search)

The pricing engine first tries to find the price from a price list after doing all the qualifications necessary to qualify for this passed price list. If the price is not found, the pricing engine attempts to get the price from a secondary price list (if present).

If the price is not found on the secondary list, the pricing engine searches for the price across all available price lists and tries to give the price from a price list with highest precedence. For the pricing engine to do this extended search across all price lists in the system, the Search Flag on the pricing phase under ALL the Pricing Events (which include the pricing phase sequence = 0) needs to be consistently set to Yes. This can be done by setting the User Search Flag to Yes.

Example

Passed price list = Corporate

Step 1: The pricing engine searches for the price from the Corporate price list. If the price is not found, the pricing engine tries to find a price from all the secondary

price lists from the Corporate price list. If the price is still not found, the pricing engine completes Step 2.

Step 2: If your business requires that the pricing engine searches across all the price lists in the system, set the User Search Flag to Yes on all the pricing phases in all pricing Events.

13

Pricing Engine Request Viewer window

This chapter discusses the Pricing Engine Request Viewer which is used to capture and display the inputs and outputs of the pricing call. The following topics are discussed:

Overview

The Pricing Engine Request Viewer window captures and displays the inputs and outputs of the pricing call. It captures the pricing engine call from any calling application, such as OM, iStore, Order Capture, or Oracle Contracts Core. The information displayed by the Pricing Engine Request Viewer enables you to diagnose which lines were selected or rejected by the pricing engine to determine why certain prices and adjustments were or were not applied.

Historical data is maintained because the Pricing Engine Request Viewer window updates the display information each time the pricing engine captures a new transaction. The latest pricing request is displayed. Previous pricing requests are saved in Pricing tables.

Using the Pricing Engine Request Viewer window, you can do the following:

- View the controls passed by the calling application to the pricing engine such as Events, Rounding Flag, Search and Calculate Flag, GSA Flag.
- View price request line passed in by the calling application.
- View which modifier lines the pricing engine applied or rejected for benefit adjustments along with the details of the modifier line.
- View the pricing, qualifiers, and product attributes passed to the pricing engine along with the other data generated by the engine.
- View the relationship between order lines for promotional modifiers, price breaks, and service lines (OID, PRG, PBH, Service Items).
- View the formula step values generated by the pricing engine used in formula calculation.
- View and query fields in the Pricing Debug Log.

Setting up the user profiles

QP: Debug: To start the Pricing Engine Request, set the value of the profile option QP: Debug to Request Viewer On; alternately select Request Viewer Off to turn it off. This profile option can be updated at the user level. The pricing engine request viewer is active for the transactions of the user who set this Profile option-other users' transactions are not affected. The default value is set to Request Viewer Off.

If the profile is set to Request Viewer On, but Debug Log is not visible in the Request Viewer. The Request Viewer captures pricing request details into the pricing debug tables, but the debug log information is not written into the debug log table. The debug log text file will be created.

QP: Set Request Name: Set the value of the profile option QP: Set Request Name to append the value of the profile with Order ID to be stored in the Request Name field. The default value is Null.

> **Note:** The Pricing Engine Request Viewer window is available from within Oracle Order Management. The navigation path is: Sales Order window > Tools > Pricing Engine Request Viewer. It is also available on the menu for Pricing Manager Responsibility.

The Pricing Engine Request window is available from within Oracle Order Management. The navigation path is: Sales Order window > Tools > Pricing Engine Request Viewer. It is also available on the menu for Pricing Manager Responsibility.

End-to-End Process for the Pricing Engine Request Viewer

The following series of activities occur when a pricing call is made:

1. The calling application makes a call to the Build Attribute Mapping Rules package to generate the attributes defined by the Attribute Mapping Function.
2. The calling application then calls the pricing engine with the attributes generated by attributes mapping.
3. The pricing engine processes the request. Then searches for and evaluates eligible price list and modifier lines.
4. If the profile option QP: Debug is set to Request Viewer On, then pricing engine inserts records into the permanent pricing debug tables and generates a unique request ID, storing the information from the calling application.

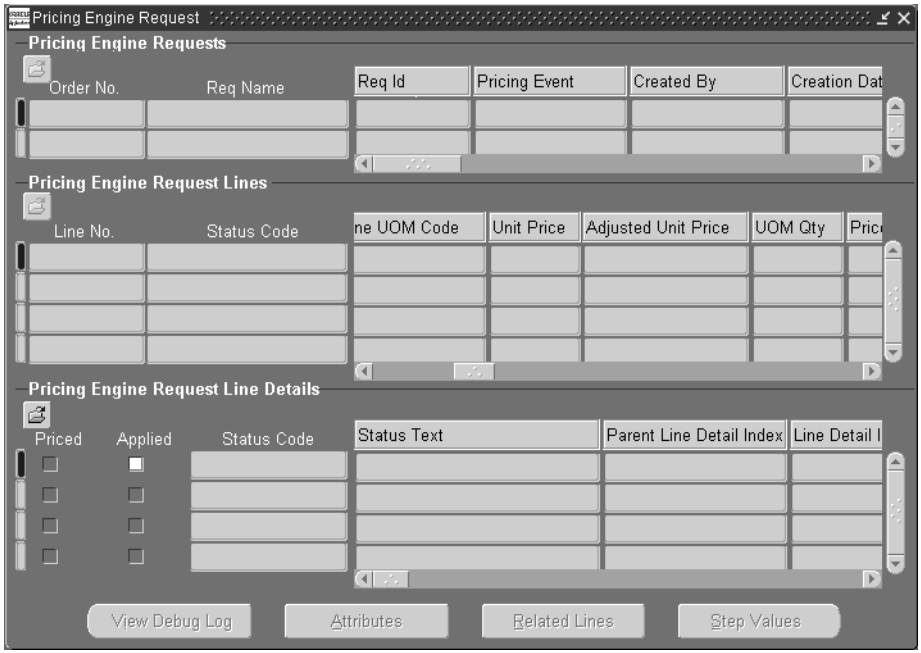
5. The pricing request information can then be viewed by querying the request in the Pricing Engine Request Viewer from the OM Sales Order Pad or through the Pricing Manager responsibility menu.

Regions in the Pricing Engine Request Viewer

The pricing engine request details are displayed in one or more of the following regions in the Pricing Engine Request window:

- Pricing Engine Requests region
- Pricing Engine Request Lines region
- Pricing Engine Request Line Details region.

Figure 13–1 Pricing Engine Request Viewer window



Pricing Engine Requests region

This region maps to the QP_DEBUG_REQ table and displays the following information about the pricing engine requests with associated controls sent by the calling application:

Order No: For Request Type ONT only, the order number associated with the request is displayed.

Note: Depending on the version of Oracle Pricing installed, the order and line numbers for orders created in prior releases may not display in the Pricing Engine Request window. However, order and line numbers created in subsequent releases can be viewed.

Req Name: Order Id. of the calling application is appended with the value of the profile QP: Set Request Name and is stored in this field.

Req Id: Request Id is the sequence number, which the window provides to uniquely identify the Pricing Engine requests.

Pricing Event: A point in the transaction life cycle of your transaction at which you wish to price it. The pricing event determines which phases the search engine processes according to the mapping in QP_EVENT_PHASES.

Created By: Name of the user who created this request.

Creation Date: Standard WHO Column. Contains the Date+ Time Stamp at which the record was created.

Calculate Flag: This refers to the Calculation_Flag in control record, which is passed to the pricing engine. From all the requesting systems the value for this flag is always passed as Y. Eligible values are:

- *N*: Search engine: If you do not want the engine to calculate the selling price.
- *C*: Calculate engine: If you are passing the adjustment records to the engine and you want the engine to recalculate the selling price, without retrieving new adjustments.
- *Y*: Both calculate and search engine: Regular engine call. Retrieves new adjustments and calculates the selling price

Request Type: Identifies the transaction system making the pricing request.

Rounding Flag: Indicates whether the calculation engine should round the list price and selling price based on the price list rounding factor or the pricing request line record rounding factor. When rounding, the calculation engine rounds all intermediate subtotals for each pricing group sequence.

- If set to Y , engine should apply the rounding factor defined in the price list.
- If set to N, unrounded figures would be returned.

- If set to Q, then refer to the value of the profile QP: Selling Price Rounding Options.

GSA Check Flag: Indicates whether the pricing calculation engine should test for GSA Violations:

The evaluation is performed if a request is for a non-GSA customer, and GSA rules are violated if the selling price of an item is calculated to be less than the price of the item on any GSA price list. Allowable values are:

- *Yes*: Price Calculation engine tests for GSA violations, any violating request lines are returned to the calling application with a status of GSA violation.
- *No*: Do not test for GSA violations.

The value of this field is controlled by profile in the requesting systems.

GSA Dup Check Flag: Indicates that the engine should perform GSA duplicate check. The calling application sets this flag to Yes to have the pricing engine test for GSA violations or No so that the pricing engine does not test for GSA violations.

Temp Table Insert Flag: This flag is set to Y, if the calling application directly inserts data into the pricing temporary tables, set to No if the pricing engine inserts into Temporary tables.

Manual Discount Flag: This flag is introduced to support new Release 11*i* functionality. This value is set by the calling application and the value is based on the profile QP: Return Manual Discounts. Indicates how pricing engine should perform incompatibility processing for manual discounts. The possible values for this profile are:

- *Yes*: All the manual discounts will be returned. All the automatic discounts that get deleted as part of incompatibility processing will be returned as manual discounts.
- *No*: All automatic and manual discounts will go through incompatibility processing and one of them in each incompatibility group will be returned. In this process an automatic discount might get deleted and a manual discount might get selected.

Debug Flag: Displays the value of the profile: QP_DEBUG.

Source Order Amount Flag: If set to Y, Indicates to the pricing engine to source the order amount. It means the calling application will provide the order amount. If set to N, The Pricing engine should calculate the order amount.

Public API Call Flag: This indicates whether public API is being used to call the Pricing Engine.

If set to Y, the public API, QP_PREQ_PUB is used to call the pricing engine. If set to N, the group API QP_PREQ_GRP is used to call the pricing engine.

Manual Adjustments Call Flag: This value is set by the calling application. Indicates whether pricing engine should consider manual discounts. The allowable values are:

- Yes: Apply the manual discounts. Means unit price is not calculated by pricing engine.
- No: Do not apply the manual discounts. Means new unit price is calculated by pricing engine.

Check Cust View Flag: This is used for internal engine use.

Currency Code: Currency in which the pricing engine priced. The value for this field should be same across all lines.

Pricing Engine Request Lines region

This region maps to QP_DEBUG_REQ_LINES table and displays the following information about the lines being priced including unit price and adjusted unit price. You can also view information related to service and serviceable lines in this region.

Line No: Unique identifier of the request line in the calling application. For example, Order Number/Quote Number/Contract Number.

Line Id: Unique identifier of the request line in the calling application. For example, Order Number/Quote Number/Contract Number.

Line Index: PL/SQL unique identifier for request line.

Related Line Index:

Line Type Code: Type of line within the request. Eligible values are:

- ORDER
- LINE

Pricing Date: Date and Time for which the pricing engine calculates the prices.

Line Qty: Pricing request line quantity.

Line UOM Code: Pricing request line unit of measure.

Unit Price: Unit price of the item that is expressed in Priced UOM Code.

Adjusted Unit Price: Price per unit after the pricing engine applies discounts and Surcharges. It indicates the unit price for the service item, which has the percent price.

UOM Qty: This holds service duration expressed in 'Line UOM Code'. Unit of measure quantity, for example, in service pricing, LINE_UOM_CODE is Months and UOM_QUANTITY is 2.This field is used for service item pricing.

Priced Qty: Quantity of pricing request line that pricing engine has priced.

Priced UOM Code: Unit of measure in which the pricing engine priced.

Currency Code: Currency in which the pricing engine priced. The value for this field should be same across all lines.

Price Flag: Indicates the degree to which the price is frozen. Allowable values, based on lookup type CALCULATE_PRICE_FLAG are:

- Y (Calculate Price): Apply all prices and modifiers to the request line.
- N (Freeze Price): Do not apply any prices or modifiers to the request line. Consider the volume of the request line when processing LINEGROUP modifiers for other lines.
- P (Partial Price): Apply prices and modifiers in phases whose freeze override flag is Y.

Percent Price: Price calculated as a percentage of the price of another item.

Parent Price: When the pricing engine determines the price of an item from the price of another item, the price of the related item. This is used only for service items and it is populated from the serviceable item.

Parent Qty: When the pricing engine determines the price of an item from the price of another item, the quantity of the related item.

Parent Uom Code: When the pricing engine determines the price of an item from the price of another item, the unit of measure of the related item.

Processing Order: This field is used for service pricing. It indicates the order in which pricing will be done for order lines related to service pricing.

Processed Flag: Indicates whether line has been processed by engine or not. Possible values:

- *No*: Not processed
- *Yes*: Processed

Used for internal engine use.

Processed Code: Internal code which indicates the stage of engine processing when an error occurred.

Active Date First Type: The date type of ACTIVE_DATE_FIRST based on lookup type EFFECTIVE_DATE_ TYPES. Default value is 'date Ordered'.

Start Date Active First: In addition to the pricing effective date, you can specify two additional dates for the pricing engine to use to qualify pricing entities. The pricing engine compares this date against the first date range on the modifier list - QP_LIST_LINES.START_DATE_ACTIVE_FIRST and QP_LIST_LINES.END_DATE_ACTIVE_FIRST.

Active Date Second Type: The date type of ACTIVE_DATE_SECOND based on lookup type EFFECTIVE_DATE_ TYPES. Default value is 'Requested Ship Date'.

Start Date Active Second: In addition to the pricing effective date, you can specify two additional dates for the pricing engine to use to qualify pricing entities. The pricing engine compares this date against the first date range on the modifier list - QP_LIST_LINES.START_DATE_ACTIVE_SECOND and QP_LIST_LINES.END_DATE_ACTIVE_SECOND.

Status Code: Returned status. Allowable values are:

- N: New record created (All 'N' records are returned back from the pricing engine. These are success records)
- X: Unchanged (Default status when the line is passed to the pricing engine for processing)
- D: Deleted
- U: Updated
- IPL: Invalid price list (When passed in price list is not found, then an error is given)
- GSA: GSA violation
- FER: Error processing formula
- OER: Other error
- CALC: Error in calculation engine
- UOM: Failed to price using unit of measure
- INVALID_UOM: Invalid unit of measure
- DUPLICATE_PRICE_LIST: Duplicate price list
- INVALID_UOM_CONV: Unit of measure conversion not found
- INVALID_INCOMP: Could not resolve incompatibility
- INVALID_BEST_PRICE: Could not resolve best price.

Status Text: Returned message from Pricing Engine.

Group Qty: Sum of the quantity of group of lines. Used for internal engine use.

Group Amount: Sum of the price of group of lines. Used for internal engine use.

Line Amount: Price for the line quantity. Used for internal engine use.

Rounding Factor: If ROUNDING_FLAG = Y and the pricing event does not include the base price phase, the rounding factor that the pricing engine should use.

Updated Adjusted Unit Price: To update the adjusted unit price or to manually override the selling price, the calling application writes the new manual updated price into this field and calls the pricing engine. Pricing engine will try to apply the eligible manual adjustments if it can and calculate the new price or if it can't apply the manual adjustments then it will raise an error at that line.

Price Request Code: Unique identifier for order line used by limits processing. It will have the structure 'Request Type Code-Order Id-Line Id'. Used for internal engine use.

Hold Code: Whenever the limit is adjusted or exceeded and 'limit_hold_flag' is 'Y' the engine sets value of this field to 'LIMIT'. Used for internal engine use.

Hold Text: Whenever the limit is adjusted or exceeded and 'limit_hold_flag' is 'Y' the engine sets the value of the field 'HOLD_CODE' to 'LIMIT' and an appropriate message is set to 'HOLD_TEXT'. Used for internal engine use.

Price List Header: Name of the list header used to create or update the pricing line.

Validated Flag: This field is related to PRICE_LIST_HEADER_ID. If set to 'Y', Indicates that the price list is validated and no qualification check is necessary. If set to 'N', Indicates that the price list is not validated.

Qualifiers Exist Flag: This field is related to PRICE_LIST_HEADER_ID. If set to 'Y', Indicates that the qualifiers exist for the price list. If set to 'N', Indicates that the qualifiers doesn't exist for the price list.

Pricing Attrs Exist Flag: This field is related to PRICE_LIST_HEADER_ID. If set to 'Y', Indicates that pricing attributes exist for the price list. If set to 'N', Indicates that pricing attributes doesn't exist for the price list.

Primary Qual Match Flag: This field is related to PRICE_LIST_HEADER_ID. If set to Y, Indicates that qualifiers exist for primary price list. If set to 'N', Indicates that qualifiers doesn't exist for primary price list.

Usage Pricing Type: Indicates the usage pricing type. Allowable values are:

- Regular
- Billing
- Authoring

The lines region can be used to locate the source of a problem. For example, from the lines region of the Pricing Engine Request window, a specific line in the lines region shows the expected adjusted unit price (map to unit selling price). From the Sales Order window, we observe that the unit-selling price is blank and does not display an expected unit-selling price. The problem is in the pricing integration code. A price is generated, but Oracle Order Management does not display it.

Pricing Engine Request Line Details region

This region maps to the QP_DEBUG_REQ_LDETS table. The Req Id + Line Index column maintains the master-detail relationship between lines and line details. This region shows information regarding processed price list lines and modifiers lines selected and/eliminated by the engine.

The Priced box indicates which lines were finally selected for pricing by the pricing engine. The Applied box indicates which lines were considered in calculating the selling price. This region also displays the information for item upgrades, coupon issue, term substitution, freight and special charges, and relationships between price breaks. The information that displays includes the following:

Selected: This indicates whether the pricing engine successfully selected all the adjustments (Both manual & automatic). Set to Y if the line is considered for pricing by the Pricing Engine. Set to N if the line is rejected by the pricing engine.

Applied: The lists or list lines that this pricing event or a prior pricing event applied. Allowable values are:

- Yes: Applicable when the attribute context is a list or list line
- No: Not applicable when the attribute context is a list or list line

Status Code: Indicates returned status. Possible Values:

- New record created (All 'N' records are returned back from the pricing engine. These are success records)
- Unchanged (Default status when the line is passed to the pricing engine for processing)
- Deleted
- Updated
- Invalid price list (When passed in price list is not found, then an error is given)
- GSA violation
- Error processing formula
- Other error
- Error in calculation engine
- Failed to price using unit of measure
- Invalid unit of measure

- Duplicate price list
- Unit of measure conversion not found
- Could not resolve incompatibility
- Could not resolve best price

Pricing Status Text: Returned message from Pricing Engine.

Parent Line Detail Index: PL/SQL unique identifier. Unique identifier of 'request line detail' in calling application.

Line Detail Index: PL/SQL unique identifier. Unique identifier of 'request line detail' in calling application.

List Type Name: List type of the line used. Possible values can be found from the lookup type LIST_TYPE_CODE from qp_lookup table.

Price List Name: Price List Name of the line used.

Modifier Number: Modifier list number.

Modifier Name: Modifier list name. This field is not updateable by user.

List Line Type: Line type of the list line used to update the pricing line. Possible values can be found from the lookup type LIST_LINE_TYPE_CODE from qp_ lookups table.

List Line Number: Modifier list line number.

Modifier Level Code: The level at which the list line qualified for the transaction. Based on lookup type MODIFIER_LEVEL_CODE.

Application Method: Type of operand. Allowable values are:

- Adjustment percent (for discounts)
- Adjustment amount (for discounts)
- Adjustment New Price (for discounts)
- UNIT_PRICE (for price lists)
- PERCENT_PRICE (for price lists)

- LUMPSUM

Value: Value of pricing request detail line.

Adjustment Amount: It indicates the dollar value of the adjusted amount. It holds the value of the bucketed adjusted amount for line types like PLL, DIS, and SUR etc. For price break (PBH) child lines, the field is populated if the pricing engine derived the value of the request line or request line detail from a price break.

Automatic Flag: If set to 'Y' it indicates that the pricing engine automatically applied the request line detail to the request line. The engine derives the value from the list line.

Bucket: Indicates the pricing bucket in which the pricing engine applied this list line. If 'MODIFIER_LEVEL' is 'ORDER' or if 'AUTOMATIC_FLAG' is set to 'N', the value in this field can't be modified.

Pricing Phase: The pricing phase which created the request line detail.

Price Formula: Formula attached to the price list line or modifier line.

Incompatibility Group: This specifies that the discount is incompatible with all other discounts in this incompatibility group. Incompatibilities can be specified for discounts across Modifier types.

Override Flag: Indicates if a user in the calling application can override the modifier value. No restriction in place to modify the 'OPERAND_VALUE' irrespective of value in this flag.

Charge Type: Indicates the type of charge based on lookup type FREIGHT_CHARGES_TYPE. Used for Freight/Special Charge-type modifiers.

Charge Sub Type: Indicates the type of charge based on lookup type CHARGE_TYPE_CODE.

Item Upgrade Value From: Original Item.

Item Upgrade Value To: Upgraded Item. The Item and its upgrade item must be related and the relationship is defined in Oracle Inventory screen.

Ask for Flag: Used for internal engine use. If set to 'Y' it indicates that the selected modifier is 'ASK FOR' modifier.

Processed Flag: Used for internal engine use. Indicates whether line has been processed by engine or not. Possible values:

- No: Not processed
- Yes: Processed

Created From SQL: Indicates which cursor was used by the engine to select the modifier. Used for internal engine use. Possible values:

- PRODUCT_ONLY
- EXCLUDED_PRODUCT_ONLY
- QUALIFIER_ONLY
- PRODUCT_QUALIFIER_ONLY
- PRODUCT_PRICING_ONLY
- PRODUCT_QUALIFIER_PRICING_ONLY
- INSERTED IN SECONDARY LIST HEADER SEARCH
- INSERTED IN VALIDATED LIST_HEADER_SEARCH1
- INSERTED IN NOT VALIDATED LIST_HEADER_SEARCH1
- INSERTED IN VALIDATED ASKED FOR PROMOTION SEARCH
- INSERTED IN NOT VALIDATED QUAL_LIST_HEADER_SEARCH
- INSERTED BY CREATE_QUALIFIER_FROM_LIST

Line Quantity: Quantity on the price break line. Populated if the pricing engine derived the value of the request line or request line detail from a price break. A not null value indicates that this particular break line was used in the calculation.

Product Precedence: It indicates the rank of preference given for the Qualifiers/Pricing Attributes. For the same item if there are more than one incompatible discounts qualifying then the discount with the higher precedence is given.

Best Percent: Modifier percentage that gives the best price. Used for internal engine use.

Primary Uom Flag: If set to Yes it indicates that if the price cannot be found for a product in the UOM passed from the calling application, the pricing engine will

convert the transaction quantity to the primary UOM specified. Applicable only to price lists.

Benefit Quantity: The accrual quantity for non-monetary accruals or, for promotional goods, item quantity

Benefit Uom Code: The accrual unit of measure for non-monetary accruals, or for promotional goods, item unit of measure.

Accrual Flag: Indicates whether the discount is an accrual.

Accrual Conversion Rate:

The rate to use when converting a non-monetary accrual to a monetary value.

Estim Accrual Rate: Indicates the percentage at which to accrue or, for a coupon, the expected rate of redemption of the coupon. Liability is defined as: ACCRUAL OR COUPON VALUE * ESTIM_ ACCRUAL_RATE. Default Value: 100.

Rounding Factor: If ROUNDING_FLAG = Y and the pricing event does not include the base price phase, the rounding factor that the pricing engine should use. This value is passed in by the calling application.

Secondary Price List Ind: Indicates that the pricing used a secondary price list instead of the price list that the calling application requested. Applicable only to price lists.

Group Quantity: Sum of the quantity of group of lines. Used for internal engine use.

Group Amount: Sum of the price of group of lines. Used for internal engine use.

Process Code: This is set by the engine and used for selecting lines for calculation. Used for internal engine use. Possible values:

- *N*: New
- *D*: Deleted
- *X*: Unchanged

Updated Flag: This value is passed in by the calling application. Used for internal engine use.

Limit Code: The value of this field is set to ADJUSTED when limit is adjusted, set to EXCEEDED when the limit is exceeded. Used for internal engine use.

Limit Text: Returned message from Pricing Engine whenever limit is Exceeded or Adjusted.

Header Limit Exists: Value is set to Y if the Header Limit exists.

Line Limit Exists: Value is set to Y if the Line Limit exists

Attributes window

The line attributes region maps to QP_DEBUG_REQ_LINE_ATTRS table. This region displays information about the pricing attributes that the attribute mapping function passed to the pricing engine. The pricing engine uses these attributes to qualify a line or an order for price and adjustments. From the Pricing Engine Request Viewer window select the Attributes button to display all attributes for a selected line or line detail. If you select the Attributes button from Request Lines region, the attributes displayed will be attributes passed to the pricing engine.

If you select the Attribute from the Request Line Details region, the attributes displayed will be the attributes related to the selected price list lines and modifier lines.

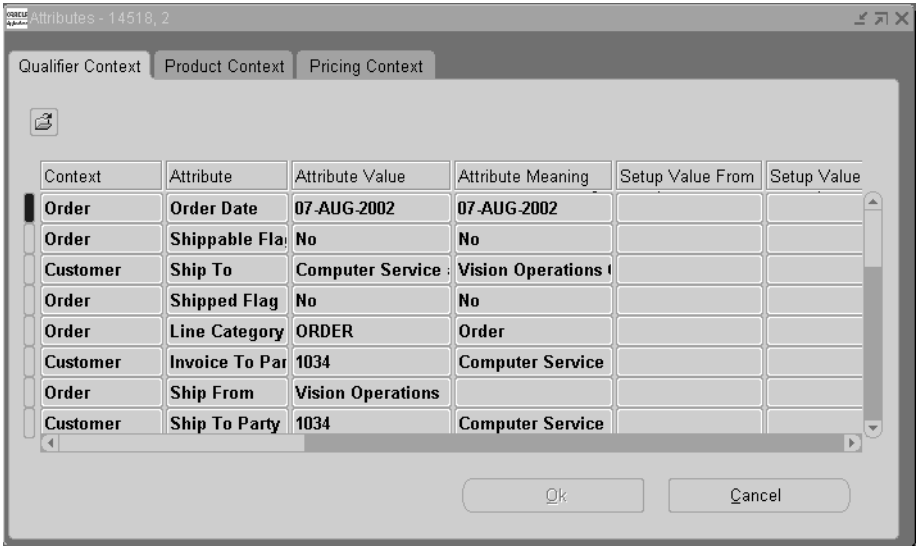
Use this table when Oracle Order Management does not return an expected discount or adjustment and the list line ID do not appear in line details region.

The Attributes window consists of three tabs: Qualifier Context, Product Context and Pricing Context.

Qualifier Context tab

The following image depicts the Qualifier Context tab from the Attributes window:

Figure 13–2 Qualifier Context tab: Attributes window



Columns in this tab include:

Context: Context for a product or pricing attribute, for example, Product Hierarchy.

Attribute: Product or pricing attribute, for example, PRICING_ATTRIBUTE11: Customer Item ID.

Value From: Passed in value for product or pricing attribute.

Setup Value From: Setup value for product or pricing attribute.

Setup Value To: Setup value for product or pricing attribute.

Grouping Number: It indicates the qualifier grouping number which is used to group qualifiers together to create AND/OR relationships.

Validated Flag: If set to 'Y', Indicates that the price list is validated and no qualification check is necessary. If set to 'N', Indicates that the price list is not validated.

Comparison Operator Type: The relational operator code used to define how the pricing engine should evaluate the pricing attributes or qualifier attributes, based on lookup type COMPARISON_OPERATOR.

Applied Flag: The lists or list lines that this pricing event or a prior pricing event applied. Allowable values are:

- *Yes*: Applicable when the attribute context is a list or list line
- *No*: Applicable when the attribute context is a list or list line

Qualifier Precedence: The precedence number, or selectivity of the qualifier attribute in the Qualifier Descriptive Flex field. It is used by the pricing engine for incompatibility resolution.

Data Type: Indicates the data type of the pricing attribute value or qualifier attribute value.

Processed Code: This is set by the engine and used for selecting lines for calculation. Used for internal engine use. Possible values:

- N: New

- D: Deleted
- X: Unchanged.

Distinct Qualifier Flag: To determine qualification engine sets this flag to indicate that this is unique qualifier.

Primary Uom Flag: If set to Yes it indicates that if the price cannot be found for a product in the UOM passed from the calling application, the pricing engine will convert the transaction quantity to the primary UOM specified.

Modifier Number: Modifier list number.

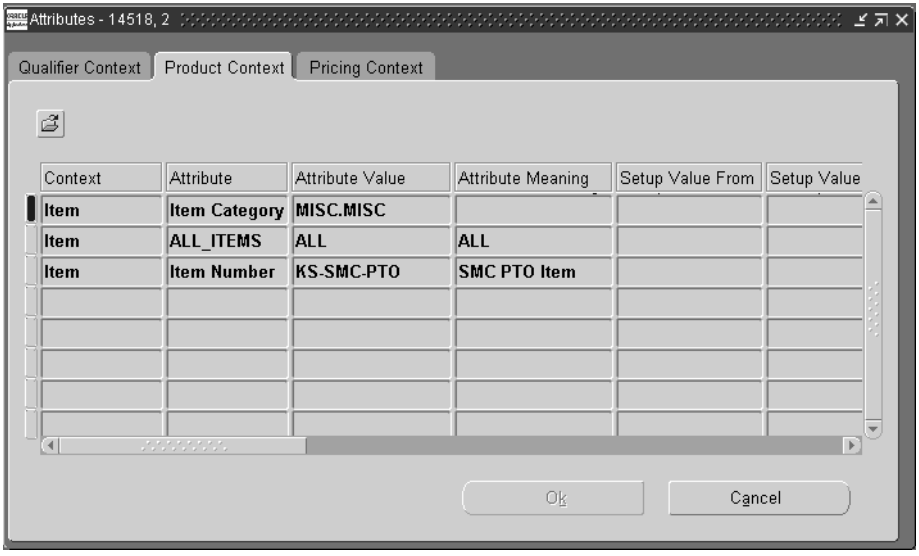
Modifier Name: Modifier list name.

List Line Number: Modifier list line number.

Product Context tab

The following image depicts the Product Context tab:

Figure 13–3 Product Context tab: Attributes window



Columns in this tab include:

Context: Context for a product or pricing attribute, for example, Product Hierarchy.

Attribute: Product or pricing attribute, for example, PRICING_ATTRIBUTE11: Customer Item ID.

Value From: Passed in value for product or pricing attribute.

Setup Value From: Setup value for product or pricing attribute.

Setup Value To: Setup value for product or pricing attribute.

Applied Flag: The lists or list lines that this pricing event or a prior pricing event applied. Allowable values are:

- *Yes*: Applicable when the attribute context is a list or list line
- *No*: Applicable when the attribute context is a list or list line

Qualifier Precedence: The precedence number, or selectivity of the qualifier attribute in the Qualifier Descriptive Flex field. It is used by the pricing engine for in-compatibility resolution. This field is not updateable by user.

Data Type: Indicates the data type of the pricing attribute value or qualifier attribute value.

Product Uom: unit of measure of the item, product group etc. for which the price or modifier is defined.

Processed Code: This is set by the engine and used for selecting lines for calculation. Used for internal engine use. Possible values:

- N: New
- D: Deleted
- X: Unchanged.

Excluded Flag: If set to Yes, it indicates that product value was defined as an excluded item on the modifier line.

Group Qty: Sum of the quantity of group of lines. Used for internal engine use.

Group Amount: Sum of the price of group of lines. Used for internal engine use.

Primary Uom Flag: If set to Yes it indicates that if the price cannot be found for a product in the UOM passed from the calling application, the pricing engine will convert the transaction quantity to the primary UOM specified.

Modifier Number: Modifier list number.

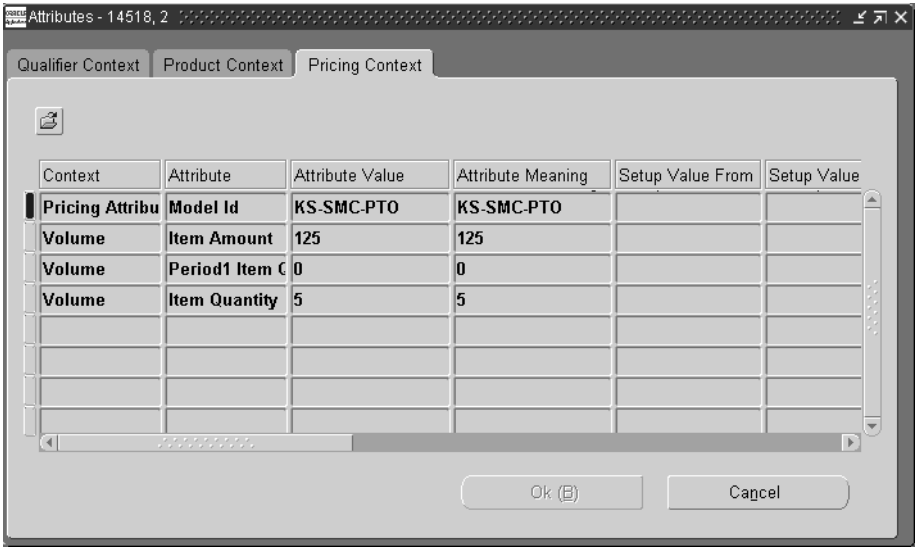
Modifier Name: Modifier list name.

List Line Number: Modifier list line number.

Pricing Context tab

The following image depicts the Pricing Context tab:

Figure 13–4 Pricing Context tab: Attributes window



Columns in this tab include:

Context: Context for a product or pricing attribute, for example, Product Hierarchy.

Attribute: Product or pricing attribute, for example, PRICING_ATTRIBUTE11: Customer Item ID

Value From: Passed in value for product or pricing attribute.

Setup Value From: Setup value for product or pricing attribute.

Setup Value To: Setup value for product or pricing attribute.

Comparison Operator Type: The relational operator code used to define how the pricing engine should evaluate the pricing attributes or qualifier attributes, based on lookup type COMPARISON_OPERATOR.

Applied Flag: The lists or list lines that this pricing event or a prior pricing event applied. Allowable values are:

- Yes: Applicable when the attribute context is a list or list line
- No: Applicable when the attribute context is a list or list line

Data Type: Indicates the data type of the pricing attribute value or qualifier attribute value.

Processed Code: This is set by the engine and used for selecting lines for calculation. Used for internal engine use. Possible values:

- N: New
- D : Deleted
- X: Unchanged.

Primary Uom Flag: If set to 'Y' it indicates that if the price cannot be found for a product in the UOM passed from the calling application, the pricing engine will convert the transaction quantity to the primary UOM specified.

Modifier Number: Modifier list number.

Modifier Name: Modifier list name.

List Line Number: Modifier list line number.

The line attributes window of the Pricing Engine Requests Viewer window can be used to locate the source of a problem. For example, customer 1006 receives a discount. If Oracle Order Management does not return the discount and the line detail region does not display the list line ID in the Pricing Engine Requests Viewer window, the solution would be as follows:

In the line attributes region, query the record that has Context = CUSTOMER,

Attribute = QUALIFIER_ATTRIBUTE2 and Value = 1006.

If you do not find the attribute this qualifier was not passed to the pricing engine. It is possible that the Build Attribute Mapping Rules program was not run after the first use of new type of qualifier (the qualifier is not sourced by the attribute mapping). If you are using your customer-defined qualifier then make sure that the attributes mapping setup is properly defined.

If you do find the attribute, the problem occurred in pricing engine. Contact Oracle Support for a pricing engine debug script.

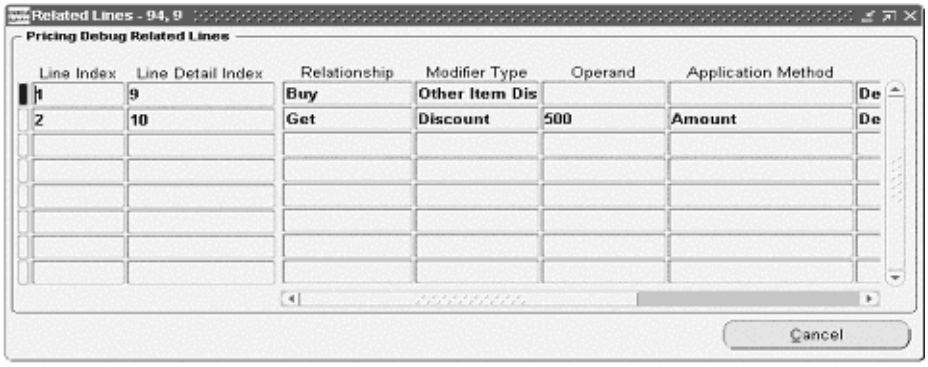
Related Lines window

This window maps to QP_DEBUG_REQ_RLTD_LINES table.

Select the Related Lines button from the Pricing Engine Request Viewer window to display the Pricing Debug Related Lines window. The cursor needs to be in the Request Line Details region. You can view the relationship between the Buy and Get items for Other Item Discounts and Promotional Goods.

The following image depicts the Related Lines window:

Figure 13–5 Related Lines window



Columns in this window include:

Line Index: PL/SQL unique identifier for request line.

Line Detail Index: PL/SQL unique identifier for request detail line.

Relationship: Type of relationship between pricing lines. Allowable values are:

- BUY
- GET

Modifier Type: Line type of the list line used to update the pricing line. Possible values can be found from the lookup type LIST_LINE_TYPE_CODE from qp_ lookups table.

Operand: Value of pricing request detail line, for example, 10 currency unit list price with 3% discount.

Modifier number: Modifier list number.

Modifier Name: Modifier list name.

Application Method: Type of operand. Eligible values are:

- Adjustment percent (for discounts)
- Adjustment amount (for discounts)
- Adjustment New Price (for discounts)
- Unit Price (for price lists)
- Percent Price (for price lists)
- Lumpsum

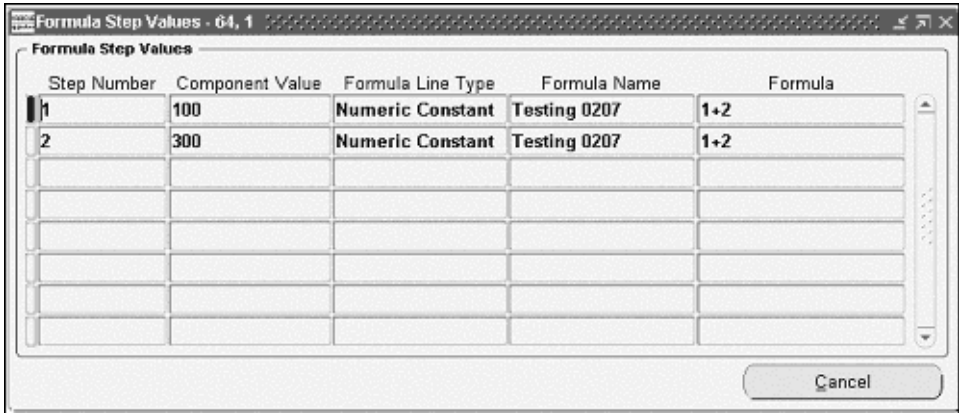
Formula Step Values window

This window maps to QP_DEBUG_FORMULA_STEP_VALUES table. This region shows information about the formula step values, which are inserted into the table QP_FORMULA_STEP_VALUES during the evaluation of formula attached to the price lists. The pricing engine inserts step values into QP_FORMULA_STEP_ VALUES only if the profile QP: Insert Formula Step Values into Temp Table is set to Y.

Select the Step Values button from the Pricing Engine Request Viewer window to display the Formula Step Values window. The cursor needs to be in the Request Line Detail region.

The following image depicts the Formula Step Values window:

Figure 13–6 Formula Step Values window



Columns in this window include:

Step Number: Step number corresponding to a formula line.

Component Value: Evaluated value of a formula step.

Formula Line Type: Type of the formula line. Possible values are:

- FUNC: Function
- LP: List Price
- ML: Factor List

- MV: Modifier Value
- NUM: Numeric Constant
- PLL: Price List Line
- PRA: Pricing Attribute

Formula Name: Name of the Pricing Formula.

Formula: Indicates the mathematical formula for a rule.

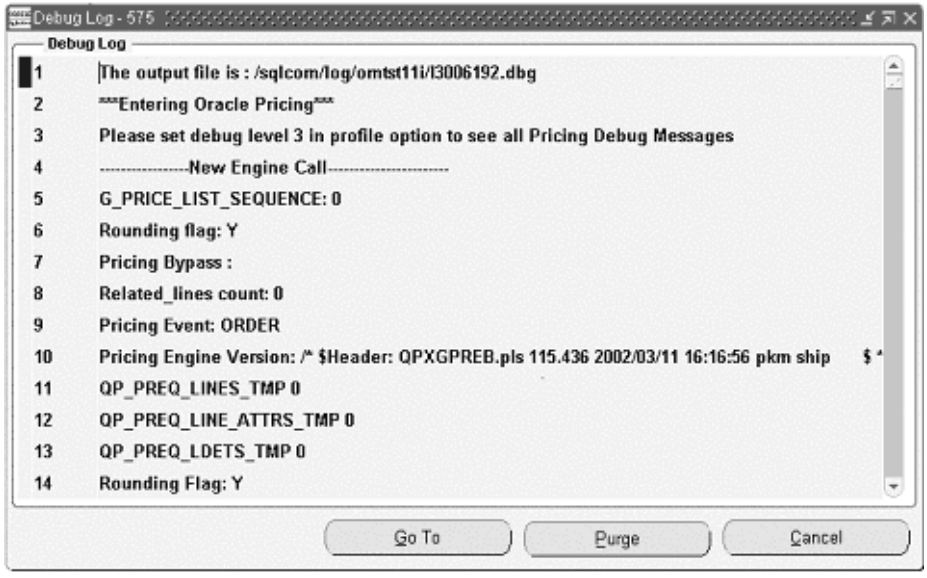
Debug Log window

This window maps to qp_debug_text table. The contents of the debug log file are shown in this window.

Searching the Debug Log (Using GOTO button): User executes a query to search for a string in the Debug Log and the records/lines matching the search criterion are returned. Now user selects any one of the returned records, then selects the GOTO button to see 10 previous records from that line number and the rest of the lines in the debug log.

Purging the Debug Log: User selects the Purge button to delete all the records in the Debug Log.

Figure 13–7 Debug Log



Analyzing error messages

From the Line Details region of the Pricing Engine Requests Viewer window, you can analyze error messages to locate the source of a problem. For example, if Oracle Order Management does not return an expected adjustment (list line), the solution is as follows:

In the Pricing Engine Requests Viewer window, look for the list line ID in the created_from_list_line_id column. If you find the expected list line ID in the window and it has a pricing_status_code of D_GRP, the grouping operation of the pricing engine deleted it.

Check your grouping condition in the Modifier window to see what other conditions must be met to receive the adjustment. If the list line ID has a pricing_status_code of N (accepted by engine) and it is not reflected in pricing integration, then the problem occurs in pricing integration.

Other pricing status codes that can be issued are listed in the table below.

The following table lists the three success codes for line:

Table 13–1 Success codes

Pricing Status Code	Error
N	New record created
U	Updated
X	Unchanged

The following table displays the available internal processing status:

Table 13–2 Internal processing status codes

Pricing Status Code	Error
D	Deleted
T	Transient
B	Best price evaluation
OTHER_ITEM_BENEFITS	Deleted in PRG processing
P_UOM_FLAG	Removed due to primary UOM conversion
I	Deleted during incompatibility
G	Deleted in grouping

The following table lists codes that require action from the calling application:

Table 13-1 Pricing status codes and error description

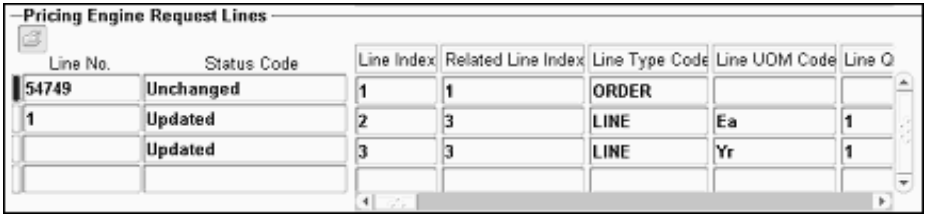
Pricing Status Code	Error
IPL	Invalid price list
GSA	GSA violation
NMS	Item not found in primary or secondary price list
FER	Error processing formula
OER	Other errors
S	System generated message
CALC	Error in calculation engine
UOM	Failed for price unit of measure
INVALID_UOM	Invalid unit of measure
DUPLICATE_PRICE_LIST	Duplicate price lists
INVALID_UOM_CONV	Unit of measure conversion not found
INVALID_INCOMP	Unable to resolve incompatibility
INVALID_BEST_PRICE	Unable to resolve best price
LIMIT	Put limit on hold
EXCEEDED	Limit exceeded

Viewing Service and Serviceable Lines

This Pricing Engine Request Lines region has information related to Service Lines relationship. The fields Related Line Index and Line Index in the Pricing Engine Request Lines region are related. By looking at the values of Related Line Index and Line Index we can identify if the lines are related; for example, Service Item and Serviceable Item.

Relationship between the Service Item and Serviceable Item:

Figure 13–1 Pricing Engine Request Lines region



In this example, Line Index 2 could be a serviceable item (such as Oracle 8i) and Line Index 3 could be a service item (such as Gold Support for 8i).

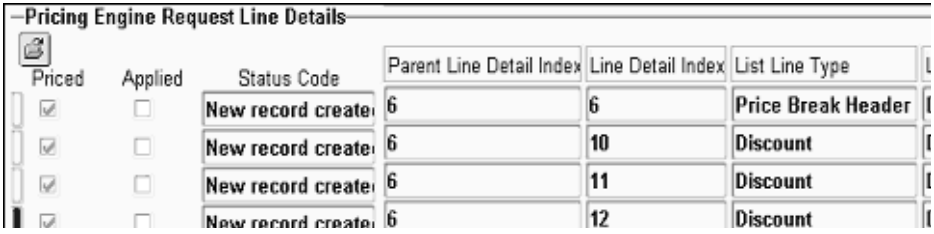
Viewing Price Break lines

This Pricing Engine Request Line Details region has information related to Price Break Lines Relationship. The Parent Line Detail Index field and Line Detail Index field in the Pricing Engine Request Line Details region are related. By looking at the values of Parent Line Detail Index and Line Detail Index we can identify if there is a Price Break. Price Break setup has a Price Break Parent Record, which has a line type called PBH. This PBH record can have more than 1 child record that actually define the breaks.

The following graphic displays a Price Break setup where the Price Break Parent Record (Line Detail Index 6) has a line type called Price Break Header.

This PBH record has three child records (Line Detail Index 10, Line Detail Index 11, and Line Detail Index 12) which define the breaks.

Figure 13–2 Pricing Engine Request Line Details region

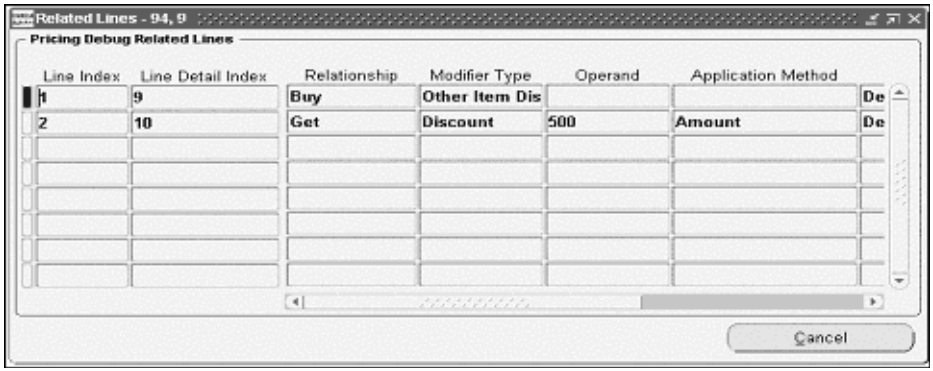


Viewing Other Item Discounts (OID)

In this case 2 request lines are passed to the pricing engine such as Buy ITEM1 and get $500 off on ITEM2.

In this example both ITEM1 & ITEM2 need to be ordered on two order lines. So 2 request lines are created and passed to the pricing engine. When the engine processes the other Item Discounts, it creates discount line for -$500 on the second request line. Also a relationship record is created and this relationship is shown in the Related Lines window. Line Detail Index 9 is the actual Discount Line, which is the OID line and Line Detail Index 10 is the actual benefit line, which is $500 off line. The relationship is as follows:

Figure 13–3 Related Lines window



Viewing Promotional Goods Discount (PRG)

In this case only original buy item request line is passed to the pricing engine such as Buy ITEM1 and get ITEM2 for free.

In this example only request line/order line with ITEM1 is sent to the pricing engine. ITEM2 need not be ordered. The pricing engine selects the PRG Modifier because of purchase of ITEM1 and creates a Line Detail Record (Line Index 1 - Line Detail Index 2). Then it tries to give the benefit, which is a free item ITEM2. In the process engine does the following things:

A new request line (LINE RECORD) is created (Line Index 3).

A new relationship (RELATED LINES RECORD) between the ITEM1 line and ITEM2 line is created. It is a Line-Line relationship (Line Index 1 - Line Index 3).

A new Price List Line (LINE DETAIL RECORD) is created for the new request line (Line Index 3 - Line Detail Index 3).

A new adjustment line for 100% discount is created for the new request line (LINE DETAIL RECORD) (Line Index 3 - Line Detail Index 4).

A new relationship line (RELATED LINES RECORD) between the original PRG Line Detail Line and the new 100% off line detail is created (Line Detail Index 2 - Line Detail Index 4).

A new record (LINE ATTRIBUTE RECORD) is created for the new ITEM2. (Line Index 2 - PRICING_CONTEXT = 'ITEM', PRICING_ATTRIBUTE = 'PRICING_ATTRIBUTE1', PRICING_ATTR_VALUE = 'ITEM2').

The relationship is as follows:

Purging Pricing Engine Requests

The concurrent program Purge Pricing Engine Requests will purge the pricing engine requests. It needs to be run on a regular basis to purge the historical data from the pricing debug tables. Periodically purging the historical data from the pricing debug tables will improve the performance of the 'Pricing Engine Request Viewer' window.

This concurrent program does not affect any of the user procedures on Pricing Engine Request Viewer UI. This concurrent program needs to be run on a periodic basis, especially when the user observes a marked deterioration in the performance of the 'Pricing Engine Request Viewer' window. For more details on this concurrent program, please refer to the Oracle Pricing Users Guide.

Deleting Pricing Engine Requests

Follow these steps to delete a previously saved pricing engine request:

1. From the Pricing Engine Requests Viewer window, choose View > Find to find the Pricing Engine Request to delete.
2. Choose Edit > Delete.

14

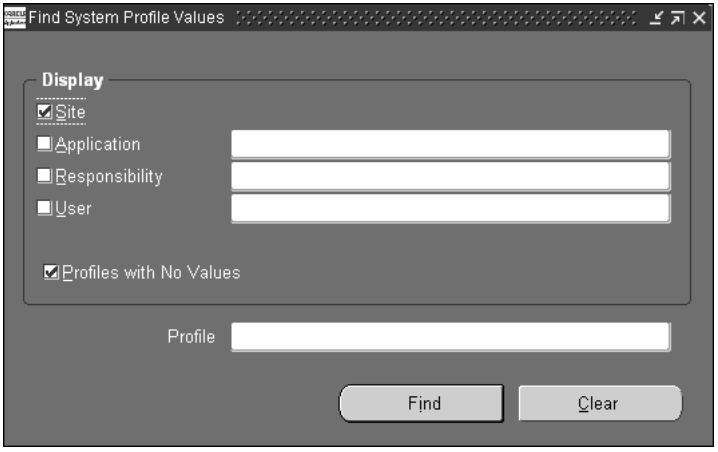
Profile Options

This chapter discusses implementation considerations for profile options and system parameters in Oracle Advanced Pricing. The following topics are discussed:

Profile Options

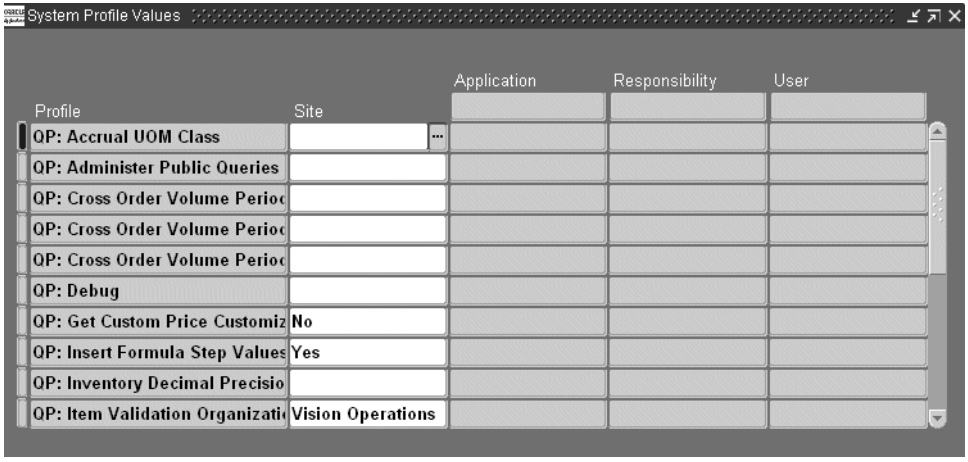
During implementation, you set values for profile options at various levels to specify how Oracle Advanced Pricing controls access to and processes data. The system administrator sets and updates profile values. See: *Oracle Applications System Administrator's Guide*, Setting User Profile Options for more information.

Figure 14–1 Find System Profile Values



In the Profile field, enter QP% and click Find to display all the site level profile options for pricing in the System Profile Value window:

Figure 14–2 System Profile Value window



For each profile option, select a Site value that supports your implementation requirements for Oracle Advanced Pricing.

Profile Options Setup Summary

This section provides a summary of pricing profile options and the the System Administrator level(s) at which the profile options can be viewed and updated. The System Administrator levels are:

- Site
- Application
- Responsibility
- User

The following table lists the short names and definitions used in the following Pricing profile options table:

Table 14–1 Conventions used in the Pricing profile options table

Value	Definition
SA Site	System Administrator: Site level
SA App.	System Administrator: Application level
SA Resp.	System Administrator: Responsibility level

Table 14–1 Conventions used in the Pricing profile options table

Value	Definition
SA User	System Administrator: User level
User	User level
Yes	You can update the profile option.
No	You cannot change the profile option value.
Blank	You must specify a default value or you may encounter a system error.
Req? (Required)	▪ Required (Req.): Requires you to provide a value for the profile option. ▪ Optional: (Opt.) Already provides a default value, so you must change it only if you do not want to accept the default.

For the following table, if the value No is displayed in the Default Value column, the default for the profile option is No.

If implemented with Oracle Order Management, the OM profiles indicated must also be considered in order to work with related QP profile options.

Table 14–2 Pricing profile options and default settings

Profile Option	SA Site	SA App	SA Resp	SA User	Default Value	User	Req?
OM: Allow Negative Pricing	View Only	View Only	View Only	No	-	View Only	Opt
OM: Discounting Privileges	No	Yes	Yes	Yes	Full	No	Opt
OM: GSA Discount Violation Action	View Only	No	No	No	Warning	No	Opt
OM: Show Discount Details on Invoice	Yes	Yes	Yes	No	No	View Only	Req
QP: Accrual UOM Class	Yes	Yes	No	No	Null	View Only	Opt
QP: Administer Public Queries	Yes	Yes	No	No	Null	No	Opt
QP: Allow Duplicate Modifiers (Used by Basic Pricing Only)	Yes	No	No	No	Yes	No	Opt

Table 14–2 Pricing profile options and default settings

Profile Option	SA Site	SA App	SA Resp	SA User	Default Value	User	Req?
QP: Blind Discount Option	Yes	Yes	No	No	Yes	View Only	Opt
QP: Build Attributes Mapping Options	Yes	No	No	No	No	View Only	Opt
QP: Cross Order Volume Period1	Yes	Yes	No	No	Blank	View Only	Req
QP: Cross Order Volume Period2	Yes	Yes	No	No	Blank	View Only	Req
QP: Cross Order Volume Period3	Yes	Yes	No	No	Blank	View Only	Req
QP: Custom Sourced	View Only	View Only	View Only	View Only	No	View Only	Opt
QP: Debug	View Only	No	No	Yes	Request Viewer Off	Yes	Opt
QP: Get Custom Price Customized	Yes	No	No	No	No	No	Opt
QP: Insert Formula Step Values into Temp Table	Yes	No	No	No	No	No	Opt
QP: Inventory Decimal Precision	Yes	Yes	Yes	Yes	10	Yes	Opt
QP: Item Validation Organization	Yes	No	Yes	No	Blank	No	Opt
QP: Limit Exceed Action	Yes	No	Yes	Yes	Hard - Adjust Benefit Amount	Yes	Opt
QP: Line Volume UOM Code	Yes	Yes	No	No	Blank	No	Req
QP: Line Weight UOM Code	Yes	Yes	No	No	Blank	No	Opt
QP: Multi Currency Installed	Yes	No	No	No	No	No	Opt
QP: Negative Pricing	Yes	No	No	No	No	No	Opt
QP: Pass Qualifiers to Get_ Custom_Price API	Yes	No	No	No	No	No	Req
QP: Price Rounding	Yes	No	No	No	Blank	No	Req

Table 14–2 Pricing profile options and default settings

Profile Option	SA Site	SA App	SA Resp	SA User	Default Value	User	Req?
QP: Pricing Transaction Entity	Yes	No	Yes	Yes	Order Fulfillment	No	Opt
QP: Promotional Limits Installed	Yes	No	No	No	No	No	Opt
QP: Qualify Secondary Price Lists	Yes	No	No	No	No	Yes	Opt
QP: Return Manual Discounts	Yes	Yes	Yes	Yes	Yes	Yes	Opt
QP: Satisfied Qualifiers Option	Yes	Yes	Yes	No	Yes	Yes	Opt
QP: Security Control	Yes	No	No	No	Off	No	Opt
QP: Security Default Maintain Privilege	Yes	No	No	No	Global	No	Opt
QP: Security Default ViewOnly Privilege	Yes	No	No	No	Global	No	Opt
QP: Selling Price Rounding Options	Yes	No	No	No	Individual: = round(listprice) + round(adj)	No	Opt
QP: Set Request Name	Yes	Yes	Yes	Yes	Blank	Yes	Req
QP: Source System Code	Yes	Yes	No	Yes	Oracle Pricing	Yes	Opt
QP: Time UOM Conversion	Yes	Yes	Yes	Yes	Oracle Pricing	Yes	Req
QP: Unit Price Precision Type	Yes	Yes	No	No	Standard	No	Opt
QP: Valueset Lookup Filter	Yes	Yes	Yes	Yes	Yes	Yes	Opt
QP: Verify GSA Violations	Yes	No	No	No	No	No	Opt

OM: Allow Negative Pricing

This profile option determines whether or not a negative list price or selling price can be entered on an order. Select either Yes or No. For example: a trade-in item may be entered on an order with the trade-in value as a negative value.

OM: Discounting Privilege

This profile option determines control of a user's ability to apply discounts to an order or order line.

Values

- Full (default value): Ability to apply any valid discount against an order or order line, as long as the order type of the order does not enforce list prices.
- Non-overridable only: Ability to apply only non-overridable discounts against an order or order line.
- Unlimited: Ability to apply any valid discount against any order or order line, regardless of whether the order type of the order enforces list prices.

OM: GSA Discount Violation Action

This profile option determines the user is notified when you define a discount that results in GSA violation.

This profile must be set as Yes if you want order entry personnel to receive a GSA violation warning message, and if the QP: Verify GSA is set to Yes.

QP: Accrual UOM Class

Default value: Null

This is required if your business gives non-monetary accruals as benefits.

Specifies the unit of measure class to be used for defining accrual units of measure. The Modifier Setup window displays all units of measure in this class when entering the Benefit UOM for an accrual.

Values

- All UOM classes defined to Oracle Applications.

This profile option is visible and can be updated at site and application levels.

QP: Administer Public Queries

Default value: Null

Query saved in Public folders can be deleted or renamed only if the value of the profile QP: Administer Public Queries is set to Yes.

Values

- Yes: Enables queries saved in Public folders to be deleted or renamed.
- No: Queries saved in Public folders cannot be deleted or renamed.

QP: Allow Duplicate Modifiers

Default Value: Yes

Used by Basic Pricing Only. The profile option QP: Allow Duplicate Modifiers, which is typically set by the System Administrator, determines if duplicate modifiers are permitted. If set to Yes (the default), an existing modifier can be duplicated. If set to No, you must change the new modifier line before you can save it. A modifier line is considered a duplicate if any of the following attributes of the original and duplicated line match within the same modifier list:

- List Line Start Date Active
- List Line End Date Active (Lines with overlapping dates are considered duplicates)
- Modifier Level Code (Order/Line)
- Automatic Flag (Selected/Cleared)
- Product UOM code
- Product Attribute
- Product Attribute Value
- Pricing Attributes
- Set of Qualifiers

Values

- Yes: Duplicate modifiers can be copied within the same modifier list for Basic Pricing.
- No: Duplicate modifiers cannot be copied within the same modifier list for Basic Pricing.

This profile option is visible and can be updated at the site level.

QP: Blind Discount Option

Default value: Yes

The default value for this profile option should only be changed if you never define blind discounts. If you never define blind discounts, set this profile option to No; this bypasses part of the search engine processing. A blind discount is defined as a modifier that has all of the following:

- No list qualifiers on the modifier list header
- No line qualifiers on the modifier
- No products or pricing attributes

> **Note:** If your business must define a modifier as previously described, verify that this profile option is set to Yes. If this is not done, the modifiers will not be selected by the search engine.

Values

- Yes: Blind discounts are enabled.
- No: Blind discounts are disabled; bypass blind discount processing in search engine.

This profile option is visible and can be updated at the site and application levels.

QP: Build Attributes Mapping Options

Default value: No

When set to Yes, mapping rules can be generated for attributes used in active pricing setup. When set to No, it generates mapping rules for all the attributes.

The Build Attribute Mapping Rules program will source the attributes that are used only in the active pricing setup if the profile value QP: Check For Active Flag is set to Yes. If the profile value QP: Check For Active Flag is set to No, this program will source the attributes that are used in both the active and inactive setups.

This profile option is visible and can be updated only at the site level.

QP: Cross Order Volume Period1

Default value: Blank

This is required if you will be running the cross order volume load program. This defines the number of days of order lines that the load program will accumulate

and total. This value must not be the same as the value in QP: Cross Order Volume Period 2 or QP: Cross Order Volume Period 3.

Values

Always expressed in days.

This profile option is visible and can be updated at the site and application level.

QP: Cross Order Volume Period2

Default value: Blank

This is required if you will be running the cross order volume load program. This defines the number of days of order lines that the load program will accumulate and total. This value must not be the same as the value in QP: Cross Order Volume Period 1 or QP: Cross Order Volume Period 3.

Value

Always expressed in days.

This profile option is visible and can be updated at the site level.

QP: Cross Order Volume Period3

Default value: Blank

This is required if you run the cross order volume load program. This defines the number of days of order lines that the load program will accumulate and total. This value must not be the same as the value in QP: Cross Order Volume Period 1 or QP: Cross Order Volume Period 2.

Value

This value is always expressed in days.

This profile option is visible and can be updated at the site level.

QP: Custom Sourced

Default Value: No

Value

- Yes: When this profile is set to Yes, the Build Contexts program builds the contexts from the dynamic package generated by the Build Attribute Mapping Rules program and from the custom package created by the customers.

- No: When this profile is set to No, the Build Contexts program builds the contexts only from the dynamic package generated by the Build Attribute Mapping Rules program.

QP: Debug

Default value: Request Viewer Off

Value

- Request Viewer On: When set to on, the Request Viewer captures pricing request details into the pricing debug tables and debug log information into the debug log table. The debug log text file is also created.
- Request Viewer Off: When set to off, nothing is written into pricing debug tables and debug log table. The debug log text file will not be created.
- Request Viewer On, but Debug Log is not visible in Viewer: When this is set, the Request Viewer captures pricing request details into the pricing debug tables, but debug log information is not written into the debug log table. The debug log text file will be created.

Another profile option, QP: Set Request Name, can be used in conjunction with the QP: Debug profile option. When the QP: Set Request Name is set to Yes, the Request Name field will be prefixed with the Order ID.

This profile option can be updated at the user level and is active for the transactions of the user who set this Profile option—other users' transactions are not affected.

QP: Get Custom Price Customized

Default value: No

Indicates if, when processing formulas, the pricing engine evaluates the line type function. If your organization wants to use this formula line type, you must:

- Customize the GET_CUSTOM_PRICE function.
- Set this profile option to Yes.

Values

- Yes: When processing formulas, the pricing engine evaluates the line type function. This profile option must be set as Yes if the package body for QP_CUSTOM is coded.
- No: When processing formulas, the pricing engine does not evaluate the line type function.

This profile option can only be updated at the site level.

This profile is required if your business needs more pricing formulas than are provided in basic pricing.

A pricing formula consists of a formula expression (mathematical expression) consisting of step-numbers and formula lines that correspond to each step-number in the formula expression. Each formula line associates with a formula line type. There are six formula line types in Oracle Advanced Pricing (three in basic pricing). One type is function.

Oracle Advanced Pricing provides a function called Get_Custom_Price with a standard set of input parameters. The user is provided the flexibility of writing any custom code in the function body and use the input parameters supplied by the pricing engine. The value returned by the Get_Custom_Price function can be used in a formula expression.

To use the customized Get_Custom_Price function's return value in a formula, the user must set up a formula with a line type of function (the formula may or may not have other formula lines).

The Get_Custom_Price function can be used in any number of formulas at the same time because the formula ID is an input parameter to the function and assists the user differentiate code logic.

QP: Insert Formula Step Values into Temp Table

Default value: No

This profile option can be set only at the site level using the System Administrator responsibility.

Values

- Yes: If set to Yes, the step values of each formula attached to a price list line is evaluated by the Pricing Engine and inserted into the temporary table QP_FORMULA_STEP_VALUES_TMP. This can be referenced by customized code.
- No: If set to No, the step values are not inserted into the temporary table described above.

QP: Inventory Decimal Precision

Default value: 10

Used to set maximum decimal precision for uom conversion when calculating pricing quantity. If not set, it is defaulted to 10 digits decimal precision.

Example 1

Consider the following set up:

- Primary UOM = YR (year)
- Order UOM = MTH (month)
- Order Quantity = 12

The pricing engine rounds the pricing quantity based on the decimal precision setting. If the default precision is 10 digits, then the resulting pricing quantity will be 12 * (1/12) = 0.999999999999…the number will be rounded to 1YR.

Example 2

Consider the following set up:

- Primary UOM = DZ
- Order UOM = EA
- Order Quantity = 16

The pricing engine rounds the pricing quantity based on the decimal precision setting. If a user sets the Profile QP: Inventory Decimal Precision to 6 digits, then the resulting pricing quantity is calculated as follows: 16 * (1/12) = 1.33333333333333… which is rounded to 1.333333 DZ.

QP: Item Validation Organization

Default value: None

Set this profile, by site or responsibility, to an organization at the level in your organization hierarchy at which you set prices for items.

Values

This profile option is visible and can be updated at the site and responsibility levels.

Note: Before setting this profile option, you need to set up values for:

- HR: Security profile
- HR: Business Group profile options

Valid inventory master organizations will be available based on values of HRMS profile settings.

For more information on these profiles, see: Configuring, Reporting and System Administration in *Oracle HRMS*, Security chapter.

QP: Limit Exceed Action

Default value: Hard–Adjust Benefit Amount

This profile option defines the default action codes for promotion and modifier limits. It specifies the action for the pricing engine if a pricing request exceeds a promotional limit.

This profile option is based on the lookup type Limit Exceed Action.

Values

- Soft--Full Benefit Amount: Applies the full benefit to the order even if the transaction exceeds the defined limit.
- Hard--Adjust Benefit Amount: Adjusts the order benefit amount so that the order meets but does not exceed the promotional limit. A status message is sent to the calling application such as Order Management to place a promotional hold on the order.

This profile option is visible and can be updated site level using the System Administrator responsibility.

QP: Line Volume UOM Code

Default value: Blank

Required if your business must define qualifier rules which include the seeded qualifier line volume.

Specifies the unit of measure of the line volume qualifier. The attribute sourcing API converts the item on the request line to its primary UOM, then uses the volume attributes of the item to derive the line volume of the item in the UOM specified in the profile option.

Order Volume:

The Order Weight qualifier is calculated as the total volume of all the order lines for the order. In order to use this qualifier the following must be set up:

a. The profile, QP: Line Volume UOM Code, must be set to the correct volume uom code. This uom code is the uom for the total order volume.

b. There must be a uom conversion set up to convert from the ordered_uom for each order line to the volume uom specified in the profile. For example if line 1 of your order is in Ea and the profile is set to CBM, there must be a conversion set up from Ea to CBM for the item.

The calculation of Order Volume:

Each order line will be converted into the line volume (order_quantity *

uom_conversion_rate) and the line volumes will be totaled to get the Order Volume.

Values

All units of measure currently defined to Oracle.

This profile option is visible and can be updated at the site and application levels.

QP: Line Weight UOM Code

Default value: Blank

This is required if your business needs to define qualifier rules which include the seeded qualifier line weight.

Specifies the unit of measure of the line weight qualifier. The attribute sourcing API converts the item on the request line to its primary UOM, and then uses the weight attributes of the item to derive the line weight of the item in the UOM specified in this profile option.

The Order Weight qualifier is calculated as the total weight of all the order lines for the order. In order to use this qualifier the following must be set up:

1) The profile QP: Line Weight UOM Code must be set to the correct weight uom code. This uom code is the uom for the total order weight.

2) There must be a uom conversion set up to convert from the ordered_uom for each order line to the weight uom specified in the profile. For example if line 1 of your order is in KGM and the profile is set to LBS, there must be a conversion set up from KGM to LBS for the item.

How order weight is calculated: Each order line will be converted into the line weight (order_quantity * uom_conversion_rate) and the line weights will be totaled to get the Order Weight.

This setup is basically the same as the setup needed to use the Line Weight qualifier.

Values

All units of measure currently defined to Oracle.

This profile option is visible and can be updated at the site and application levels.

QP: Multi-Currency Installed

Default value: No

If you have global customers or do pricing in different currencies, the multi-currency feature enables you to maintain a single price list for multiple currencies.

Once the profile option is set to Yes, the concurrent program Update Price Lists with Multi-Currency Conversion Criteria must be run to enable the price list windows for multi-currency usage.

> **Warning:** Once the concurrent program has been run successfully, all existing price list and agreement windows are converted to multi-currency price lists. Users should not return to NON multi-currency price lists. Changing the profile option back to No may cause undesired results if conversion criteria have been used. Oracle does not support changing the setting back to No.
>
> The program must be run initially only once to activate the multi-currency feature; otherwise, data corruption may result.

Values

- Yes: If the profile QP: Multi-Currency-Installed is Yes, the incoming pricing request will send order currency, pricing date, product attributes, pricing attributes and qualifier attributes. The pricing engine matches the order currency with the price list's Base Currency or Conversion Criteria To-Currency
- No: When the profile QP: Multi-Currency-Installed is No, the order currency is matched with the price list base currency, which is the current behavior.

This profile option is visible and can be updated at the site and application level.

QP: Negative Pricing

Default value: No

The default value should only be changed if your business needs to define a negative price on a price list line. Controls whether a negative price can be entered in the Price List setup window.

Values

- Yes: Allows a negative price to be entered.
- No: Does not allow a negative price to be entered.

This profile option is visible and can be updated at the site and application levels.

QP: Pass Qualifiers to Get_Custom_Price API

Default value: Blank

Values

- Yes: If selected, then only the qualifiers are passed to Get_Custom_Price.
- No: If selected, qualifiers will not be passed to Get_Custom_limit.

QP: Price Rounding

Default value: Blank

This profile option controls how the value for the rounding factor is derived and used in price lists and related windows. The Advanced Pricing - Price Lists window rounds, stores and displays the list price based on the profile option setting.

The value entered in the Round To field from the price list is used to store the rounded value, while currency precision determines the displayed list price.

For example, if the Round To value is -2 and the currency precision is -5, the following list prices display:

115.24000

9.23000

100.00000

If the QP: Price Rounding profile option is set to Enforce Currency Precision, the value in the Round To field in the Advanced Pricing - Price Lists window cannot be updated. Also, the values permitted for the rounding factor will be limited to the price list currency precision.

The following table shows how different settings for QP: Price Rounding affect the list price (assume that the price list price = 6.15 and the markup change = 1.52% resulting in 6.24348).

Table 14–3 Rounding Example

If QP:Price Rounding value is: (see right)	Blank (Default)	Enforce Price List Rounding Factor	Enforce Currency Precision
Value in Round To field on Price List	-2	-2	-4
List Price	6.15	6.15	6.15
Mark up	1.52 %	1.52 %	1.52 %
List Price (new)	6.24348	6.24	6.2435

Values

- Blank (Default): If this option is selected, the following occurs:
  - No limit is imposed on the number of places that can be entered on the price list after the decimal point.
  - The value for a price list line is not rounded.
  - The list price that displays will not be rounded by either Currency Precision or the Round To value.
  - Static formula calculation results will not be rounded.
  - The Round To value specified for the price list rounds the pricing engine results.
- Enforce Price List Rounding Factor: If this option is selected, the value entered in the Round To field of the Advanced Pricing - Price Lists window is used for:
  - Rounding the value on the price list line.
  - Rounding the pricing engine calculation results.
  - Calculating the results for static formula calculations.

  The currency precision setting determines the display of the list price.
- Enforce Currency Precision: If selected, the Rounding Factor field on the price list cannot be updated by the user. Instead the Rounding Factor value defaults from the profile QP: Unit Price Precision Type (either Standard/Extended

precision) for the price list currency. The decimal places that display for the list price is determined by the Currency Precision.

Formula Prices: For dynamic formulas, the calling application passes the rounding factor and the resulting rounding factor displays regardless of the profile setting.

QP: Pricing Transaction Entity

Default value: Order Fulfillment

This profile option indicates the current Pricing Transaction Entity (PTE) in use. Only those contexts and attributes assigned to the current Pricing Transaction Entity will be available in the list of values on the setup forms. Likewise, querying up setup data for price list lines, modifiers, and qualifiers, etc. will cause the description to be shown only for those contexts and attributes that are assigned to the current Pricing Transaction Entity. Therefore it is important to set this profile option to the correct value before creating or querying any setup data in the Pricing Application.

It is not a good idea to changes this profile often as you will see the other context-attributes combinations for a different PTE when querying on the pricing setup forms. If this happens, you will see the internal ID code..

This profile can be set at the site, application, and user levels.

Values

The valid values for this profile are all the Pricing Transaction Entities that belong to QP Lookup type QP_PTE_TYPE.

The seeded value of this profile for various applications is shown in the table below.

Table 14–4 Seeded values for applications

Application name	Default Value
Site Level Profile Value	Order Fulfillment (ORDFUL)
Oracle Transportation (FTE)	Logistics (LOGSTX)
Oracle Inventory (INV)	Intercompany Transaction (INTCOM)
Oracle Demand Planning (MSD)	Demand Planning (DEMAND)
All other Applications	Defaulted from Site

QP: Promotional Limits Installed

Default value: No

This profile option activates the promotional limits feature in Oracle Advanced Pricing to enable users to manage promotional limits and related functions. The initial default value is N for No which means the limit feature is not active. The System Administrator must change this value to Y for Yes to be able to use limits. Only the System Administrator can change the value of this profile at site and application levels only. Other users can view the profile only.

Values

- Yes: Enables the promotional limits features to be used.
- No: Disables the promotional limits features.

> **Warning:** Once the QP: Promotional Limits Installed profile option is enabled, leave the promotional limits active with the value of Y. Do not disable the QP: Promotional Limits Installed profile option once it has been enabled!

QP: Qualify Secondary Price Lists

Default value: Yes

This profile option enables secondary price lists to be checked for qualifiers when the primary price list is not validated. If the profile is set to Yes an item on a non-validated line will not be picked up from secondary price lists when the primary price list is not validated.

Values

- Yes: Secondary price lists will be checked for qualifiers when the primary price list is not validated.
- No: Secondary price lists will not be checked for qualifiers when the primary price list is not validated.

QP: Return Manual Discounts

Default value: No

A modifier can be applied to the order automatically at the time of pricing or can be returned to the calling application (Oracle Order Management) and stored in price

adjustments table, so that users can choose adjustments to be applied on the order line.

The following modifiers types can be manual:

- Discounts
- Surcharges
- Price break lines
- Freight Charges

Manual modifiers can be set at all levels (line, group of lines, and order), and always be in the null bucket. Manual adjustments can be overrideable.

Values

- Yes: In basic pricing, *Y* is the default and should not be changed (this duplicates R11 functionality). One automatic discount is applied. All manual discounts are returned for user selection. All the automatic discounts that are deleted as part of incompatibility processing return as manual discounts available for user selection.
- No: All automatic and manual discounts run through incompatibility processing and one from each incompatibility group is returned. An automatic discount may be deleted and a manual discount may be selected. Discounts (automatic or manual) deleted as part of incompatibility processing are not returned as manual discounts.

QP: Satisfied Qualifiers Option

Default value: Yes

The profile option QP: Satisfied Qualifiers Option impacts performance when entering and booking an order. It controls whether satisfied qualifiers are returned to the calling application or not.

Values

Yes: When set to Yes (the default), the pricing engine returns all the satisfied qualifiers to the calling application. This increases pricing engine processing time.

No: When set to No, processing time is reduced because the pricing engine does not return the satisfied qualifiers to the calling applications.

QP: Security Control

Default value: Off

Controls the activation of the pricing security feature for your entire installation. It can be set On or Off. Before turning this profile option on, ensure that you have completed all the set up and implementation steps. For more information, see: Pricing Security, *Oracle Advanced Pricing Implementation Manual*.

Values

Off: After initial upgrading to security, the QP: Security Control is set to Off (the default) which maintains the pre-upgrade functionality. This means any user with functional access to the Pricing Manager responsibility has full access to maintain and update all pricing entities regardless of operating unit.

On: The Security Control profile should not be turned ON until all mapping has been completed. All functional users require access privileges to view or maintain their pricing entities.

When the profile QP: Security Control is turned on, each newly created price list and modifier form will be assigned a unique system-generated operating unit identifier.

QP: Security Default Maintain Privilege

Default value: Global

Controls the default maintain privileges for NEWLY CREATED price lists and modifiers after security is turned on. For example, if the profile option is set to Operating Unit, then the maintain privileges for that price list or modifier are restricted to the operating unit where the price list or modifier was created. This controls which users (if any) have access to maintain specific price lists and modifiers.

Values

Global (Default), Operating Unit, Responsibility, User, or None.

QP: Security Default ViewOnly Privilege

Default value: Global

Determines the default view-only privileges for NEWLY CREATED price lists and modifiers after security is turned on. View and maintain responsibilities are controlled separately by different profile options. This profile option enables you to set the view-only privileges at one of the following levels: Global (Default),

Operating Unit, Responsibility, User, or None. This controls which users (if any) have access to view specific price lists and modifiers.

Values

Global (Default), Operating Unit, Responsibility, User, or None.

QP: Selling Price Rounding Options

Default value: Individual: = round(listprice) + round(adj)

This rounding option rounds the selling price after adding unrounded list price and adjustments: selling price = round (list price + adjustments)

> **Note:** The profile OM: Round Unit Selling Price has been migrated to QP: Selling Price Rounding Options.

Values

- NO: = unrounded listprice + unrounded adjustments: No rounding.
- Individual: = round(listprice) + round(adj): Rounds selling price and adjustments.
- Additive: =round(listprice + adj); unrounded Freight: Rounds selling price after adding unrounded list price and adjustments.

This profile option can b e viewed and updated at the site level.

Freight Charge Rounding: If the QP: Selling Price Rounding Options profile is set to NO or ADDITIVE then freight charges will not be rounded. If the profile is set to INDIVIDUAL then freight charges will be rounded. The rounding flag in the control record passed by calling application may have one of the following values:

- Y (Yes): Rounds selling price and adjustments.
- N (No): No rounding.
- Q: Behavior depends on the profile setting for QP: Selling Price Rounding (NO, INDIVIDUAL, ADDITIVE). If rounding flag is passed as Q, but QP: Selling Price Rounding Options is NULL, the default behavior is to round selling price and adjustments.
- Null: Rounds selling price and adjustments.

Case 1)

Rounding Flag = Q

Profile QP: Selling Price Rounding Options = NO

List Price = 12.60, Rounding Factor = 0, Discount 25%

Adjustment amount = -3.15

Selling price = 12.60-3.15=9.45

Case 2)

Rounding Flag = Q

Profile QP: Selling Price Rounding Options = INDIVIDUAL

List Price = 12.60, Rounding Factor = 0, Discount 25%.

Adjustment amount = -round (round(12.60)*0.25) = -3

Selling price = round(12.60) - 3 = 10

Case 3)

Rounding Flag = Q

Profile QP: Selling Price Rounding Options = ADDITIVE

List Price = 12.60, Rounding Factor = 0, Discount 25%.

Adjustment amount =-3.15

Selling price = -round(12.60 - 3.15) = 9

Case 4)

Rounding Flag = N

List Price = 12.60, Rounding Factor = 0, Discount 25%.

Adjustment amount = -3.15

Selling price = 12.60-3.15=9.45

Case 5)

Rounding Flag = Y

List Price = 12.60, Rounding Factor = 0, Discount 25%.

Adjustment amount = -round (round(12.60)*0.25) = -3

Selling price = round (12.60) - 3 = 10

Case 6)

Rounding Flag = NULL

List Price = 12.60, Rounding Factor = 0, Discount 25%

Adjustment Amount = -round (round(12.60)*0.25) = -3

Selling price = round(12.60) - 3 = 10

QP: Set Request Name

Default value: Blank

Values

Any valid values such as the Name or User ID of the user submitting the price request.

The profile option QP: Set Request Name can be used in conjunction with the QP: Debug profile option so that the Request Name field will be prefixed with the Order ID.

The QP: Set Request Name profile option is visible and can be updated at the site, application, responsibility, and user levels.

QP: Source System Code

Default value: Oracle Pricing

This profile option is used in all pricing setup windows to identify the application through which the pricing information is being entered. This source system code is held on all price and modifier lists to identify the origin of the data. At the time of pricing, the pricing engine may restrict its search to pricing information which originated from a particular application depending on the request type to source system setup.

By default, the value of the profile option QP_SOURCE_SYSTEM_CODE is QP (Oracle Pricing) at the Site level. However, to differentiate between applications and prevent update of modifiers among several applications, the value of this profile can be set up at the Application level.

If modifiers are created in different applications such as Oracle Advanced Pricing, Order Management, or Trade Management, then changes to the modifier can only be made in the application where the modifier was originally created. These changes include changes to the modifier header (as deleting is already prevented) or changes such as inserting, updating, or deleting modifier lines, pricing attributes, and qualifiers.

If you do not want to differentiate between applications or prevent update of modifiers created by different applications, then no specific setup is required. In this case, all modifiers are created with same value for source system code (assuming the value of this profile is set only at site level by default or same value if set for all applications which create modifiers).

Example

The following customer requirements need to be established when setting up the profile options:

- Modifiers created by the Trade Management application can only be changed by Trade Management.
- Modifiers created by Oracle Pricing and Order Management applications can be interchangeably updated but not by other applications.
- All other applications can update the modifiers of other applications except Trade Management, Oracle Pricing and Order Management.

To accommodate these requirements, the value of the profile QP_SOURCE_SYSTEM_CODE is set up as follows:

- Site level: QP (default)
- Trade Management: XXX
- Oracle Pricing: YYY
- Order Management: YYY (same as Oracle Pricing)

The default value should only be changed if the source of the pricing data is any application other than Oracle Advanced Pricing. Set this to the source system lookup code of the application which is interfacing the pricing data.

After Attribute Mapping Manger is installed, this profile can be set at Site, Application and User level. The default value for this profile at the time of installation at the Site level is Oracle Pricing. The seeded value of this profile for various applications is shown in the table below:

Table 14–5 Seeded values for source system applications

Application name	Default Value
Site Level Profile Value	Oracle Pricing (QP)
Oracle Transportation (FTE)	Oracle Transportation (FTE)
Oracle Inventory (INV)	Oracle Inventory (INV)

Table 14–5 Seeded values for source system applications

Application name	Default Value
Oracle Marketing (AMS)	Oracle Marketing (AMS)
All other Applications	Defaulted from Site

Note: Set the QP: Source System Code Profile to the source system lookup code of the application from which the QP Setup or other application setup windows are called.

Values

QP: Oracle Pricing (and other source system lookup codes of related applications).

This profile option can be updated at the site and responsibility levels.

QP: Time UOM Conversion

Default value: QP: Oracle Pricing

This profile option enforces the unit of measure (UOM) conversion for price lists when the Primary UOM box in a price list is selected. For example, suppose that in a price list, the UOM selected is EA (each) and that the Primary UOM box is selected. (The primary uom in the price list could be different from the primary uom set in Oracle Inventory for the item.)

If an order line is entered with the unit of measure as Dozen, the pricing engine would convert the order quantity to EA and provide a price in EA This occurs only if no eligible price lists are found with Dozen as the unit of measure. To do the conversion, Advanced Pricing uses the inventory UOM conversion APIs.

As for time periods, the inventory UOM conversion uses a static conversion factor of 30 days per month. Since days are set as the base UOM, one year is converted to 12.16666… months by the inventory uom APIs. This result is obtained by dividing 365 days by 30 days rather than one year being converted to 12 months.

For Oracle Contracts, if the price list does not calculate correctly when the Period type = Year, confirm that the UOM conversion for month = 730. Otherwise, when the QP: Time UOM Conversion profile is set to Standard, pricing takes the duration into consideration but converts the hours for 12 months to be 12.16666666666666666666666667 months. When the QP: Time UOM Conversion profile is set to Oracle Contracts, the duration is not considered in the pricing.

To confirm that month = 730, confirm the setup in Oracle Contracts: SetUp > Contract > UOM > UOM Classes > Unit of Measure Classes > select Time. (Responsibility: Service Contracts Manager or Corporate Contracts Manager). The following is an example of the Unit of Measure Conversions you may have set up for Time:

- Day = 24 Hours
- Hour = 1 Hour
- Minute = .016667 hours
- Month = 730 Hours
- Quarter = 2190 Hours
- Week = 168 Hours
- Year = 8760 Hours

Values

- Standard: When set to Standard, the existing method of passing the uom quantity to the pricing engine API can be used by the calling applications.
- Oracle Contracts: When set to Oracle Contracts, the pricing engine use the APIs provided by Oracle Contracts to convert the date and time. If the Oracle Contracts API does the uom conversion, the following values from each request line processed need to be passed by the calling applications (such as Order Management, Order Capture or Oracle Contracts) to the pricing engine:
  - Period Start/End Dates
  - Ordered UOM (Time uom in which the price needs to be fetched)
  - Ordered quantity

QP: Unit Price Precision Type

Default value: Standard

This profile option changes the default value if you need to round to the currencies extended precision.

Used to determine the value for the rounding factor which is defaulted on the price list. The rounding factor is limited by the number of positions allowed in the standard or extended precision format of the price list currency.

Note: If you update either the extended or standard precision value for a currency, then the updated value is applied only to new price lists but not existing price lists.

Values

- Extended: Rounding factor defaults to the currency's extended precision.
- Standard: Rounding factor defaults to the currency's standard precision.

This profile option can be updated at the site and application levels.

QP: Valueset Lookup Filter

Default value: Yes

Use this profile option to enable or disable a search criteria window for qualifier value lookups in qualifiers, price lists, and modifiers. Some qualifiers use large valuesets, for example, those based on all customers, and searches may take a long time. If you want to reduce the number of items that display in the list of values, you can enter search criteria. If you do not enter search criteria and click the list of values indicator for the fields Value From or Value To, you see a window which advises that you have not entered search criteria and that the search may take a long time.

Values

- Yes: The message displays.
- No: The message does not display. Use this value is you do not expect to have large qualifier valuesets and do not need to enter search criteria to reduce the display.

This profile option is visible and can be updated at the site, application, responsibility, and user levels.

QP: Verify GSA Violations

Default value: No

Change the default value to Yes if you require GSA pricing functionality. This profile option indicates whether the pricing calculation engine should test for GSA violations. The evaluation is performed if a request is for a non-GSA customer, and GSA rules are violated because the selling price of an item is calculated to be less than the price of the item on any GSA price list.

Values

- Yes: Pricing calculation engine tests for GSA violations; any violating request lines are returned to the calling application with GSA violation status.
- No: Does not test for GSA violations.

This profile option can be updated at the site, application, responsibility, and user levels.

15

Integrating with Oracle Advanced Pricing

This chapter provides information about integrating with Oracle Advanced Pricing. The following topics are discussed:

Overview

Oracle Advanced Pricing is an API-based engine that can be called by any integrating application. This chapter provides certain essential information needed for the successful integration with Advanced Pricing.

For more information, see:

- *Oracle Advanced Pricing Implementation Manual,* Attributes Mapping and Technical Considerations
- *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual* for Pricing API and example scripts
- *Oracle Advanced Pricing User's Guide*

Integration Steps Required for Pricing

The following steps describe the process required to integrate with Oracle Pricing.

1) Determine your end to end pricing business needs.

a. See the *Oracle Advanced Pricing User's Guide* and the *Oracle Advanced Pricing Implementation Manual* to determine which pricing features need to be supported by your application.

b. Evaluate your transaction variations and choose the pricing events to be called at what phase of your transaction cycle. To create new pricing phases or learn more about pricing phases and events, see: *Oracle Advanced Pricing User's Guide*.

c. Determine your business need to choose a pricing transaction entity (PTE). If a new PTE is required, see: Defining Pricing Transaction Entity on page 10-8 to learn about how to create a PTE. When you create a new PTE, the seeded contexts and attributes are not available. New contexts and attributes need to be created for the newly created PTE, or existing attributes need to be linked to the PTE.

d. The pricing engine needs to be called with a request type. Each request type is linked to a PTE. Request type determines two things: the price lists and modifiers to be searched during pricing engine call and the attribute mapping rules to be run when build_context API is called. Note that each modifier/price list is associated with a source system and hence with a

PTE. The pricing engine will look at modifiers or price lists belonging to the source systems linked with the PTE.

e. Attribute Mapping rules are seeded by individual transaction systems for their respective PTEs. The PTE is linked to a set of request types and source systems. Determine if your transaction data corresponds to the attribute mapping rules defined already. If not, refer to attribute mapping to learn how to define a custom mapping.

2) Determine your pl/sql global structure.

Before calling the API to build the contexts, it is necessary to populate a global record structure corresponding to the request type with the information pertaining to the summary line (order header) or the request line (order line) for which the qualifier/pricing attribute information needs to be built. Request type ONT uses a global structure oe_order_pub.g_line for order line and oe_order_pub.g_hdr for order header. The request type ASO uses a global structure aso_pricing_int.g_line_rec and aso_pricing_int.g_header_rec.

For more information on the oe_order_pub.g_line/g_hdr structures and the Order Capture API doc for the aso_pricing_int.g_line_rec and aso_pricing_int.g_header_rec, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*.

Determine if your request can be mapped to one of these structures. Attribute mapping rules need to be defined specific to the structure that will be used by the calling application. If you find that your request structure cannot be mapped to one of these, then you need to define a new structure and write attribute mapping rules using this new structure.

3) Determine the tables to hold the adjustment information.

If you need to hold the information returned by pricing engine, for showing the breakup of all the benefits given to your order/quote/pricing request, determine if you can use the price adjustment tables used by Order Management/Order Capture.

`OE_PRICE_ADJUSTMENTS/ASO_PRICE_ADJUSTMENTS`

Holds the price adjustments

`OE_PRICE_ADJ_ATTRIBS/ASO_PRICE_ADJ_ATTRIBS`

Holds the qualification conditions applied to give the adjustments

```
OE__PRICE_ADJ_ASSOCS/ ASO_PRICE_ADJ_RELATIONSHIPS -
```

Holds relationships between multiple adjustment records for price breaks, promotional goods and other item benefits

If you need to create your own tables, please use the table definition of the above tables as a guideline and create your tables.

4) Populate the pricing request structures.

The following is the API structure:

QP_PREQ_PUB.PRICE_REQUEST

```
(p_control_rec IN
QP_PREQ_GRP.CONTROL_RECORD_TYPE,
x_return_status OUT VARCHAR2,
x_return_status_text OUT VARCHAR2
);
```

a. Populate the control record p_control_rec.

For details on the parameters of the control record, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*, Pricing APIs. The control record determines the way the pricing engine returns the price.

b. Call the API: QP_Price_Request_Context.Set_Request_Id();

Set the pricing request_id to enable the pricing engine to identify the data in the pricing temporary tables that belong to the current pricing engine call. The pricing engine will not delete the temporary table data before each pricing engine call. Instead, the data from the previous pricing engine call may remain in the pricing temporary tables. The calling application needs to set the request_id each time the pricing engine call is made. This will be the first step.

The API: QP_Price_Request_Context.Set_Request_Id() callsets a unique request_id for every pricing engine call made in the same session without commits or rollbacks. This negates the need to delete the data in the pricing temporary tables between calls. Also, if the request_id is not set, then the calling application needs to delete the data in the pricing temporary tables before making the next pricing engine call without a commit or rollback. Remember that a commit or rollback purges the data in the temporary tables so that the records do not need to be deleted.

c. Populate the global structure used by the attribute mapping. For example if you are using request type ONT, populate the PL/SQL structure OE_ORDER_PUB.G_LINE.

d. Call the API: QP_ATTR_MAPPING_PUB.Build_contexts

```
QP_Attribute_Mapping_PUB.Build_Contexts
(          p_request_type_code    IN      VARCHAR2,
    p_line_index                  IN      NUMBER,
    p_pricing_type_code           IN      VARCHAR2
);
```

This API looks at the public record structures listed in the attribute mapping rule. So, it is necessary that the request line information for each line is loaded into the request structure and the API is called for each line. The build context API inserts the mapped attributes into the pricing temporary tables.

If there are no attribute mapping rules or record structure, the qualifiers and pricing attributes can inserted into the qp_preq_line_attrs_tmp. For example, if you want the pricing engine to fetch the price from a particular price list, pass the price list in QP_PREQ_QUAL_TMP with qualifier_context=MODLIST.

qualifier_attribute=QUALIFIER_ATTRIBUTE4 and qualifier_attr_value_ from holds the price_list_id. Set the validated_flag to Y, if you do not want the pricing engine to check for the qualifiers.

Remember that the inventory_item_id is passed in the temporary table qp_preq_line_attrs_tmp with pricing_context=ITEM pricing_ attribute=PRICING_ATTRIBUTE1 and pricing_attr_value_from holds the invetory_item_id.

The build context API needs to be called with a p_pricing_type_code of L for the request lines.

e. For more information on the Copy_Line_to_request to load the request lines, see: Sample Code using Order Management Structure. Call the API QP_PREQ_GRP.Insert_Lines2 to insert the request lines to the pricing temporary table qp_preq_lines_tmp.

For more information on API structures and parameter descriptions, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*.

f. Append any user-entered pricing attributes and Asked for promotions/deals or coupons to the temporary table qp_preq_line_

attrs_tmp. Set the validated_flag of qp_preq_line_attrs_tmp to Y for "Asked For" promotions, if you do not want pricing to check for the qualifications , before applying the promotion.

For more information on the Append_ask_for procedure, see: Sample Code using Order Management Structure on page 25. These attributes can be inserted into the temporary table qp_preq_line_attrs_tmp using the API QP_PREQ_GRP.insert_line_attrs2.

For more information API structures and parameter descriptions, see: *Oracle Order Management Suite APIs and Open Interfaces Manual*.

g. If there are any manual adjustments to be applied or you need engine to consider any other adjustments , those records can be inserted into the pricing temporary table qp_preq_ldets_tmp with the applied_flag and updated_flag set to Y.

h. Ensure that the Summary Line is populated.

Every request to the pricing engine must contain a summary record in qp_preq_lines_tmp with

information from order header. The pricing engine will apply the order level modifiers against the summary line passed. If the summary line is not passed, the pricing engine may not apply any order level adjustments.

Set the line_type_code to ORDER for the summary line. (Repeat step# 4.3.1 through 4.3.6 for order header). Use p_pricing_type=H while calling QP_ATTR_MAPPING_PUB.build_context().

For more information on how to populate the summary record using copy_Header_to_request(), see: Sample Code using Order Management Structure on page 25.

i. For Service Lines with Percentage Price, ensure that the Parent Line is passed. For more information on how to price service items, see: Service Item Pricing on page 15-18.

5) Call Price_request()

Call the API:

```
QP_PREQ_PUB.PRICE_REQUEST
(p_control_rec IN QP_PREQ_GRP.CONTROL_RECORD_TYPE,
x_return_status OUT VARCHAR2,
x_return_status_text OUT VARCHAR2);
```

6) Interpreting the results of price request.

a. Handling errors:

Pricing engine can return hard errors and soft errors. The pricing engine call is a success if the value of x_return_status is FND_API.G_RET_STS_ SUCCESS.

The soft errors can indicates the line level exceptions while pricing. These errors are populated in qp_preq_lines_tmp.pricing_status_code. The following three are the success codes for line:

```
G_STATUS_NEW
G_STATUS_UPDATED;
G_STATUS_UNCHANGED
```

The following need some action from the calling application.

```
G_STATUS_DELETED
G_STATUS_TRANSIENT
G_STATUS_GROUPING
G_STATUS_INVALID_PRICE_LIST
G_STATUS_GSA_VIOLATION
G_STS_LHS_NOT_FOUND
G_STATUS_FORMULA_ERROR
G_STATUS_OTHER_ERRORS
G_STATUS_SYSTEM_GENERATED
G_STATUS_BEST_PRICE_EVAL
G_STATUS_INCOMP_LOGIC
G_STATUS_CALC_ERROR
G_STATUS_UOM_FAILURE
G_STATUS_PRIMARY_UOM_FLAG
G_STATUS_OTHER_ITEM_BENEFITS
G_STATUS_INVALID_UOM
G_STATUS_DUP_PRICE_LIST
G_STATUS_INVALID_UOM_CONV
G_STATUS_INVALID_INCOMP
G_STATUS_BEST_PRICE_EVAL_ERROR
G_STATUS_LIMIT_HOLD
G_STATUS_LIMIT_EXCEEDED
G_STATUS_LIMIT_ADJUSTED
G_STATUS_LIMIT_CONSUMED
```

The GSA violation G_STATUS_GSA_VIOLATION is a special case. Refer to the GSA Violation topic in the following section detailing the GSA Violation behavior in detail.

For more information on the different status codes, see: *Oracle Advanced Pricing Implementation Manual*, Troubleshooting.

b. Price List fetched for an request line.

The price_list_id is populated on the price_list_id column in the qp_preq_lines_tmp.

The List price (undiscounted base price) is returned in unit_price and the discounted price (after applying all the discounts/surcharges) is in adjusted_unit_price. Remember that these are per unit price expressed in unit of measure pricing_uom_code. Pricing_uom_code could be different from the line_uom_code (Ordered UOM). That means if the price list is set in a unit of measure EACH and has been marked as primary, and if the order is in Dozen and there is no price list line for Dozen, pricing engine would return the price in EACH.

Priced_quantity holds the line_quantity (Order Line quantity) expressed in pricing_uom_code. If the price list line has a percent price set, the percentage is returned in percent_price and the price of the parent line used in price calculation is in parent_price.

c. Modifiers fetched for an order line.

The modifiers/adjustments are returned in the temporary table qp_preq_ldets_tmp. Pricing has a view qp_ldets_v which the calling application needs to look at to insert/update the adjustment details into the transaction tables. If the created_from_list_line_type for this record is of type OID (Other Item Discounts), PRG (Promotional Goods) CIE (Coupon Issue), PBH (Price Breaks), the Qp_preq_rtd_lines_tmp would hold the relationship. For example, you can find the new line created for a free good offer by looking for an adjustment line of type PRG in qp_ldets_v. then look for the line_detail_index in qp_preq_rltd_lines_tmp and fetch related_line_detail_index. Now search qp_ldets_v, with line_detail_index = related_line_detail_index. The line_index corresponding to this would be the Free good line.

If you find a record with qp_ldets_v.list_line_type_code = IUE, there is free upgrade of the item given to the order line.

If qp_ldets_v.list_line_type_code=TSN, the adjustment is of type terms upgrade.

If qp_ldets_v.accrual_flag is set to Y, the adjustment is of type accrual is not included in calculating the adjusted_unit_price. Benefit_qty is populated for non-monetary accruals.

All the adjustments with automatic_flag=Y have been applied by the engine along with any manual adjustments with automatic_flag=N passed in by the user and having applied_flag=Y and updated_flag=Y.

```
qp_ldets_v .operand_calculation_code holds qp_list_lines.arithmetic_
operator (%,AMT,LUMPSUM,NEWPRICE).
```

The value of this is held in qp_ldets_v.operand_value. The $ value of operand_value is contained in qp_ldets_v.adjustment_amount. This adjustment_amount is per unit expressed in pricing unit of measure (UOM).

Freight charges have qp_ldets_v.list_line_type_code=FREIGHT CHARGE and they have charge_type_code and charge_subtype_code populated. At present, pricing returns only one freight charge (of highest monetary value) for a charge_type_code and sub_type_code combination.

d. Qualifiers and Attributes

The table qp_preq_qual_tmp holds the qualifiers that were matched for a price adjustment. This can help in tracking why a particular adjustment was given for any given order line. The structure qp_preq_line_attrs_tmp holds the product and pricing attributes matched for an adjustment record.

Pricing Engine Interaction Details

This section provides an overview of the features supported by the pricing engine and how the pricing engine processes them. This information includes what the pricing engine returns for each feature and how the request information needs to be passed on subsequent calls for the same request lines to get the same results back again.

Passing adjustments/modifiers to the pricing engine

If the calling application needs to apply specific modifiers to the pricing engine in order for it to apply them against a request line, it needs to be inserted into qp_preq_ldets_tmp with pricing_status_code = QP_PREQ_GRP.G_STATUS_UNCHANGED.

Manual adjustments

All the automatic modifiers (automatic_flag = Y) of type Discounts and Surcharges that the user has qualified for, that are deleted as part of incompatibility resolution (due to incompatibility setup rules), are returned as manual discounts to the calling application if the profile QP: Return Manual Adjustments is set to Y. In addition to these discounts, all the qualified manual modifiers of type Discounts. Surcharge discounts are also returned to the calling application, unapplied.

Applying manual adjustments:

Manual adjustments can be applied in two ways:

- The calling application can pass the manual adjustment to the pricing engine with applied_Flag = Y and updated_Flag = Y. The pricing engine will apply this manual adjustment. The calling application can override the manual adjustment by passing the new operand qp_preq_ldets_tmp.operand_value. Manual adjustments can be passed to the pricing engine by inserting the adjustment into qp_preq_ldets_tmp against the request line to which it needs to be applied.

  For example, the calling application makes a pricing engine call with 3 request lines. He wants to apply a manual adjustment of 10% to be applied to the second request line. Then, the calling application should pass the manual adjustment in the line_detail_tbl with columns updated_flag = Y and applied_flag = Y and with the line_index of the second request line. The pricing engine API, will calculate the adjustment amount and will apply this

manual adjustment to the second order line. The applied_flag and updated_ flag will be returned as Y to indicate that it has been applied.

- The calling application can override the selling price by passing the new selling price in the qp_preq_lines_tmp.updated_adjusted_unit_price. The engine will then pick up a suitable manual overrideable modifier that the request line has been qualified for and back-calculate the adjustment amount and operand to match the new selling price. In this case the pricing engine will pass back this manual modifier with calculation_code as BACK_ CALCULATE, updated_flag = Y and applied_flag = Y. The back calculation is done after applying all the automatic and overridden manual adjustments. So the calculated selling price will reflect the desired selling price.

  For example, if the calling application passes 3 request lines to the pricing engine and the unit selling price on the second request line is $80 and the unit list price is $100 and he wants to override the selling price to $90. In this case, the user has to pass 80 in qp_preq_lines_tmp.updated_adjusted_unit_ price on the request line in the record corresponding to the second request line. In this case, the pricing engine applies all the automatic adjustments and any manual overridden adjustments passed by the user.

  The pricing engine registers that the selling price has been overridden. It picks up all the qualified manual overrideable adjustments, decides whether it needs to apply a discount or a surcharge. It prefers a manual overrideable adjustment that has been applied already. If none has been applied already, it randomly picks up a manual overrideable surcharge, back calculates the adjustment amount, $10 surcharge in this case, and returns it with calculation_code as BACK_CALCULATE.

  If there are no surcharge adjustment, it tries to apply a manual overrideable discount adjustment. If there are no qualified manual overrideable adjustments, it returns an error status on the second request line and the pricing_status_text indicates that there are no manual adjustments. In case, there is an error during the back calculation, the engine returns an error status on the second request line. The pricing_status_code on the second request line has the error code QP_PREQ_PUB.G_BACK_CALCULATION_ STS in case of an error.

Deleting manual adjustments that are applied: To delete manual adjustments that are applied, they need to be deleted from the transaction table on the calling application's side which stores the applied adjustments returned by the pricing engine. This means that the calling application also needs to delete the attributes and the associations (relationships) for the adjustment. In case the manual

adjustment is a price break, the child lines of the price break, the attributes of the price break and the child lines and the relationships between the price break and the child lines also need to be deleted.

Once they are deleted from the transaction tables, they will not get passed to the pricing engine in any consequent calls to the pricing engine and will not get applied.

Deleting automatic overridden adjustments that are applied

In case your business supports deleting automatic overridden adjustments, the calling application needs to insert these adjustments as applied_flag = N and updated_flag = Y and these need to be passed to the pricing engine during any consequent pricing engine calls. The pricing engine will not apply this automatic adjustment even if the request line qualifies for it.

Apply automatic overrideable modifiers

In order to apply automatic overrideable modifiers, pass the modifiers to pricing engine by inserting the modifier into qp_preq_ldets_tmp with updated_flag = Y and applied_flag = Y and pricing_status_code = QP_PREQ_GRP.G_STATUS_ UNCHANGED. The pricing engine will apply this modifier.

Manual/Overrideable price breaks

To apply manual overrideable price breaks, pass the price break header adjustment and its child lines with the relationships between the price break modifier and the child break lines with the overridden operand. The pricing engine evaluates the break and applies the overridden price breaks. Also, remember to pass the volume attributes that the pricing engine returned previously along with the price break modifier and the child lines. The volume attributes are necessary because the pricing engine does not look at the pricing setup to derive the item quantity/amount information on the price break modifier. To insert relationships, use the QP_PREQ_GRP.insert_rltd_lines2 API.

GSA Violation

Pricing setup allows users to create GSA price lists. GSA Violation occurs when the price of an item goes below the GSA price for the item for a non-GSA customer. The pricing engine checks for GSA violation at the end of the pricing engine call and sets the pricing_status_code on the request line to QP_PREQ_ GRP.G_STATUS_GSA_VIOLATION in case of a GSA Violation. The calling application may handle the GSA violation appropriately, which means that a warning or an error message may be raised. In case of a GSA customer, the

pricing engine gives the price on the GSA price list. This request line can qualify for further discounts for the GSA customer.

The pricing engine does GSA violation checks only if the GSA_CHECK_FLAG on the control record is passed as Y and if the GSA qualifier (Context = CUSTOMER, Attribute = QUALIFIER_ATTRIBUTE15) is passed on the request line with a value N indicating that the customer is a non-GSA customer. The attribute mapping API returns a GSA qualifier based on the gsa_indicator flag checked on the customer in AR or the invoice location has the gsa_indicator set to Y.

In OM, if the GSA Violation profile is set to Warning, a warning message is raised. In case it is set to Error, the application throws an error message.

Bucketing

The pricing engine applies bucketing rules after getting all modifiers to calculate the unit selling price.

Pricing formulas

The formulas are processed and the operand is evaluated based on the attributes or factors passed to the pricing engine.

Rounding

The rounding refers to the rounding of the list/selling price. Rounding is controlled by the rounding_flag on the control_record which is input to the pricing engine and the value of the profile QP: Selling Price Rounding Options. For more information, see: *Oracle Advanced Pricing Implementation Manual,* Profile Options and for information on the values for the rounding flag, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*.

Freight Charges

The freight charges that the order line qualified for are evaluated and the pricing engine applies the maximum freight charge for each charge type(charge_type_ code and charge_subtype_code) for request line. The freight charge does not affect the selling price. Manual/Overridden charges that need to be applied, need to be passed to the pricing engine with applied_flag set to Y and updated_ flag set to Y just like any manual adjustment (discount/surcharge).

Coupon Issue

When a coupon is issued, the engine generates a unique coupon number. The pricing engine inserts a record into QP_COUPONS table with this unique coupon number. This number may be displayed in the list to show the available coupons for the user to pick. This number is unique. This is the number that can be used to redeem the discount at a later time. One coupon number equals one redemption. Once the coupon number is used (redeemed) it cannot be used again. The user should know the coupon number they were given in order to redeem the coupon.

The next time an order is placed, if the user enters the coupon number, the engine qualifies the discount the coupon is eligible for and applies the discount. The coupon is deleted once the coupon has been redeemed.

The coupons are stored in QP_COUPONS table and are marked redeemed when they are redeemed.

In order to redeem a coupon, the coupon needs to be passed as a qualifier to the order line to the pricing engine (context = MODLIST, attribute =QUALIFIER_ATTRIBUTE3, value = <coupon number>). The coupon may be stored as an attribute of the line along with the ask_for_promotions. In order for the pricing engine to give the coupon discount on consequent reprice calls, the coupon needs to be passed as an attribute.

It is possible that the Coupon Issue modifier has qualifiers in the setup. When the coupon is redeemed, the pricing engine does a qualification check to see if all qualifiers are passed and if the qualifiers are not passed, the pricing engine does not qualify the request line for the modifier associated with the coupon. In order to prevent the pricing engine to do this qualification check, pass a value of Y to the validated_flag in the qp_preq_line_attrs_tmp against this coupon qualifier record.

In OM, in case an order qualifies for a coupon, when the order is cancelled or if the item is returned, the coupon is not deleted. Also during a reprice, the pricing engine keeps issuing new coupons.

Item Upgrade

If an order line qualifies for an item upgrade, the item upgrade modifier is applied against that line and can be found in the qp_ldets_v temporary table. In the temp table qp_ldets_v, the column related_item_id has the inventory_item_id of the upgraded item and the column inventory_item_id has the inventory_item_id of the original item. The calling application can replace the originally ordered item on the line with the upgraded item. During reprice, the calling

application needs to pass back the original item on the line so that the order is repriced the same way. This can be done by loading at the original inventory_item_id from the item upgrade modifier to the public record structure before attribute mapping.

In order to setup an item upgrade modifier, there needs to be a item relationship defined between the buy item and the upgrade item in Inventory item relationships with relationship type of Promotional Upgrade.

Promotional Goods

If an order line qualifies for a promotional goods modifier (PRG), the pricing engine creates a new order line for the free good line. The PRG usually is setup as buy item A get item B at x% off. In this case, if item A is ordered and if it qualifies for the PRG, the pricing engine creates the new order line with item B and it also applies the x% discount to this new line. In case of buy 1 get 1 free, this discount is 100%.

The new line that has been created by the pricing engine can be identified by the value of processed_code in the temp table qp_preq_lines_tmp which is set to QP_PREQ_GRP.G_BY_ENGINE.

The calling application needs to create a new order line to represent the free good line. Remember that the free good item gets passed as an attribute in the qp_preq_line_attrs_tmp to that line. The discount on the free good line gets passed in the temp table qp_ldets_v. The order line which qualified for the PRG modifier is the parent modifier and the discount on the free good line is the child line.

The engine creates a parent child relationship in the temp table qp_preq_rltd_lines_tmp with relationship_type_code as QP_PREQ_GRP.G_GENERATED_LINE, the line_detail_index is the line_detail_index of the parent line and related_line_detail_index is the line_detail_index of the child line. The above information returned by the pricing engine needs to be stored by the calling application in their transaction tables.

During a reprice, the pricing engine public API QP_PREQ_PUB compares the existing free good line and the promotional goods modifier in the pricing setup. If the promotional goods modifier has not changed, it populates a value of QP_PREQ_GRP.G_STATUS_UNCHANGED on the column process_status on qp_preq_lines_tmp. In this case, the calling application need not make any changes to the free good line. If the pricing setup has changed, then the process_status is populated with a value QP_PREQ_GRP.G_STATUS_UPDATED. For a fresh pricing engine call where one of the request lines qualify for the promotional goods line,

the new line is created in qp_preq_lines_tmp with process_status = QP_PREQ_GRP.G_STATUS_NEW. In this case, the calling application may insert this line into their transaction system.

When the free good line is created with process_status = QP_PREQ_GRP.G_STATUS_NEW/G_STATUS_UPDATED, the calling application needs to make another call to calculate freight charges for the free good lines. For more information on the implicit call for freight charges for promotional goods, see: Freight Charges for the free good line below.

In cases where users do not want the free good line, they can remove it from the order. To make the pricing engine recognize that it does not need to create the free good again, complete the following steps:

1. Delete the free good line, its attributes, its adjustments and the associations between the buy line and the free good line from the system. Retain the PRG adjustment on the buy line, but mark it as updated by setting the updated_flag = QP_PREQ_GRP.G_YES.
2. If some other free good line results because of the same PRG (if the promotional goods modifier is set up as buy A, get B and C free) on the order/quote, mark the discounts against the remaining free good lines as updated by setting the updated_flag = Y.

Now the pricing engine will not re-create the deleted freegood lines. The same approach needs to be followed if the calling application wants to substitute the free good item with some other item.

If the buy line is deleted, the calling application needs to delete all the free goods associated with this buy line, the adjustments on the buy line, the adjustments on the freegood lines and the associations between the buy line and the freegood line.

Freight Charges for the free good line

The pricing engine does not apply freight charges on the free good line. To get freight changes for the free good item, a second pricing engine call needs to be made which will insert just the free good line with price_flag QP_PREQ_GRP.G_PHASE and the pricing engine will just search for freight charges for the free good line. The freight charges do not affect the price on the free good line.

The reason why the pricing engine cannot apply the freight charges (if any) on the free good line is because, the pricing engine does not have any attributes mapped for the free good line to look for freight charges. The pricing engine

inserts the free good line with the pricing information and the product is the only attribute inserted against this line.

To get the freight charges for free good lines, the calling application needs to make another implicit call to the pricing engine. The freight call needs to be made only when there are lines in qp_preq_lines_tmp with process_status QP_PREQ_GRP.G_STATUS_NEW / G_STATUS_UPDATED. In this case, the calling application can make this implicit call to pass all the free good lines to the pricing engine.

If there are any line group-based modifiers, the order or quote can qualify for a free good line when the order or quote is repriced. For example, if there are two promotional goods modifiers, if the user adds A and then gets B free:

- PRG1: buy A, get B free
- PRG2: buy A and B, get C free

When freight charges for B are calculated, the pricing engine could have qualified the order or quote for the free good C if A had also been passed. For this to happen, the calling application needs to pass all the lines on the quote or order in the implicit call to evaluate the freight charges. Make sure the QP_UTIL_PUB.Get_Order_Line_status is called and pass all the lines only when the output of Get_Order_Line_Status passes all_lines_flag = QP_PREQ_GRP.G_YES; otherwise, pass the free good lines only. Get_Order_Line_Status passes all_lines_flag = QP_PREQ_GRP.G_YES if there are line group modifiers or other item discount modifiers. By passing all lines, we are making sure that the quote/order will qualify for all the applicable modifiers and that the price will not change on subsequent reprice.

For more information on QP_UTIL_PUB.Get_Order_Line_status, see:

Other Item Discount

Other Item discount (OID) behavior is very similar to the free goods or promotional goods as explained above except that the engine does not create a new order line. Other Item Discount is set up as buy item A and get x% discount on item B where both item A and item B are ordered.

This means, to qualify for the OID, both item A as well as item B need to be on the request lines (lines with item A and item B need to be passed to the pricing engine). In this case, the pricing engine applies the OID modifier on the request line with item A and it applies the discount on the line with item B.

The OID modifier can be found in the qp_ldets_v with created_from_list_line_type = QP_PREQ_GRP.G_OTHER_ITEM_DISCOUNT with line_index of the request line with item A which is the primary item on the OID modifier. The OID discount is inserted with line_index of request line having item B. The pricing engine also creates a relationship record between the OID modifier and the discount adjustment on the other line in qp_preq_rltd_lines_tmp with relationship_type_code = QP_PREQ_GRP.G_GENERATED_LINE with line_detail_index as the line_detail_index of the OID modifier and the related_line_detail_index as the line_detail_index of the discount adjustment. All this information need to be stored in the calling application's transaction tables.

Pricing from Configurator

The configurator in Order Management (OM)/Order Capture (OC) uses OM/OC APIs to call pricing engine. In case of included items and config items, the unit_price is defaulted to zero. In case the pricing engine is called, the order lines having included items or config items are passed to the pricing engine with price_flag as *N* or *P*, depending upon the OM profile, OM: Charges for Included Item.

If the profile is set to *Y*, then freight charges need to be calculated for the included item. In this case, the price_flag is *P*, else it is *N*. The pricing engine does not calculate the unit_price in either case.

Service Item Pricing

If there is an item, which has price list line set as percentage of price of a parent line; for example, the price of a service item called gold support might be 10% of the price of a parent serviceable item called oracle 8I), pass the parent line in qp_preq_lines_tmp and give the relationship between the parent and the child in qp_preq_rltd_lines_tmp. Set the

relationship_type Line_Index := <parent line index>;

Related_Line_Index := <Child line>;

Relationship_Type_Code := QP_PREQ_GRP.G_SERVICE_LINE;

If the item needs to be priced over a duration of time, as in price a service item for a duration of 12 Months, pass the duration as uom_quantity in the structure qp_preq_lines_tmp. The unit price would be multiplied by the uom_quantity. Insert the relationships into qp_preq_rltd_lines_tmp.

The pricing engine fetches a price list line for the service item and calculates the percent price based on the parent line's list price. Ideally, the parent line can

belong to a different order/quote. Remember to delete/pass the service line when the parent line is deleted/updated.

Pricing also provides a time-UOM conversion API. For more information on Time-UOM conversion API, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*.

Ask for promotions

The calling application can ask for a promotion to be applied to the request line by passing the promotion/modifier as a qualifier to the request line. If the promotion is passed as a qualifier with context = MODLIST, attribute = QUALIFIER_ATTRIBUTE1, value_from = list_header_id of the ask for promotion, the pricing engine applies all the modifiers under the ask for promotion to the request line. The calling application can also ask for a specific modifier line to be applied in which case, the modifier line needs to be passed with context = MODLIST, attribute = QUALIFIER_ATTRIBUTE2, value_from = list_line_id of the modifier line that is asked for. This can be done by inserting the ask for promotion/modifier as an attribute in qp_preq_line_attrs_tmp as a part of append ask for procedure.

Remember that the pricing engine does not look at the qualifiers set up on the asked for promotion/modifier if the validated_flag is passed as Y in the qualifier record in qp_preq_line_attrs_tmp. Make sure the validated_flag is passed appropriately. Also remember to store the asked for promotion into the transaction tables and pass them during every reprice call so that the pricing engine applies it consistently.

Deleting ask for promotions that are applied: If the ask for promotion needs to be deleted, it needs to be deleted from the transaction table and a pricing engine call needs to be made so that the pricing engine does not apply it any more. Also, if the ask for promotion has a modifier that has been overridden, the pricing engine will not delete it. The calling application needs to delete it from the transaction table. When the asked for promotion is deleted, the adjustments with the list_header_id of the ask for promotion need to be deleted. In case the calling application asked for a specific modifier line, the adjustment with the list_line_id as the value_from of the asked for modifier line needs to be deleted.

Terms Substitution

If a request line qualifies for a terms substitution (TSN) modifier, the pricing engine inserts a line detail record in qp_ldets_v with created_from_list_line_type = QP_PREQ_GRP.G_TERMS_SUBSTITUTION.

The following three types of terms upgrade are supported by pricing:

If qp_ldets_v.substitution_attribute=QUALIFIER_ATTRIBUTE1, substitution_to holds payment_term_id.

If qp__ldets_v.substitution_attribute=QUALIFIER_ATTRIBUTE10,

substitution_to holds Freight Term Code.

qp_ldets_v.substitution_attribute=QUALIFIER_ATTRIBUTE11,

substitution_to holds Shipment Method.

If a request line qualifies for a TSN modifier, based on the substitution_attribute which is the term type, the corresponding term e.g. payment/shipment/freight term on the order/quote/request needs to be replaced by the value in the qp_ldets_v.substitution_to. Remember to populate the record structure with the old term before any repricing calls so that the old term gets returned, in case there is a attribute mapping rule. Or make sure the old term gets passed to the pricing engine if there are no attribute mapping rules and if the attributes are passed directly.

Also, if the pricing engine does not pass the TSN modifier on subsequent reprice calls if the request line does not qualify for it any more, make sure to replace the old term before deleting the TSN modifier from the transaction table.

Cross Order Volume Based Modifiers

Pricing engine supports cross order based volume modifiers. These modifiers are setup with pricing or qualifier attributes that are operating unit specific. These are seeded attribute mapping rules under the Order Fulfillment PTE. Examples for pricing attributes are Period1 Item Quantity and Period1 Item Amount.

There is a concurrent program cross order volume loader which looks into the Order Management tables for the order information and sums up cross order volume total for the a given operating unit and populates the OM tables OE_ITEM_CUST_VOLS_ALL and OE_ITEM_CUST_VOLS_ALL. This concurrent program needs to be run from time to time. For more information, see: *Oracle Advanced Pricing Implementation Manual,* Concurrent programs.

Also, the seeded attribute mapping rules get the cross order volume total for specific attributes from the OM cross order volume tables. The pricing engine qualifies the request lines for modifiers based on cross order volume attribute value the build_contexts API returns.

If the calling application is using a new request_type_code or a different global structure, then they need to do the following to make the pricing engine work for cross order volume based discounts:

1. Write a concurrent program to populate the cross order volume total into the above tables or new tables for each cross order volume attribute used in the modifier setup.
2. Write attribute mapping rules to extract the cross order volume total for each cross order volume attribute used in modifier setup.

   For for information on the cross order volume attributes and attribute mapping, see: *Oracle Advanced Pricing User's Guide*, Modifiers setup.

Time-UOM Conversion

Pricing supports unit of measure (uom) conversion for price lists. The unit of measure conversion is done only if the primary uom flag in the price list line is checked. To do this conversion the advance pricing uses inventory uom conversion APIs.

This poses a problem when the units of measure are date and time as in MTH (month), YR (year), WK (week) etc. The inventory uom conversion holds a static conversion factor of 30 days per month. Since days is set as base uom, one year is converted to 12.16666… months by the inventory uom conversion API. (365 days/30days), instead of one year being converted to 12 months.

Pricing also provides option to use the OKS (Oracle Contracts) APIs to perform the uom conversion for date and time which does the conversion based on calendar year. This is controlled by a QP profile, QP: Time UOM Conversion.

Set this profile QP: Time UOM Conversion to Standard if you want the pricing engine to use the inventory API to perform the conversion for date/Time and set it to Oracle Contracts to use the APIs provided by Oracle Contracts to do time UOM conversion.

Remember that the effect of this profile set to Oracle Contracts can take place only if the product code 515: Oracle Contracts Service Module is fully Installed or Shared installed. Also, in cases that cause contracts APIs returns pricing quantity being null or 0, the standard inventory uom conversion takes place. These cases are QP: Time UOM Conversion profile value is null or when contracts APIs fail to generate correct pricing quantity, for example, Contract_Start_Date or Contract_ End_Date being null, order uom is not time related, order uom code is not valid such as MI or MO.

Promotional Limits

Promotional limits are setup against modifiers. When a pricing engine call is made, the pricing engine qualifies the request line for suitable modifiers. If there are limits setup against those modifiers, then the modifiers applied are consumed from this limit. The pricing engine calls the limits engine which maintains the limits balance details. In order to use the limits functionality, the user needs to pass a price_request_code to uniquely identify each request line.

For example, the price_request_code may be built by concatenating the request_type_code||'-'||header_id||'-'||line_id. The limits consumption is updated in the pricing limits tables for each request line and the price_request_code is used as the key to identify the request line. After the pricing engine call, make sure to process any errors related to limits. In case there are errors due to limits, the pricing engine populates error status in qp_preq_lines_tmp.pricing_status_code for each request line. Make sure you handle the errors appropriately. In case the pricing engine call is to price an order, if the pricing engine throws a error status on one of the lines due to limits, then the order or that specific line may be placed on hold, or a warning message may be raised depending upon the business requirement.

The limit balances need to be updated when an order line is returned, amended or cancelled. In case the calling application makes a pricing engine call, the pricing engine does this against the price_request_code passed. If no pricing engine call is made, pricing provides an API to do this. Make sure you call the following API to update the limits consumption details:

```
QP_UTIL_PUB.Reverse_Limits (p_action_code             IN  VARCHAR2,
    p_cons_price_request_code IN  VARCHAR2,
    p_orig_ordered_qty        IN  NUMBER   DEFAULT NULL,
    p_amended_qty             IN  NUMBER   DEFAULT NULL,
    p_ret_price_request_code  IN  VARCHAR2 DEFAULT NULL,
    p_returned_qty            IN  NUMBER   DEFAULT NULL,
    x_return_status           OUT VARCHAR2,
    x_return_message          OUT VARCHAR2);
```

For more information about this API, see: *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*.

Multi-Currency

In multi-currency functionality, the price list passed must exist in the currency passed and vice versa. In case the currency code passed is not a part of the multi-currency price list, then the pricing engine will not be able to find a price. In order to ensure this, pricing provides an API to validate the passed in

currency and price list. This validation needs to be done in the integration code. The API looks like:

```
QP_UTIL_PUB.Validate_Price_list_Curr_code
(
    l_price_list_id             IN NUMBER
    l_currency_code             IN VARCHAR2
    l_pricing_effective_date    IN DATE
    l_validate_result          OUT VARCHAR2
);
```

For more information, see *Oracle Oracle Order Management Suite APIs and Open Interfaces Manual*. The pricing engine will look for multi currency price lists if the profile QP: Multi-Currency Installed is set to Y to derive a price.

Also, pricing provides an API to return the rounded selling price or adjustment amount and uses the rounding factor based on the multi-currency price list and the order currency. This may be used if there is a requirement to round the price apart from the rounding that the pricing engine does. Call the API,

```
QP_UTIL_PUB.Round_price(p_operand        => null
    ,p_rounding_factor       => null
    ,p_use_multi_currency    => p_use_multi_currency
    ,p_price_list_id         => p_price_list_id
    ,p_currency_code         => p_currency_code
    ,p_pricing_effective_date  => p_pricing_effective_date
    ,x_rounded_operand       => l_rounding_factor
    ,x_status_code           => l_status_code
    ,p_operand_type          => 'R'
);
```

Remember to reprice the request if the price list or the currency changes on the order or quote which was priced earlier because, the pricing engine might return a different price based on the different currency.

Sample scripts to make pricing engine calls:

There are sample scripts to make pricing engine calls. The sample scripts can be found in $QP_TOP/patch/115/sql.

1. qp_engine_testing.sql - Demonstrates making a pricing engine call by populating the price request pl/sql structure
2. qp_manual_adjustments.sql - Demonstrates how to apply manual adjustments
3. qp_override_selling_price.sql - Demonstrates overriding the selling price

4. qp_direct_insert.sql - Demonstrates inserting price request information into pricing temporary tables for performance improvement.
5. qp_test_service.sql - Demonstrates Service Item Pricing Setup
6. qp_test_oid.sql - Demonstrates Other Item Discount Setup

Uptake Requirements for Multi-Currency functionality by other Oracle Applications

In Oracle Advanced Pricing, the multi-currency price lists feature allows you to attach multiple currencies to the same price list or agreement. This reduces the volume of data processed and saves significant maintenance work for users. Additionally, you can set up different ways of deriving the conversion rate such as GL daily rate, fixed rate, user entered, and function call.

The following steps are used by the calling applications to support installations using multi-currency price lists:

1. Install Order Management Minipack-H or application release 11.5.8 or higher.
2. Set the Site level profile option QP: Multi-Currency Installed to Yes.
3. Run the concurrent program: Update Price Lists with Multi-Currency Conversion Criteria. After running the concurrent program, all price lists and agreement price lists are converted to multi-currency price lists.
4. If a user has converted to multi-currency price lists, applications calling the pricing engine must pass the control record variable `use_multi_currency` as 'Y' in the pricing engine call for the currency conversion to occur. This variable is the deciding factor for the calling application to start using the Multi-currency functionality.
5. The calling application LOV (list of values) for the price list name at header and lines must be modified to show the multi-currency price lists and currencies.
    - When the user first enters the order currency and clicks the price list, the list of values displays only those price lists whose Currency Conversion's Currency-To is the same as order (transaction) currency. Also, the pricing effective date (if entered) on the sales order must be within the Currency-To effective dates. This is applicable for both the header and line level list of values. The calling application to call an API provided by pricing is called QP_UTIL_PUB.Get_Price_List.
    - When the user first enters the price list and clicks the Currency, the list of values displays all the Currency-To in its Currency Conversion. Also, the

pricing effective date (if entered) on the sales order (transaction) must be within the Currency-To effective dates. This is applicable for both the header and line level list of values. The calling application to call an API provided by pricing is called QP_UTIL_PUB.Get_Currency.

- The Process Order API currently validates the currency and price list passed to it. Now, the currency will have to be one of the Currency-To of the currency conversion criteria attached to this price list. The calling application needs to call the pricing API called QP_UTIL_PUB.Validate_Price_List_Curr_Code.

6. Uptake new rounding API for price list. For re-price processing, the calling application needs to call QP_UTIL_PUB.round_price for price list rounding during re-price processing. This will use the Round To value to round the price.
7. For Conversion Type of Transaction, the calling application integration needs to pass the conversion rate and conversion type entered in the Sales Order header (if any) to the pricing engine.
8. The calling application integration needs to pass the functional currency to the pricing engine control record variable - function_currency.

Sample Code using Order Management Structure

procedure copy_Line_to_request(

```
p_Line_rec OE_Order_PUB.Line_Rec_Type
,px_req_line_tbl in out nocopy QP_PREQ_GRP.LINE_TBL_TYPE
,p_pricing_event varchar2
,p_Request_Type_Code varchar2
)
is
l_line_index pls_integer := nvl(px_req_line_tbl.count,0);
l_uom_rate NUMBER;
v_discounting_privilege VARCHAR2(30);
begin
l_line_index := l_line_index+1;
px_req_line_tbl(l_line_index).Line_id := p_Line_rec.line_id;
px_req_line_tbl(l_line_index).REQUEST_TYPE_CODE :=
p_Request_Type_Code;
px_req_line_tbl(l_line_index).LINE_INDEX := l_line_index;
px_req_line_tbl(l_line_index).LINE_TYPE_CODE := 'LINE';
If p_Line_rec.pricing_date is null or
```

```
p_Line_rec.pricing_date = fnd_api.g_miss_date then
px_req_line_tbl(l_line_index).PRICING_EFFECTIVE_DATE :=
trunc(sysdate);
Else
px_req_line_tbl(l_line_index).PRICING_EFFECTIVE_DATE :=
p_Line_rec.pricing_date;
End If;
px_req_line_tbl(l_line_index).LINE_QUANTITY :=
p_Line_rec.Ordered_quantity ;
px_req_line_tbl(l_line_index).LINE_UOM_CODE :=
p_Line_rec.Order_quantity_uom;
px_req_line_tbl(l_line_index).CURRENCY_CODE :=
```

OE_Order_PUB.g_hdr.transactional_curr_code;

```
If (p_Line_rec.service_period = p_Line_rec.Order_quantity_uom) Then
px_req_line_tbl(l_line_index).UOM_QUANTITY :=
p_Line_rec.service_duration;

Else
INV_CONVERT.INV_UM_CONVERSION(From_Unit =>
p_Line_rec.service_period
,To_Unit => p_Line_rec.Order_quantity_uom
,Item_ID => p_Line_rec.Inventory_item_id
,Uom_Rate => l_Uom_rate);
px_req_line_tbl(l_line_index).UOM_QUANTITY :=
p_Line_rec.service_duration * l_uom_rate;
End If;
px_req_line_tbl(l_line_index).Active_date_first_type := 'ORD';
px_req_line_tbl(l_line_index).Active_date_first :=
OE_Order_Pub.G_HDR.Ordered_date;
If p_Line_rec.schedule_ship_date is not null then
px_req_line_tbl(l_line_index).Active_date_Second_type := 'SHIP';
px_req_line_tbl(l_line_index).Active_date_Second :=
p_Line_rec.schedule_ship_date;
End If;
px_req_line_tbl(l_line_index).PRICE_FLAG :=
nvl(p_Line_rec.calculate_Price_flag,'Y');
end copy_Line_to_request;
```

procedure copy_attributes_to_Request

```
p_line_index number
,p_pricing_contexts_Tbl
QP_Attr_Mapping_PUB.Contexts_Result_Tbl_Type
,p_qualifier_contexts_Tbl QP_Attr_Mapping_PUB.Contexts_Result_Tbl_Type
```

```
,px_Req_line_attr_tbl in out nocopy
QP_PREQ_GRP.LINE_ATTR_TBL_TYPE
,px_Req_qual_tbl in out nocopy
QP_PREQ_GRP.QUAL_TBL_TYPE
)
is
i pls_integer := 0;
l_attr_index pls_integer := nvl(px_Req_line_attr_tbl.last,0);
l_qual_index pls_integer := nvl(px_Req_qual_tbl.last,0);
begin
i := p_pricing_contexts_Tbl.First;
While i is not null loop
l_attr_index := l_attr_index +1;
px_Req_line_attr_tbl(l_attr_index).VALIDATED_FLAG := 'N';
px_Req_line_attr_tbl(l_attr_index).line_index := p_line_index;
-- Product and Pricing Contexts go into pricing contexts...
px_Req_line_attr_tbl(l_attr_index).PRICING_CONTEXT
:=
p_pricing_contexts_Tbl(i).context_name;
px_Req_line_attr_tbl(l_attr_index).PRICING_ATTRIBUTE :=
p_pricing_contexts_Tbl(i).Attribute_Name;
px_Req_line_attr_tbl(l_attr_index).PRICING_ATTR_VALUE_FROM :=
p_pricing_contexts_Tbl(i).attribute_value;
i := p_pricing_contexts_Tbl.Next(i);
end loop;
-- Copy the qualifiers
i := p_qualifier_contexts_Tbl.First;
While i is not null loop
l_qual_index := l_qual_index +1;
If p_qualifier_contexts_Tbl(i).context_name ='MODLIST' and
p_qualifier_contexts_Tbl(i).Attribute_Name
='QUALIFIER_ATTRIBUTE4' then
If OE_Order_PUB.G_Line.agreement_id is not null and
OE_Order_PUB.G_Line.agreement_id <>
fnd_api.g_miss_num then
px_Req_Qual_Tbl(l_qual_index).Validated_Flag := 'Y';
Else
px_Req_Qual_Tbl(l_qual_index).Validated_Flag
:= 'N';
End If;
Else
px_Req_Qual_Tbl(l_qual_index).Validated_Flag := 'N';
End If;
px_Req_qual_tbl(l_qual_index).line_index := p_line_index;
px_Req_qual_tbl(l_qual_index).QUALIFIER_CONTEXT :=
```

```
p_qualifier_contexts_Tbl(i).context_name;
px_Req_qual_tbl(l_qual_index).QUALIFIER_ATTRIBUTE :=
p_qualifier_contexts_Tbl(i).Attribute_Name;
px_Req_qual_tbl(l_qual_index).QUALIFIER_ATTR_VALUE_FROM :=
p_qualifier_contexts_Tbl(i).attribute_value;
i := p_qualifier_contexts_Tbl.Next(i);
end loop;
end copy_attributes_to_Request;


procedure copy_Header_to_request(
p_header_rec OE_Order_PUB.Header_Rec_Type
,px_req_line_tbl in out nocopy QP_PREQ_GRP.LINE_TBL_TYPE
--,p_pricing_event varchar2
,p_Request_Type_Code varchar2
,p_calculate_price_flag varchar2
)
is
l_line_index pls_integer := px_req_line_tbl.count;
begin
l_line_index := l_line_index+1;
px_req_line_tbl(l_line_index).REQUEST_TYPE_CODE
:=p_Request_Type_Code;
--px_req_line_tbl(l_line_index).PRICING_EVENT :=p_pricing_event;
--px_req_line_tbl(l_line_index).LIST_LINE_LEVEL_CODE
:=p_Request_Type_Code;
px_req_line_tbl(l_line_index).LINE_INDEX := l_line_index;
px_req_line_tbl(l_line_index).LINE_TYPE_CODE := 'ORDER';
-- Hold the header_id in line_id for 'HEADER' Records
px_req_line_tbl(l_line_index).line_id := p_Header_rec.header_id;
if p_header_rec.pricing_date is null or
p_header_rec.pricing_date = fnd_api.g_miss_date then
px_req_line_tbl(l_line_index).PRICING_EFFECTIVE_DATE :=
trunc(sysdate);
Else
px_req_line_tbl(l_line_index).PRICING_EFFECTIVE_DATE :=
p_header_rec.pricing_date;
End If;
px_req_line_tbl(l_line_index).CURRENCY_CODE :=
p_Header_rec.transactional_curr_code;
px_req_line_tbl(l_line_index).PRICE_FLAG := p_calculate_price_flag;
px_req_line_tbl(l_line_index).Active_date_first_type := 'ORD';
px_req_line_tbl(l_line_index).Active_date_first :=
p_Header_rec.Ordered_date;
end copy_Header_to_request;
```

Refer to the sample script qp_direct_insert.sql in $QP_TOP/patch/115/sql to call the insert APIs of the pricing engine.

Changed Lines API

This API permits only the modified and related lines to be passed to the pricing engine. This reduces unnecessary pricing processing and improves performance.

In typical pricing, there are multiple order lines or quote lines per order or quote. The calling applications such as Order Management and Order Capture that use Oracle Advanced Pricing for pricing their transactions make pricing engine calls with one or more order lines in a single pricing request. Instead of passing all the order lines in every order, it is efficient to just pass the modified lines (newly entered lines or existing lines with update) to the pricing engine so that the Pricing Engine will not have to process the unnecessary lines that will result in the same prices anywhere.

The calling applications make calls to QP_UTIL_PUB.Get order line status to determine what to pass. There are three flags on the output record of this API:

All_Lines_Flag ('Y' or 'N')

This flag tells the call applications whether all lines should be passed to the engine. The value of this is determined by the pricing setup. For these modifiers, all lines information is required in order for the pricing engine to evaluate the eligibility of discounts.

Changed_Lines_Flag ('Y' or 'N')

When this flag is set to 'Y', only the modified lines should be passed to the engine. This flag is 'Y' when there is only Line Level Modifiers.

Summary_Line_Flag ('Y' or 'N'):

The value of this flag is depended on the pricing setup with Order Level Modifiers. When there is only such a modifier, this flag will be set to 'Y' so that only the Order Summary Line should be passed to the engine.

Scenarios

Case 1

100 lines order with 5 lines changed.

Setup: All_Lines_Flag = 'N', Changed_Lines_Flag = 'Y', Summary_Line_Flag = 'N'

Result: Only 5 lines are passed to the pricing engine.

Case 2

100 lines order with all lines changed.

Setup: All_Lines_Flag = 'Y', Changed_Lines_Flag = 'Y/N', Summary_Line_Flag = 'N'

Result: All 100 lines are passed to the pricing engine.

Case 3

100 lines order with 5 lines changed and either Other Item, Promotional Goods or Group of Lines discount.

Setup: All_Lines_Flag = 'Y', Changed_Lines_Flag = 'Y/N', Summary_Line_Flag = 'N'

Result: All lines are passed to the pricing engine.

Case 4

100 lines order with 5 lines changed and order level modifier.

Setup: All_Lines_Flag = 'N', Changed_Lines_Flag = 'Y', Summary_Line_Flag = 'Y'

Result: 5 lines and a summary line are passed to the pricing engine.

Case 5

100 lines order with no line changed.

Setup: All_Lines_Flag = 'N', Changed_Lines_Flag = 'N', Summary_Line_Flag = 'N'

Result: No lines are passed to the pricing engine.

Case 6

2 new lines are added to the 100 lines order.

Setup: All_Lines_Flag = 'N', Changed_Lines_Flag = 'Y', Summary_Line_Flag = 'N'

Result: Only the 2 new lines are passed to the pricing engine.

16

Technical Considerations

This chapter addresses technical considerations and techniques for the implementation of Oracle Advanced Pricing. The following topics are discussed:

Basic versus Oracle Advanced Pricing

Oracle Advanced Pricing and Oracle Order Management, with its basic pricing component, share a common code set. Licensed users of Oracle Order Management are authorized to use the basic features of the set, while licensed users of Oracle Advanced Pricing are authorized to use the full set of features.

The following SQL statement can be issued from within SQL*Plus to obtain the installation status for Oracle Advanced Pricing:

```
SQL: select qp_util.get_qp_status from dual;
```

The following table describes the values of get_qp_status:

Table 16–1 Values of get_qp_status

Value	Meaning
S	Shared installation. Only basic pricing features are available.
I	Full installation. All Oracle Advanced Pricing features are available.

When pricing functionality is installed SHARED, the setup forms contain restricted function. This is accomplished using several methods, including:

- Restricting the list of values a user may select from. For example, the list type code shows only discount, surcharge, and freight list types.
- Disabling fields. Fields that are not available to basic pricing users appear grey in the window, or do not display at all.
- Defaulting fields not available to basic pricing users. For example, the field incompatibility group automatically defaults to LVL 1: Level 1 Incompatibility when using the basic feature set.
- Modifying regions and tabs. For example, Oracle Advanced Pricing modifier tabs (coupons, promotional upgrade) are not shown on the modifier window when Oracle Advanced Pricing is not installed.

The pricing engine and pricing setup APIs are authorized for use only if Oracle Advanced Pricing is installed.

> **Note:** Although the pricing engine is a common component of both basic and Oracle Advanced Pricing, it does not execute the API Get_qp_status. The pricing engine returns user setups as allowed by the APIs and setup forms.

Architectural Overview

Oracle Advanced Pricing is a new product that represents a significant increase in pricing functionality compared to Oracle Order Entry.

Design Objectives

Oracle Advanced Pricing and Basic pricing are designed to:

- Support the complexities of pricing while providing an intuitive user interface.
- Accommodate new inventions in promotion management and e-business pricing.
- Provide flexibility that meets the individual pricing needs of customers in different industries.

Architectural Changes and Innovations

Some of the major architectural changes and innovations of this product include:

- A new data model. All internal pricing data, including price lists, discounts, and pricing rules are kept in a common set of tables internal to Oracle Advanced Pricing. This data model includes a variety of constructs necessary to support Advanced e-business needs.
- Oracle Advanced Pricing integrates with Oracle Order Management and Oracle Customer Relationship Management products such as Oracle iStore. The pricing engine is called on-demand, passing appropriate parameters at the time of call. These parameters are passed in a pricing request structure, which may be a single or a multi-line call. Parameter and results passing are performed strictly via APIs.
- Oracle Advanced Pricing receives a pricing request structure when invoked from a calling application. This incoming pricing request references internal tables based on the parameters passed at invocation, and returns an answer set in a defined API.
- Multilingual Support (MLS) is provided for price lists, modifiers, and formula tables.

Oracle Advanced Pricing Engine Processing

The Oracle Advanced Pricing Engine is composed of a search engine and a calculation engine.

Search Engine

The search engine uses qualifier and pricing attributes to select a list of price and modifier list lines. These may be applied to the pricing request according to rules of eligibility, incompatibility, exclusivity, and precedence.

Determining Eligibility

The search engine selects lists and list lines based on effectivity dates. It compares the pricing effective date supplied by the calling application against the following dates to determine if a list or list line is effective for the given pricing date:

Price Lists

- Price list: effective date range
- Price list line: start and end date
- Price list qualifiers: start and end date

Modifier Lists

- Modifier list: effective date range
- List qualifiers: start and end date
- Modifier list line: start and end date
- Line qualifiers: start and end date

Formulas

- Pricing formula: effective date range

Modifier List Additional Dates Ranges

The calling application can pass two additional dates of a seeded date type to the search engine. The search engine uses these dates and the pricing effective date to determine which modifier lists are eligible.

Modifier lists have two date ranges and a date type for each range. This allows additional date eligibility to be defined on a modifier list. Seeded values for date type are:

- Order date
- Requested ship date

For example, a summer promotion is available between June 1 and August 31. To receive this promotion the customer must order before July 31. Effective dates of the promotion are June 1 through August 31, with an additional date range defined by the order date type that has an end date of July 31.

> **Note:** If both of the above date ranges are defined for a modifier list, the criteria of both must be met. For example, promotion XYZ has an order date type with a range of December 1 through December 31. It has a second date type of requested ship date with a range of December 15 through December 31. Both the order date and requested ship date must fall within the range to receive the promotion. If you wish to make this an OR condition, a qualifier should be used.

Active/Inactive Lists

To be selected by the search engine, a list must be marked as Active. If the active flag is set to no (inactive), the search engine does not consider the list, even if it is effective for the pricing date.

Pricing Currency

For single currency price lists, the search engine matches the currency of the transaction to the currency of the price and/or modifier lists. Only lists defined in the same currency as the transaction are selected. The currency of the transaction must be included in the pricing request structure.

To price transactions in multiple currencies, set up a price list and modifier list for each currency where a transaction may be priced. Formula functionality enables you to convert the base currency price and modifier lists to the transaction currency, passing the transaction currency as a formula pricing attribute.

For Multiple Currency price lists: The search engine matches the currency of the transaction to the currency of the price list(s) with the base or conversion currencies matching this currency. The pricing engine converts the price from the base currency and calculates the transaction currency based on the established conversion rules. The search engine matches the currency of the transaction to the currency of the modifier lists. Only modifiers in the same currency as the

transaction are selected. The currency of the transaction must be included in the pricing request structure.

To price transactions in multiple currencies, set up a Multiple Currency price list with attached multiple currency conversions, and a modifier list for each currency where a transaction may be priced.

Identifying Appropriate Transaction Pricing Data

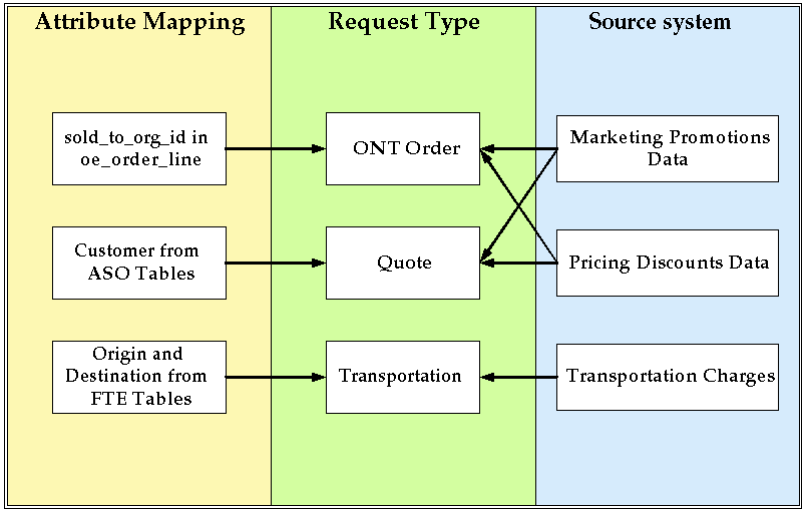
The search engine must know the data source to determine whether it is appropriate to price a transaction. For example, only price lists set up to price contracts should be considered when deriving a price for a contract line.

A source system identifier, that identifies the application which created the pricing data, is recorded on all price and modifier lists. Examples of this include Oracle Order Management, Oracle Marketing, Oracle Service Contracts, and iOracle iMarketing.

The request type identifies the type of transaction that is being priced. Each pricing request line must be identified with a request type. Examples of this include Oracle Order Management and Oracle Order Capture.

The mapping of a request type to one or more source systems defines the source(s) of pricing data that are used to price a transaction. For example, if a pricing request line has a request type of Order, the search engine considers only data from a source system that is mapped to this type of request. The following figure illustrates the source system and request mapping concept.

Figure 16–1 Source system and request mapping concept



Phasing the pricing of a transaction

A user may configure a pricing transaction, when the source system process flow requires it to occur, through pricing phases and pricing events. Pricing phases and events also define the pricing data that should be considered for application to the request at that point in the transaction process flow. Rather than pricing an entire transaction at once, you may break up pricing into discrete activities.

A pricing event occurs when a specific point in the process flow of the sourcing application requires service by the pricing engine. Pricing events in the source system trigger base price determination for the transaction, certain transaction lines, and/or price adjustments, benefits, or charges to be applied to the whole transaction or to specific transaction lines.

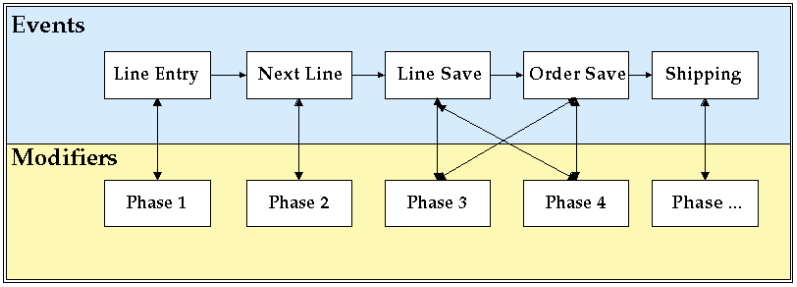
A pricing phase controls which modifiers are considered by the search engine and in what sequence they should be applied to the request. Some pricing phase attributes determine which modifiers can be placed in a phase. For example, if the list type of the pricing phase is charges, only list lines of this type may be placed in the phase. Phase attributes that are used to control the modifiers placed in the phase are:

- Modifier level code
- List type code
- List line type code

Note: All price lists are automatically placed in the seeded Phase 0 list line base price and cannot be assigned to any other phase.

A pricing event can be mapped to one or more pricing phases, and a pricing phase can be assigned to more than one pricing event. The following figure illustrates this how pricing events and phases can be mapped.

Figure 16–2 Pricing events and phases mapping



The concept illustrated in the previous figure allows the following pricing business rules types to be implemented:

- All freight and special charges should be calculated at the time of shipping.
- Once total order volumes have been derived in a high volume batch environment, all cross order volume discounts are applied at the end of the day.
- Coupons are given only after all items in the shopping cart have been priced and the user has proceeded to final checkout.

Selecting Price Lists and Modifier Lists based on Qualifiers

A qualifier is a user-defined rule that specifies one or more conditions under which the pricing engine applies a price list, price adjustment, or other benefits or charges. If a condition described in a qualifier evaluates as true when the search engine is processing, then the pricing object(s) tied to that rule are considered eligible by the search engine.

Many companies apply prices and discounts across and at different levels of their customer hierarchy. Each level at which businesses set their pricing and discounting can be conditionally tested in a qualifier.

When creating a price list, discount, or promotion, pre-defined qualifiers can be used to define who is eligible for the price or modifier.

Qualifier attributes are combined with operators to create qualifying conditions for a list header or modifier. The following operators are available:

- =
- Between (includes >= and <=)
- Not =

Qualifying conditions such as the following may be used to define eligibility for prices, promotions, etc.

- Customer = XYZ
- Sales territory = Northeastern Region
- Order amount > $100
- Contract signed date between December 1 and December 31

When pricing a transaction line, the search engine compares the qualifying conditions on the list header with the value of the qualifiers on the pricing request line. Qualifiers for a pricing request line are populated using dimension sourcing. For example, if eligibility for a price list is qualified by Market Sector = Consumer Products, and if the value for the market sector qualifier is consumer products, then the price list is available to price that transaction line. If the qualifiers on the list header are met for modifier lists, the search engine also ensures that any qualifiers at the line level are satisfied.

Qualifiers may be grouped to create AND/OR conditions using a grouping number. For example, if a 10% discount is given provided that a customer is a preferred customer AND the customer spends more than $150, the qualifying condition may be defined by creating qualifiers in the same qualifier group:

Table 16–2 Qualifier Grouping: AND Condition

Qualifier Group	Qualifier Attribute	Operator	Value From	Value To
1	Customer class	=	Preferred	Not applicable
1	Order amount	Between	150	99999999999

If the requirement is to give a 10% discount provided that a customer is a preferred customer or the customer spends more than $150, the qualifying condition may be defined by creating qualifiers in different qualifier groups:

Table 16–3 Qualifier Grouping: OR Condition

Qualifier Group	Qualifier Attribute	Operator	Value From	Value To
1	Customer class	=	Preferred	Not applicable
2	Order amount	Between	150	99999999999

Multiple qualifiers may be included in a qualifier group.

If a qualifier is mandatory for all qualifying conditions and must be included in all qualifier groups, a -1 qualifier group number can be given. The search engine always ensures that this condition is met before proceeding to check all other groups. For example, the conditions to receive a 10% discount on an order are that a customer is a preferred customer OR the customer must spend more than $150 AND must place the order on a United Kingdom website, and the customer must always pay with a Visa card. This is defined as follows:

Table 16–4 Qualifier Grouping: Combined -1, AND, OR condition

Qualifier Group	Qualifier Attribute	Operator	Value From	Value To
-1	Credit card type	=	Visa	Not applicable
1	Customer class	=	Preferred	Not applicable
2	Order amount	Between	150	99999999999
2	Website domain	=	.co.uk	Not applicable

Pricing Attributes: Selecting What is Priced

Pricing attributes are user-defined attributes which define what is being priced or modified. Attributes may include factors which affect the price of the item, provide additional definition without the need to create an item, or manage pricing and discounting at a level higher than in the product hierarchy.

Examples of pricing attributes are:

- Item X:
  - **a.** Grade A is priced at $60
  - **b.** Grade B is priced at $55
  - **c.** Grade C is priced at $50
- Servicing of a photocopier is priced based on:
  - **a.** Distance from the service center
  - **b.** Age of the photocopier
  - **c.** Environment
- Training course discounts are given based on the number of attendees:
  - **a.** 1 - 5 students, 10% discount
  - **b.** 6 - 10 students, 12% discount
  - **c.** 10 or more students, 15% discount
- All contact lenses are priced at $2.25 per pair.

Pricing attributes are defined in the Oracle Advanced Pricing descriptive flexfield - Pricing Contexts. Creating pricing attributes in different contexts allows attributes to be grouped according to their business use. The item context is reserved for defining product pricing hierarchy; every level in the product hierarchy at which pricing and discounting is set should be defined as a segment in this context.

Pricing category:	Milled goods
Pricing sub-category:	Flour
Margin group:	Branded flour
Margin sub group:	Homepride
Product group:	White
Size	1.0kg

Configuration	12*2
Stock keeping unit:	SKU1234

Product Pricing Hierarchy Example

In the product pricing hierarchy example given, each of the eight levels are defined as a segment in the item context in the pricing context descriptive flexfield.

The segments are ordered to reflect the structure of the product pricing hierarchy, with the lowest level in the hierarchy having the lowest flexfield segment sequence number. The search engine uses the sequence to select the most specific price or discount. For example, if no price is found for a promotion product, use the configuration price. If no price is found for the configuration, use the pack size price. If no price is found for the pack size, use the flour type. The item context of the pricing attribute flexfield represents a flattened view of the product pricing hierarchy.

Oracle Advanced Pricing includes the following seeded product hierarchy:

All (all products):	Enabled
Segment 1 - segment 20 item category flexfield:	Disabled
Item category key flexfield:	Enabled
Item	Enabled

As with qualifiers, the search engine compares pricing attributes on the transaction or pricing request line with pricing attributes on the price or modifier list line. This determines if the base price or modifier can be applied to the request line. Product and pricing attributes for a pricing request line can be either user entered or populated using attribute mapping.

Exclusion enables a price adjustment, benefit, or charge to be given at a level in the product pricing hierarchy but it excludes lower levels in the hierarchy from that modifier. In the product pricing hierarchy defined previously, if a 10% discount is given on all flour apart from the Homepride brand, then a discount can be created with a product attribute of Oracle Advanced Pricing Subcategory = Flour, excluding Margin Sub Group = Homepride. The search engine eliminates any pricing request lines with the margin sub group pricing attribute of Homepride.

The search engine returns only those modifiers which are defined in the same UOM as pricing (which is determined when the base price is derived as described) or where there is no product UOM specified. The latter may be the case if the modifier is not product specific.

UOM Conversion Logic

The search engine only returns modifiers which are defined in the same unit of measure as the pricing unit of measure, or when no unit of measure specified. The latter may be the case if the modifier is not product specific, or if the type of modifier does not require a unit of measure.

Modifier Level Code

Modifier level code determines which qualifiers and pricing attributes are considered by the search engine when deciding if a request line qualifies for a modifier. This code also determines at what level a modifier should be applied to the request.

Line Level Code

Only qualifiers and pricing attributes of an individual request line are considered by the search engine. For volume related pricing attributes only the quantity of the request line is considered for qualification. Modifier application is at the request level.

Group of Lines Level Code

The quantity in the pricing UOM and amount spent on an item is summed across all qualified request lines. The total item quantity and amount on the request or total quantity and amount at a level in the product hierarchy is considered by the search engine when deciding whether a modifier is qualified. Modifier application is at the request line level.

Order Level Code

Only qualifiers or pricing attributes of the summary request line or header are considered by the search engine when deciding whether a modifier is qualified. It is not possible for a header level modifier to be qualified by a request line. Modifier application is at the summary request line/header level.

Freezing Pricing Request Lines

The Calculate Price flag allows the calling application to fully or partially freeze the price on a pricing request line. Price may be completely frozen with no additional modifiers applied, or additional modifiers may be applied in certain phases, depending on the value of the flag. Possible values are as follows:

Table 16–5 Calculate Price Flag values

Flag Value	Action
Y (calculate price)	Applies all prices and modifiers to the request line.
N (freeze price)	Does not apply any prices or modifiers to the request line.
P (partial price)	Only applies prices or modifiers in certain phases.

When the calculate price flag is set to partial price, the search engine observes the freeze override flag on the phase. When the calculate price flag is set to yes the search engine applies eligible modifiers in the phase to the request line. When the calculate price flag is set to no the modifiers in the phase are not considered for application to the request line. This enables, for example, the price of a line to be fixed but still allows freight charges to be applied to the line.

Other Qualifying Items

Some types of modifiers allow multiple buy items to be specified as a qualification for the benefit or charge. For example:

> If you buy a set of six chairs and a coffee table, or two standard lamps, get $400 off a dining table.

Using this example, the search engine determines whether the pricing request contains all qualifying items in the specified quantity or amount before the modifier is applied to the pricing request line for the benefit item (the $400 discount on the dining table). There must always be a primary item in the modifier which can be combined with groups of other items. In the above example, chair is the primary item combined with two other qualifying items, coffee table and standard lamp. The OR condition is achieved by giving additional buy items different grouping numbers. If the condition were six chairs, a coffee table, and two standard lamps, the grouping numbers would have been the same.

> **Note:** The only pricing attributes that can be used with the additional qualifying items are those in the volume context.

Modifiers that allow the definition of additional qualifying items are:

- Other Item Discount
- Promotional Goods
- Coupon Issue

Resolving Incompatibility and Exclusivity Between Modifiers

Once the search engine locates all modifiers eligible for application to a pricing request line, it must be determined whether the modifiers are exclusive or incompatible.

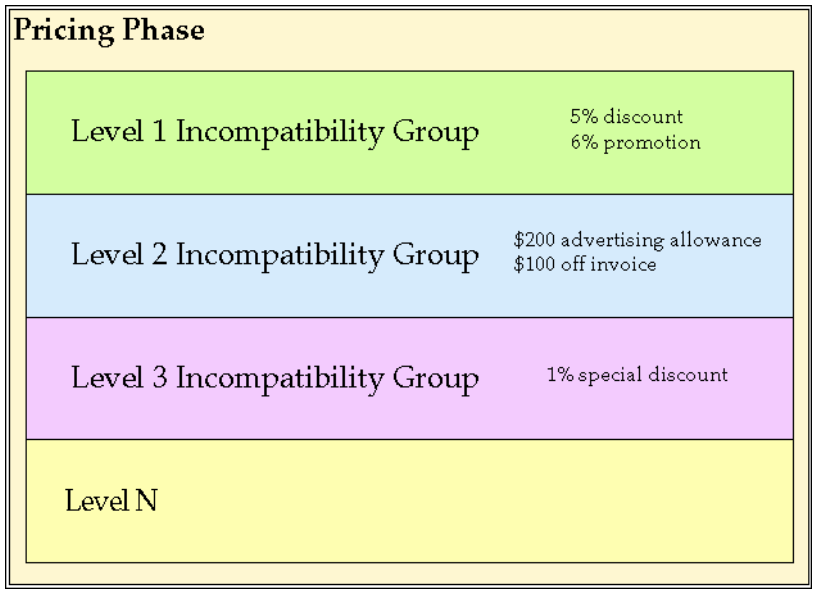
Additional modifiers may not be applied to a pricing request line if an exclusive modifier is applied to the request line in the same pricing phase. For example, a customer receives a 15% discount on an item and is not eligible for any other discount on the item.

An incompatible modifier may not be applied to the same pricing request line as any other incompatible modifier within a pricing phase. For example, a 5% discount on a tennis racquet may not be given if the customer is also eligible for "Summer Sports Promotion" that gives a 6% discount on all sporting goods.

Incompatibility between discounts is defined between modifiers at the same level in a discount application hierarchy. This is illustrated in the following image. In the image, Level 1 is base level discounts, Level 2 is specially negotiated discounts, and Level 3 is retrospective discounts/accruals.

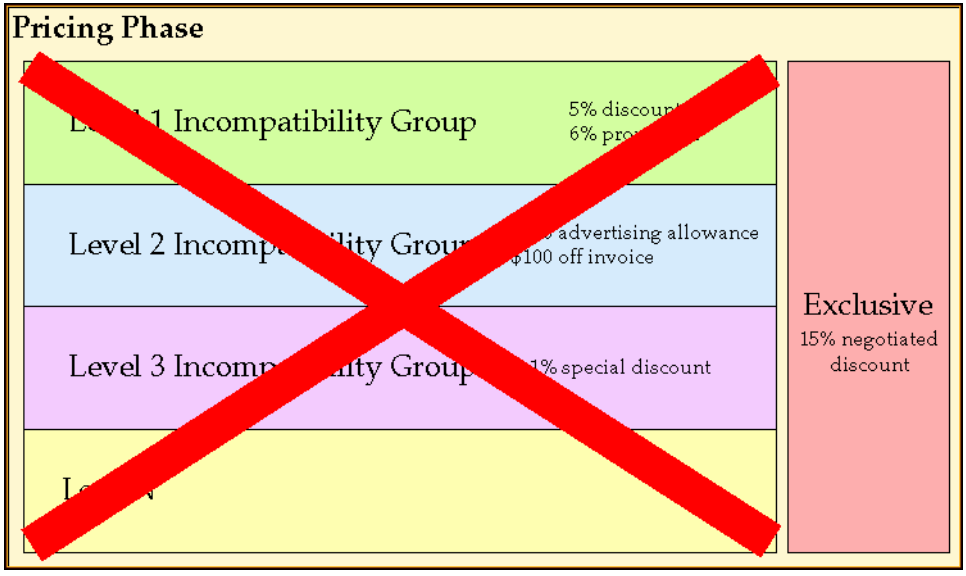
If the discounts of Levels 1 through 4 in the following image are set for the same product family (and therefore applied to the same pricing request line) a customer entitled to receive all of these discounts only receives one discount from each incompatibility group. For example, the customer could be eligible for a 5% discount, a 1% special discount, and a $100 off invoice.

Figure 16–3 Incompatibility Groups for a Pricing Phase



If the customer negotiates a 15% exclusive discount for the same product family and the search engine determines that this can be applied to the pricing request line, this is the only discount applied to the line in the current pricing phase. None of the discounts in Levels 1 through N are considered. This concept is illustrated in the following image.

Figure 16–4 Incompatibility Groups and Exclusivity rules



Note: Incompatibility and exclusivity rules only apply to modifiers that are in the same pricing phase and that are eligible to be applied to the same pricing request line.

Applying Incompatibility or Exclusive Groups

Price List Lines

Price list lines are always treated as exclusive; all price lists lines are automatically assigned to EXCL: Exclusive Incompatibility Group. When multiple prices are found for the same pricing request line in the ordered UOM or in the pricing UOM the search engine selects the price list line using precedence resolution. If the price list lines are of equal precedence the search engine returns an error for the request line.

The following image depicts price list search/incompatibility processing, part A. It contains diamond boxes, which represent decisions, and rectangular boxes, which represent processes.

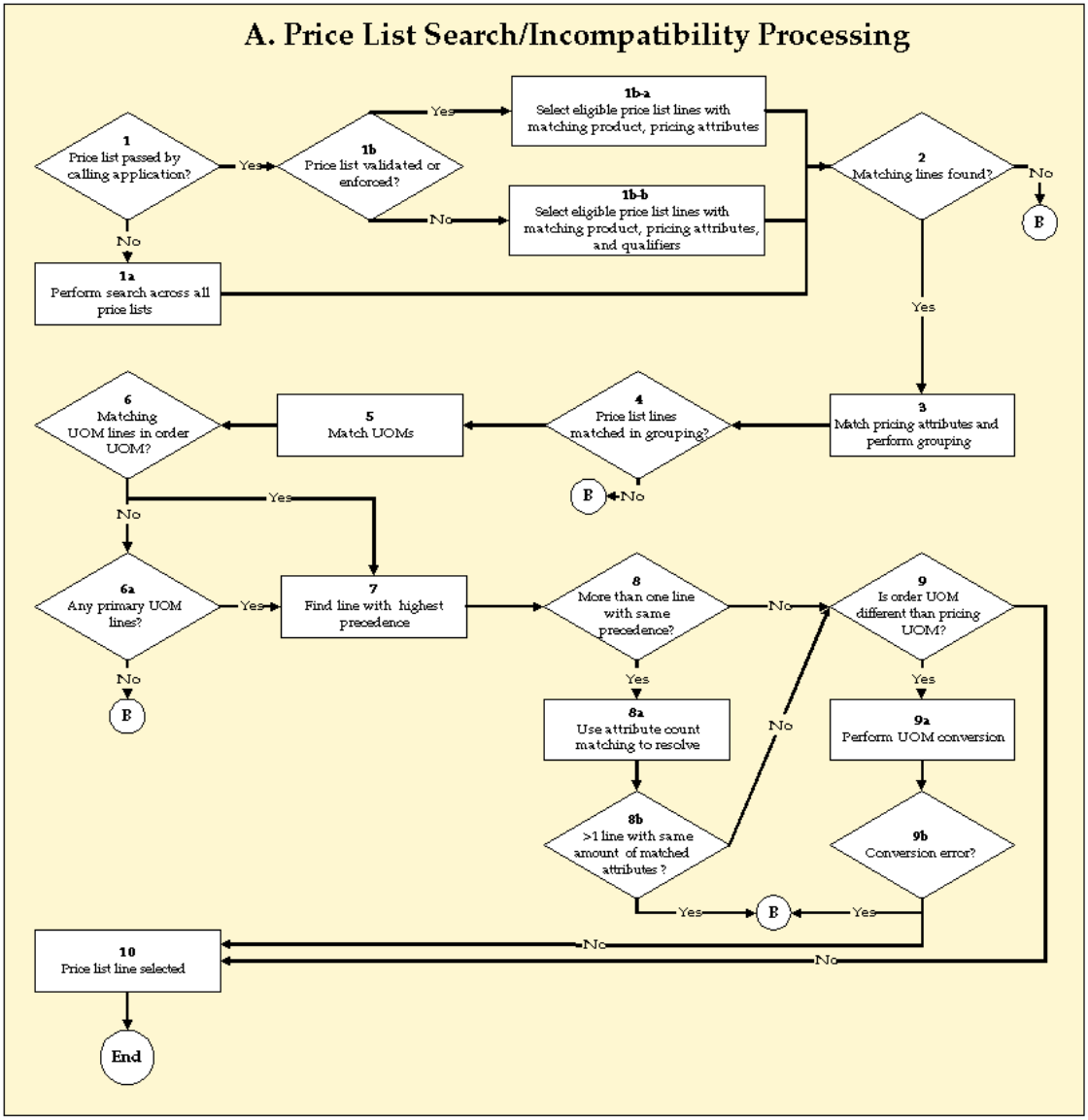
Box 1 asks whether the price list is passed by the calling application.Yes leads to Box 1a: Perform search across all price lists. Box 1a then progresses to Box 2. A no result for Box 1 leads to Box 1b, which asks whether the price list has been validated/enforced. Yes leads to Box 1b-a: Select eligible price list lines with matching product and pricing attributes. No leads to Box 1b-b: Select eligible price list lines with matching qualifiers, product and pricing attributes. Both of these boxes lead to Box 2.

Box 2 asks if matching lines are found. No leads to part B. Yes leads to Box 3: Match pricing attributes and perform grouping. Proceed to Box 4, which asks whether any price list lines are matched in grouping. No leads to part B. Yes leads to Box 5: Match UOMs. Proceed to Box 6, which asks if there are matching UOM lines in the order UOM. Yes leads Box 7. No leads to Box 6a which asks whether there are any primary UOM lines. No leads to part B. Yes leads to Box 7: Find the line with the highest precedence. Proceed to Box 8.

Box 8 asks if there is more than one line with the same precedence. Yes leads to Box 8a: Use attribute count matching to resolve the same precedence issue. Proceed to Box 8b, which asks if there is more than one line with the same amount of matched attributes. Yes leads to part B; no leads to Box 9.

A no result to Box 8 leads to Box 9, which asks whether the order UOM is different than the pricing UOM. No leads to Box 10. Yes leads to Box 9a: Perform UOM conversion. Proceed to Box 9b, which asks if there is a conversion error. Yes leads to part B. If no, proceed to Box 10: Price list line selected. This ends the process.

Figure 16–5 Price List search and incompatibility processing



The following image depicts price list search/incompatibility processing, part B. It contains diamond boxes, which represent decisions, and rectangular boxes, that represent processes.

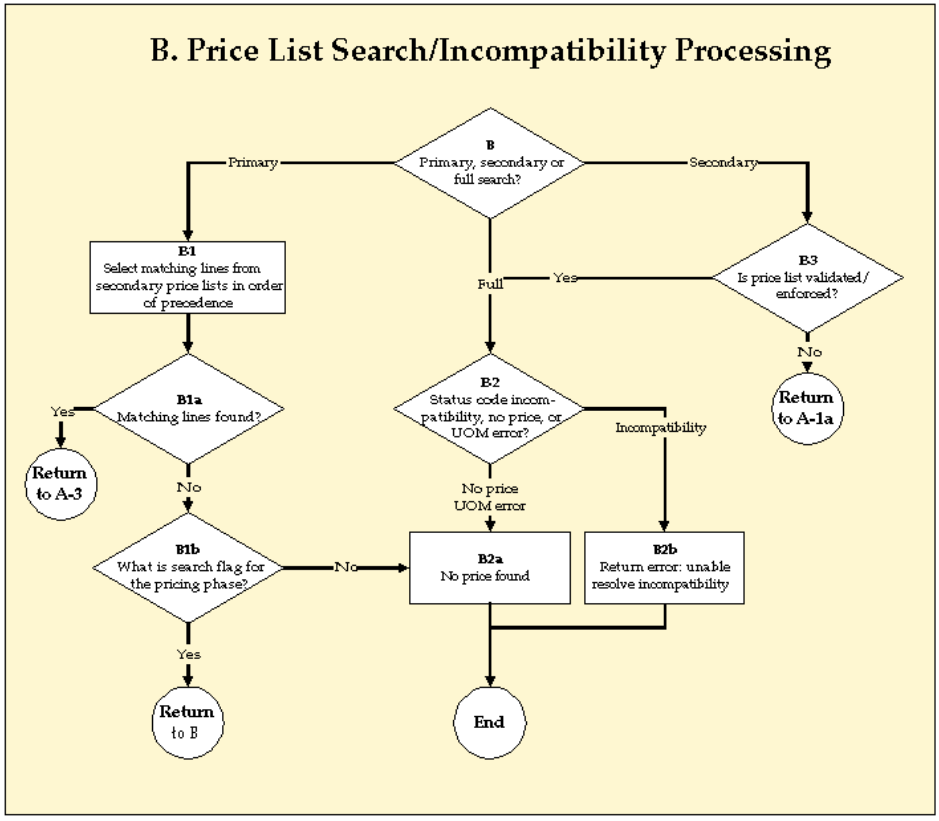
Box B asks if the search is primary, secondary or full.

If the search is primary, proceed to Box B1: Select matching lines from secondary price lists in order of precedence. Proceed to Box B1a, which asks if any matching lines are found. If yes, return to part A, Box 3. If no, proceed to Box B1b, which asks if the search flag for the pricing phase is yes or no. If yes, return to Box B. If no, Proceed to Box B2a: No price found.

If the search is secondary, proceed to Box B3 which asks whether the price list is validated/enforced. If no, return to part A, Box 1a. If yes, proceed to Box B2.

If the search is full, proceed to Box B2, which asks if the status code is incompatibility, nor price, or UOM error. No price and UOM error lead to Box 2a. Incompatibility leads to Box B2b: Return error: unable to resolve incompatibility. Boxes B2a and B2b end the process.

Figure 16–6 Price List search and incompatibility processing



Modifiers

When multiple modifiers in an incompatibility group are eligible for application to a pricing request line, the search engine must determine which modifier should be applied. There are two methods which the search engine uses to determine this: precedence and best price. Which method the engine chooses depends on the incompatibility resolution method which is set for the pricing phase. If the incompatibility resolution code for the phase is precedence, then the search engine selects the most specific modifier. If this fails (the modifiers have equal precedence), then the search engine uses best price resolution. If the incompatibility resolution method on the phase is best price, the search engine attempts to find the modifier that gives the greatest discount. If this fails, the search engine returns an error for the request line.

The following image depicts modifiers search/incompatibility processing. It contains diamond boxes, that represent decisions, and rectangular boxes, which represent processes.

Box 1 asks whether any modifier/header line is passed as certified asked for. Yes leads to Box 1a: Select modifiers for matching products without qualification. No leads to Box 1b: Select modifiers by matching qualifiers, products, attributes. Both boxes lead to Box 2: Perform price break evaluation. Proceed to Box 3: Perform line group processing. Proceed to Box 4: Perform qualifier group and attributes group matching.

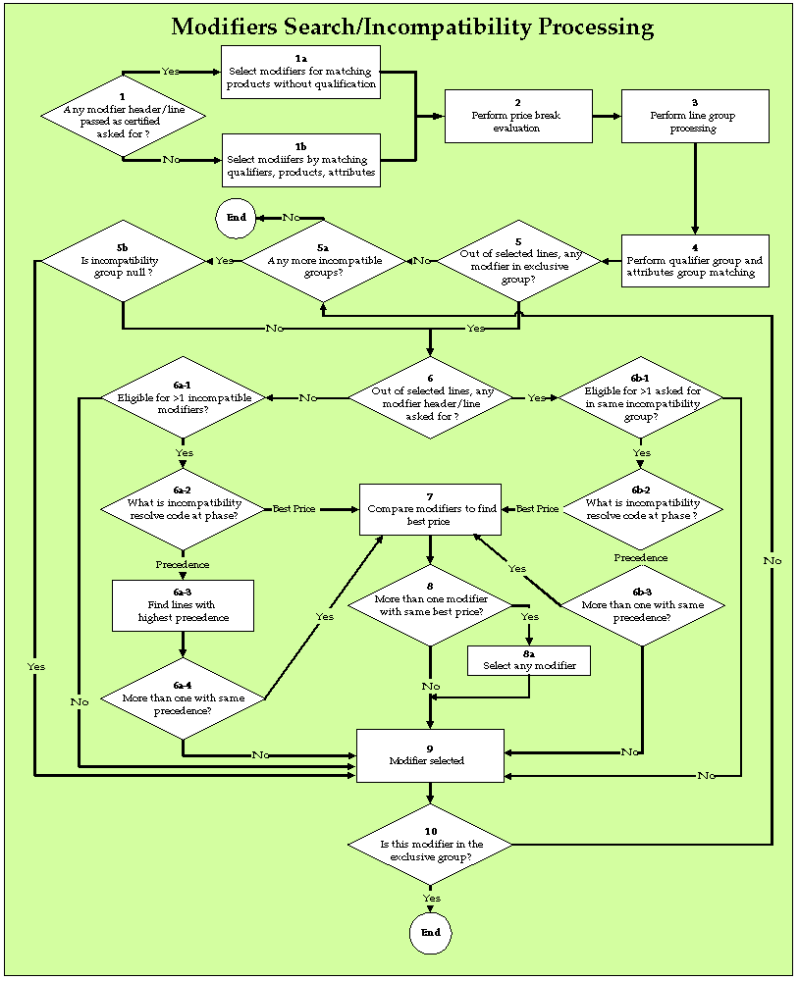
Box 4 leads to Box 5, which asks whether any modifiers of the selected lines are in an exclusive group. Yes leads to Box 6. No leads to Box 5a, which asks if there are any more incompatible groups. If no, then the process is complete. Yes leads to Box 5b which asks if the incompatibility group is null. Yes leads to Box 9. No leads to Box 6.

Box 6 asks whether any of the modifiers in the selected lines are asked for. Each answer leads to a different case. No leads to the first case, that begins with Box 6a-1. This box asks whether any of the selected lines are eligible for more than one incompatible modifiers. No leads to Box 9. Yes leads to Box 6a-2, which asks for the incompatibility resolve at phase. Best price leads to Box 7. Precedence leads to Box 6a-3: Find lines with highest precedence. Proceed to Box 6a-4 which asks if more than one line has the same precedence. Yes leads to Box 7; no leads to Box 9.

A yes result to Box 6 leads to the second case, which begins with Box 6b-1. This box asks whether the lines are eligible for more than one asked for in the same incompatibility group. No leads to Box 9. Yes leads to Box 6b-2, that asks for the incompatibility resolve code at phase. Best price leads to Box 7. Precedence leads to Box 6b-3. which asks whether there is more than one line with the same precedence. Yes leads to Box 7; no leads to Box 9.

Box 7: Compare modifiers to find best price, leads to Box 8 that asks if there is more than one modifier with the same best price. No leads to Box 9. Yes leads to Box 8a: Select any modifier. Proceed to Box 9: Modifier selected. Proceed to Box 10, which asks if this modifier is in the exclusive group. No leads back to Box 5a; yes ends the process.

Figure 16–7 Modifier search and incompatibility processing



Precedence / Specificity

The search engine always chooses the modifier which is the most specific; the modifier with the lowest precedence number is selected.

For example, all priority customers get a 5% discount on Product Family A. Customer Boomerang Emporium, although a priority customer, has negotiated a 10% discount on the same product family and is therefore not eligible to receive the 5% discount.

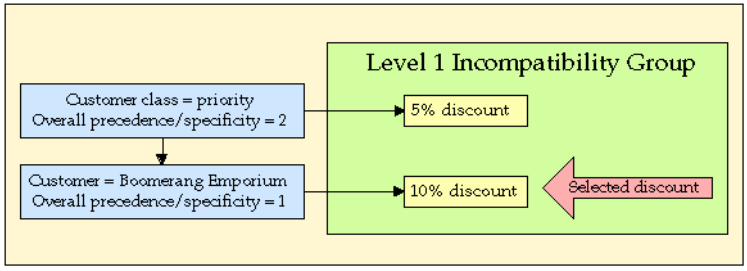
Both discounts are defined in the same incompatibility group and phase. The search engine determines that both are eligible to be applied to a Boomerang Emporium order for an item in Product Family A. Because the discounts are incompatible and both can not be applied to the order line, the search engine must determine which is the most specific discount, in this case the Boomerang Emporium negotiated discount rather than the general customer class discount. The search engine derives modifier specificity by selecting the lowest precedence number from each of the qualifiers and any product attributes on the modifier. The search engine then applies the modifier with the lowest precedence to the pricing request line. In the example given, if the precedence of the customer qualifier is 1, the precedence of the customer class qualifier is 2, and product family has a precedence of 3, then the specificity would be calculated as follows:

Table 16–6 Precedence and discount

Type of Discount	Qualifier Precedence	Pricing Attribute Precedence	Overall Precedence/ Specificity
5% discount	2 (customer class)	3	2
10% Boomerang Emporium discount	1 (customer)	3	1

The search engine selects the 10% Boomerang Emporium discount, as it has the lowest precedence (the most specific discount). This concept is illustrated in the following image.

Figure 16–8 Precedence and incompatibility



At time of setup, qualifier or product attribute precedence is defaulted from the sequence number in the qualifier and pricing descriptive flexfields. The sequence of the qualifier and product segments in and across contexts determines which qualifiers have priority when the selection engine is forced to choose between multiple prices, incompatible benefits, or charges that are eligible for application to

request line. The sequence number of each segment should be unique across the qualifier descriptive flexfield, the item context in the pricing descriptive flexfield, and the most specific qualifiers and product attributes having the lower segment sequence numbers.

For example, when defining the customer pricing hierarchy, the lowest level in the hierarchy has the lowest flexfield segment sequence number. If the search engine needed to choose between a price for an item negotiated with a specific customer and a price for the same item which was given to an entire customer class, the engine would select the customer-specific price. The customer qualifier is more specific (assigned a lower sequence number) than the customer class qualifier. This is how the customer pricing hierarchy is flattened across the qualifier descriptive flexfield and how the product hierarchy is defined in the item context of the pricing descriptive flexfield.

Best Price

Best price is an alternative method of precedence for selecting which modifier should be applied to the pricing request line when multiple incompatible modifiers are eligible. Using this method, the modifier that gives the greatest discount to the customer is selected.

This type of incompatibility resolution is used only when the monetary value of the modifier is easily estimated, and is not used for price list lines and some types of modifiers. The table below depicts this concept.

Table 16–7 Modifier types and values

Type of Modifier	Modifier Value
Discount/surcharge: percentage	List price%
Discount/surcharge: amount	Amount
Discount/surcharge: new price	List price - new price
Discount/surcharge: lumpsum	Lumpsum amount/line quantity
Price Break	Best price comparison for price break is based on estimated discount value and is list price *%.
Terms substitution	Estimated discount value of modifier such as Comparison Value. If not provided then best price is set to zero.
Item upgrade	Estimated discount value of modifier such as Comparison Value. If not provided then best price is set to zero.

Table 16–7 Modifier types and values

Type of Modifier	Modifier Value
Coupon issue	Estimated discount value of modifier such as Comparison Value. If not provided then best price is set to zero.
Other Item discount	Not included - always valued at zero
Promotional goods	Not included - always valued at zero
Freight Charges	Same as discount and surcharges.
Price list lines	Not included

The count of matched pricing attributes is considered if two or more price list lines are matched and have the same precedence. In this case the price list line with higher count is selected by the engine.

Calculation Engine

The calculation engine takes a pricing request line and its associated pricing request line details to calculate the base price, adjusted price, and/or the extended price.

Fixed Minimum Price (GSA Pricing)

Oracle Advanced Pricing enables you to set a minimum price below which the item price cannot be discounted for all, or a subset of, customers. This is commonly used to manage General Service Administration (GSA) contracts, which must ensure that commercial customers do not receive discounts equal to or greater than those fixed for customers on GSA contracts. GSA Violation is checked if the customer is a GSA violation or if the invoice location is GSA. GSA Violation is checked if the customer is a GSA violation or if the invoice location is GSA.

The minimum or GSA price for an item is set on a special GSA minimum price list; a discount list with the minimum price specified as a new (fixed) price type of discount. A GSA discount is created when the regular item price is reduced to the GSA price for a GSA customer, enabling the discount cost to be recognized. The GSA discount list is available only to GSA customers and automatically qualified by GSA = YES when created. Using attribute mapping, all orders for GSA customers are sourced with GSA = YES to ensure that these orders receive the minimum price for the item(s).

If GSA pricing is enabled and the customer is not a GSA customer, then the calculation engine determines whether the selling price for an item violates the minimum allowed price for this item. If the selling price violates the minimum price then the calculation engine returns a status of GSA violation to the request line. The calling application is responsible for error handling such as determining whether to place the order line on hold.

> **Note:** The calculation engine does not consider pricing attributes when comparing the selling price on the request line with the GSA price, therefore the GSA price can only be set for a product attribute.

In a non-GSA environment this functionality can be used to prevent pricing below the allowable minimum price. In this case the GSA price list would list all items and their possible selling price. No customers would be defined as GSA so the calculation engine would verify that the selling price of an item had not fallen below the minimum for all customers.

Calculating Price Breaks

A price break is a series of quantity delimiters to which associated prices or discounts by range of quantity, volume, value, or weight are ordered by the customer. A price break is modeled in Oracle Advanced Pricing as a set of discount, surcharge, or charge modifiers grouped through related modifiers under a price break header modifier. The price break header contains all qualifiers, product attribute, any pricing attributes (apart from those in the volume context), unit of measure, date effectivity, and other attributes that are common elements of the price break. The price break lines contain only the break unit. The break unit must be pricing attributes in the volume context of the pricing attribute descriptive flexfield.

The search engine finds any price breaks eligible for application to the pricing request line. The search engine only considers elements of the price break header, (qualifiers, product and pricing attributes) incompatibility, and whether a price break line exists to match the break unit (quantity, amount, etc.) of the pricing request line.

The calculation engine takes the price break lines and determines a base price or discount value for the modifier. The value depends on the type of price break:

- Point: Volume break in which each volume of break unit receives a base price modifier in the break range within which the total volume falls.
- Range: Volume break in which each volume of break unit gets price/discount in the break range into which it falls.
- Recurring: Volume break in which the modifier is given for each volume of break unit that falls into the break range. This is used only for modifiers.

Applying manual adjustments

The pricing engine applies any overridden manual or automatic adjustments passed in by the calling application. For more information, see: *Oracle Advanced Pricing Implementation Manual*, Integration chapter.

Pricing Engine Flow

As an aid to understanding, the following is a simplified pseudocode flow of the Oracle Advanced Pricing engine call. This flow may vary in future releases.

Invoking Application Integration Code

```
engine call preparation
    populate global record as needed in attributes mapping
    call build_Contexts for header and the line (or for every line if it is save
```

```
    or book event)
    populate engine PL/SQL record structure
call the engine (QP_PREQ_PUB.Price_Request)
```

Pricing Engine Code

```
within engine call
    clear the temporary tables
    populate temporary tables from input PL/SQL structure
for every phase of the event loop
    if pricelist phase then
        select pricelist list line in the pricelist provided by the call
        if not found then perform the secondary search
        if not found in secondary search perform big search (see Event Phases)
    end if
    if modifiers phase then
        select and insert matching modifiers into temp tables
        perform grouping, incompatibility, breaks processing
    end if
end loop
call calculation engine
populate output PL/SQL structure from temp tables
return back to the invoking application
```

Invoking Application Integration Code

- Compare the returned modifiers with existing adjustments and perform update if necessary.
- Call local calculation engine to get selling price.

For more information, see: Oracle Technical White Paper *Integrating with R11i Advanced Pricing*.

Extendibility Features

Oracle Advanced Pricing contains a number of extendibility features that are documented in the Oracle Technical White Paper *Extending 11i Advanced Pricing for Your Business - Part 1: Using Attributes Mapping*.

Oracle Advanced Pricing APIs

Oracle Advanced Pricing contains several APIs. For setup and Oracle Advanced Pricing engine calls, PL/SQL APIs are available. Sample working API calls can be downloaded by ARU and are available from Oracle*MetaLink*. These can be studied and modified for specific customer needs. For a complete listing of Oracle Advanced Pricing public APIs, see: *Oracle Manufacturing Suite APIs and Open Interfaces Manual.*

Performance Tuning Overview

Meeting performance requirements of e-businesses has been a major design objective throughout Oracle Advanced Pricing development. Significant effort has been applied to optimizing the Oracle Advanced Pricing engine call. Advanced tuning techniques have been applied in the pricing engine. There are several recommendations that, if followed, ensure optimal performance when using Oracle Advanced Pricing.

See Appendix B, "Optimal Performance" for more information on performance issues.

Diagnostics and Troubleshooting

This section contains information about the diagnosing and troubleshooting of problems in Oracle Advanced Pricing.The following table provides a summary of various methods of diagnosing and troubleshooting the results of the pricing engine.

Table 16–8 Diagnostics and troubleshooting

Diagnosing Method	When to use
Pricing Engine Request Viewer window in Sales Order Pad. How to use: The Pricing Engine Request Viewer window is available from within Oracle Order Management. The navigation path is: Sales Order window > Tools > Pricing Engine Request Viewer.	This method provides a subset of information as compared to the debug output file. It provides information such as the list line selected and deleted by the engine during processing, and the reason for deletion. It does not provide information on why the list line is not selected by the engine (qp_list_line_detail.sql is a useful mechanism for this). This is a quick method to verify the data passed to the pricing engine and data that was returned by the pricing engine. This method determines whether the expected qualifiers and pricing attribute are sourced. Since the Pricing Engine Request Viewer requests are stored in the permanent pricing debug tables, users can query previous pricing debug requests.
Debug output How to for Oracle Order Management users: Set OM: Debug Directory to directory listed in util_file_dir in init.ora. Set OM: Debug_Level to at least 5. How to for other users: Set profile QP:Debug to Yes. Set OM:Debug Directory to directory listed in util_file_dir in init.ora. Search the output file in the directory mentioned in above profile based on time-stamp.	Use this method when diagnosing why a line is not selected by the engine. This is done to compare the output of the debug with the output of qp_list_line_detail.sql. This is also useful if developers do not have access to the online windows. This also provides Oracle Order Management Integration debug messages, and provides extended debug messages for development. **Warning:** Because the Pricing Engine Request Viewer consumes a large amount of system resources, ensure it is turned off in the production system. For Oracle Order Management users make sure that the Debug Level is set to 0 and QP:Debug is No to turn it off.

Diagnosing Method	When to use
script: qp_list_line_detail.sql How to Use: Obtain the price list line ID/modifier line ID from the price list/ Modifier setup window. Open the Price List/Modifiers window. Position the Cursor on the Price List Line/Modifier Line In Question. Go to: Help > Diagnostics > Examine > Pick the LIST_LINE_ID field from the Field LOV Note the value. This list_line_id value must be provided as input to the script qp_list_line_detail.sql. Login to apps/apps@sid. Run the script $qp/patch/115/sql/qp_list_line_detail.sql to get all the price list line/modifier line information, which takes list line ID as input. Make sure that the script outputs a <list line id>.lst file.	When the price list line or modifier is not selected by the engine. The script provides information about how the price list line or modifier is setup. When the user knows the specific price list line/ modifier line that should be selected by the engine, but it is not selected. When the possibility exists that the denormalized columns are not being properly updated due to unusual user exceptions.
Verify setup using Oracle Advanced Pricing set up windows	Certain setup information has an impact on pricing engine process. Verify that the following columns have appropriate values: Incompatibility Resolve Code in event phase Search_flag in event phase Automatic_flag, active_flag in modifiers and price lists Pricing Phase in modifier Certain profile values impact pricing engine processes. Verify that the following profiles are appropriately set: QP: Get Custom Price Customized QP: Blind Discount Option QP: Verify GSA Violations QP: Return Manual Discounts
Verify engine control record and other record structure	For power users: If you call the pricing engine directly, refer to the *Oracle Manufacturing Suite APIs and Open Interfaces Manual* for examples of a pricing engine call.

Summary of Pricing Engine Messages and Diagnosis

The follow table summarizes pricing engine messages and provides an explanation and potential solutions.

Price Lists Messages and Errors

Item and UOM not on Price List

Table 16–9 Pricing Engine messages and diagnosis

Probable Cause	How to Debug
The price list header is inactive.	Check Active box on price list header.
The price list header is ineffective as of the pricing date.	Check active dates on the price list header window. Blank dates mean no restriction (pricing effectivity date can be checked in the Pricing tab of Sales Order window).
The source system code on price list is not correct.	Verify request type, source system code mapping to make sure that appropriate source system codes are attached to the request type code.
The price list line is ineffective as of the pricing date.	Check active dates on the Price List Lines window. Blank dates mean no restriction (pricing effectivity date can be checked in the Pricing tab of Sales Order window).
The qualifiers for the price list are not met or not passed to the pricing engine (attributes mapping).	Occasionally a qualifier is unintentionally created for a price list which prevents use of that price list, resulting in an error. If the price list has a qualifier, verify that the qualifier is passed to the engine.
The pricing attributes for price list are not met.	Using the Pricing Engine Request Viewer window, verify that the pricing attributes are sourced.
The price break conditions are not met based on item quantity and item amount.	Make sure that context volume and attribute line quantity is sourced properly for the descriptive flexfield Pricing Contexts. Use Pricing Engine Request Viewer window to verify this.

Table 16–9 Pricing Engine messages and diagnosis

Probable Cause	How to Debug
The product UOMs do not match.	Check the Pricing Engine Request Viewer window to make sure that the correct UOM is passed.
Pricing engine call is made before the pricing attribute in the database is saved. Verify that the line is saved before making a call to the pricing engine.	You may get an item not found on the price list error if engine is not able to find the price list due to unavailability of pricing attribute.
An unusual error causes pricing performance related columns to be out of sync.	1. Qp_list_line_detail.sql shows that the columns are not properly updated. Run the QP: Maintain de-normalized data concurrent program to correct this situation. 2. Price list header currency is different from the order currency. Check currency.

Cannot Resolve Incompatibility Between Price List X and Price list Y

Table 16–10 Cannot Resolve Incompatibility Between Price List X and Price list Y

Probable Cause	How to Debug
Engine could not use the passed-in Price List and found multiple matching price list lines while attempting to search other price lists.	In the price list window determine if there are qualifiers for the passed-in price lists. Also verify that the price list is active. If the user intends engine to use the passed in price list, then debug this issue based on suggestions in the previous table.
Engine found multiple matching price list lines with the same precedence.	Using the Pricing Engine Request Viewer window or the debug script, find the list line information of the selected list lines. Determine if one of the lines is selected unnecessarily due to missing pricing attributes. If not then update the precedence appropriately.

Invalid UOM

Table 16–11 Invalid UOM

Probable Cause	How to Debug
The pricing engine can not find the price list in the ordered UOM.	Check the UOM on the order. Open the price list window and search the price list to find out if the price is defined in the ordered UOM.
There is no other matching price list line having primary UOM flag checked.	If user intends to define price list in primary UOM and expects the engine to convert pricing quantity, then verify that the primary flag of the price list line is checked.

Invalid UOM Conversion

Table 16–12 Invalid UOM Conversion

Probable Cause	How to Debug
The ordered UOM does not match the UOM on the price list line.	If the user does not expect the pricing engine to do the conversion, then verify that the primary flag on the price list line is not set. Evaluate the reasons why the price list line with the ordered UOM is not getting selected based on the previous table: Item and UOM not on Price List.
There is no conversion defined in mtl_uom_ conversions, between ordered UOM and primary UOM.	If user expects pricing to convert the pricing quantity from ordered UOM to the pricing UOM, verify that proper conversion is defined in Oracle Inventory.

Invalid Formula, Error Returned by QP_FORMULA_PRICE_CALC_ PVT.Calculate

Table 16–13 Invalid Formula, Error Returned by QP_FORMULA_PRICE_CALC_PVT.Calculate

Probable Cause	How to Debug
User does not use NVL in the expression and a step number (formula line) has null value.	Verify that the pricing engine is selecting the expected price list line by using the debug window or debug script. Verify that the required pricing attributes are passed to the engine. If you use get_custom_price, verify that the that the function does not return a null value. If you use a factor list, verify that appropriate pricing attributes are being sourced, and that there is a matching factor line.
Formula is not a valid mathematical expression supported by the database sql.	Verify that the formula is a valid expression.
Pricing engine call is made without passing all relevant pricing attributes. Dynamic formula is attached to the price list line. However, the pricing attributes are not entered before making the pricing engine call.	Verify that the relevant pricing attributes are entered.
Formula is either not effective for the specified date or does not exist.	Verify that formula exists and effective as of pricing date.
Formula does not have at least one component.	Verify that Formula has at least one component.

Table 16–13 Invalid Formula, Error Returned by QP_FORMULA_PRICE_CALC_PVT.Calculate

Probable Cause	How to Debug
QP_CUSTOM.Get_Custom_Price( ) function does not exist or invalid in database.	Verify that function Get_Custom_Price( ) has been custom-coded in the package body of QP_CUSTOM and compiled successfully.
One of the pricing attributes is getting non-numeric values whereas it is expecting a numeric value.	a) Ensure that the pricing attributes used in the formula calculation always have the number valueset attached to it. The steps to verify are: 1. Go to Oracle Pricing Responsibility. 2. Go to Setup > Attribute Management > Context and Attributes. 3. Query the context and attribute used in the formula setup. 4. For that attribute, select if any number valueset is attached. If not attach any number valueset to it. b) Another way to check is to change the formula by making use of the function TO_NUMBER. This works only if the attribute value returned has only numeric characters. It will fail if any alphabetic characters are present in the attribute value.
There could be undefined step numbers in the formula.	Verify that all the step numbers in the formula has been defined.

Modifier Messages and Errors

Expected Modifier Not Selected by the Engine

Table 16–14 Expected Modifier Not Selected by the Engine

Probable Cause	How to Debug
Similar to message: Item and UOM not found.	Refer to previous table: *Item and UOM not on Price List*.
Qualifiers for discount list and line are not met. Qualifiers in -1 group are added to all groups.	Make sure that qualifiers in the -1 group are satisfied.
Modifier is eliminated in incompatibility.	As a test, temporarily remove the incompatibility group from the modifier line, then execute the engine call and verify that the modifier is selected. If yes then check which other modifier is selected from the same incompatibility group. Determine if the precedence must be changed. Also determine if there is any modifier in the exclusive group.
Modifier UOM is different from Pricing UOM.	Check modifier UOM. Blank UOM means that any UOM is allowed.
Modifier header currency is different from the order currency.	Check modifier currency.
Modifier is manual; it is not automatically set.	Check automatic flag on the modifier.
Asked_For flag is Y and the modifier is not asked for.	Check Asked For flag on the modifier. If the user has asked for the modifier then use debug window/output to verify that Asked For is passed to the engine. If passed, then determine whether Asked For was validated. If not passed, determine whether the qualifiers are matched. Refer to the incompatibility processing flowchart for more details.

Table 16–14 Expected Modifier Not Selected by the Engine

Probable Cause	How to Debug
Pricing phase for the modifier line is not attached to the appropriate pricing event.	Verify that the pricing phase on the modifier is attached to the appropriate event. Use caution with the event-phase setup because it can impact the pricing of the entire organization.
The qualifier, sourced item attribute, sourced pricing attribute are setup recently, however, QP Build Sourcing concurrent program does not run.	Determine in the Pricing Engine Request Viewer window/debug file whether the qualifier/pricing/item attributes are sourced. If this is a new type of attribute then run the concurrent program.
There is no record in the qp_list_header_phases.	An unusual user error can cause the qp_list_header_phases to populate incorrectly to include all phases. Run the QP: Maintain Denormalized Data concurrent program for this header to resolve the issue.

Modifiers: Incompatibility does not consider best price/precedence

Table 16–15 Modifiers: Incompatibility does not consider best price/precedence

Probable Cause	How to Debug
Incompatibility resolution code is set incorrectly.	Refer to previous table: *Item and UOM not on Price List*.
Customer has not licensed Oracle Advanced Pricing.	Oracle Advanced Pricing customers can choose to resolve the incompatibility processing by best price or by precedence. Oracle Order Management (basic pricing) customers only have the best price option.

Modifiers: Unable to override selling price/manual adjustments

Table 16–16 Modifiers: Unable to override selling price/manual adjustments

Probable Cause	How to Debug
Manual discounts are not available.	Save the order line or move the cursor out of the line and back; this causes Oracle Order Management to fire the pricing engine modifier phase.
Order level adjustments are not applied.	Order level adjustments are not applied if any lines in an order has a calculate price flag of partial price or freeze price.
The unit selling price and modifier LOVs only show unapplied manual adjustments.	Overtype the unit selling price and increase the price. This applies overridable surcharges, whereas decreasing the price applies overridable discounts.

Calculation: Back Calculation Error

This error happens when user tries to override the selling price on a quote and the pricing engine is not able to find a suitable manual overrideable adjustment to give the overridden selling price. Refer to the Integration chapter under Manual adjustments for more details.

Table 16–17 Calculation: Back Calculation Error

Probable Cause	How to Debug
No overrideable Manual adjustments available.	Check if there are any active manual overrideable adjustments.

Concurrent Program: QP: Maintains the denormalized data in QP qualifiers

Table 16–18 Concurrent Program: QP: Maintains the denormalized data in QP qualifiers

Probable Cause	How to Debug
FDPSTP failed due to ORA-06502: PL/SQL: numeric or value error: character to number conversion error.	This is caused due to mismatch in the parameters sequence. Check with Support for an ARU to correct this.
Program has been running for a long time.	This program updates rows in qp_qualifiers table. If user has selected ALL headers, then the program takes time.

Integration and Attributes Mapping Messages and Errors

Unexpected Error In Calculate_adjustments#130 User_defined Exception

Table 16–19 Unexpected Error In Calculate_adjustments#130 User-defined Exception

Probable Cause	How to Debug
QP_Attr_Mapping_PUB.Build_Contexts package is invalid due to incorrect sourcing data attributes mapping.	Check dba_errors for this package in or to determine which attribute sourcing API is causing the error. If this is a custom API then correct the API. If this is the seeded API then determine whether a correction patch is available.
Concurrent Program Build Sourcing Rules failed with error.	Execute following statement and examine the output: select text from dba_errors where name ='QP_ BUILD_SOURCING_PVT' Verify that custom sourcing causes the error.
Getting error while running Build Sourcing Rules concurrent program ORA-06502: PL/SQL: numeric or value error: character string buffer too small ORA-06512: at APPS.QP_ATTR_MAPPING_PUB, line 1445 ORA-20000: ORA-04021: timeout occurred while waiting.	This occurs when someone makes a pricing call while the concurrent program is running. Do not run the Build Sourcing Rules concurrent program when active users are calling pricing engine.
While entering the order line in sales order PAD receives an error: FND_AS_UNEXPECTED_ERROR (PKG_NAME=oe_order_adj_pvt) (PROCEDURE_ NAME=oe_line_adj.calculate_adjustments) (ERROR_ TEXT=calculate_adjustments#130 ORA-06508: PL/SQL: could not find program unit being called).	Execute following statement and examine the output: select text from dba_errors where name ='QP_ BUILD_SOURCING_PVT'; Determine whether any custom sourcing causes the errors. If the seeded sourcing rule causes this error, determine whether there is a patch available to correct the seeded rule. If the error is “Encountered the symbol ‘_’ when expecting…” then determine the relevant patch to be applied.

Freight Charges Integration Messages and Errors

Cost to Charge Conversion is not Returned by the Engine

Table 16–20 Cost to Charge Conversion is not Returned by the Engine

Probable Cause	How to Debug
There may be qualifiers attached to the freight charge header or line.	Run qp_list_line_detail.sql script for the expected freight modifier and verify that all the qualifiers are being passed to the engine.
Oracle Shipping Execution passed a different charge type than setup in the engine.	Verify that the same charge type/cost type is used in Oracle Shipping Execution and Oracle Advanced Pricing.
Pricing engine does not return a specific freight charge	Pricing engine now returns only the maximum freight charge modifier for every charge name. Make sure if the freight charge that you expect is the maximum freight charge or deactivate other freight charges higher than this freight charge or see if you can put this freight charge in a different charge name.

Common Troubleshooting Problems in Pricing Setup Screens

Price List Problems

The following is a list of possible problems and tips on how to solve them.

Problem

The price lists have a NULL value in the Multi-Currency Conversion field after the upgrade.

Suggested Action: Ensure that the concurrent program Update Price Lists with Multi-Currency Conversion Criteria has been run. Before running the program, set the profile QP: Multi Currency Installed to Y (Yes). If the concurrent program is not run, the value of the Multi-Currency Conversion field will be NULL.

When the profile QP: Multi Currency Installed is set to Y and the Price List window is open, a default multi-currency conversion record is created if no record is found for the defaulted currency code and rounding factor. This occurs when the defaulted currency is USD and the rounding factor is -2.

Problem

Problems such as no values in LOVs occur while running pricing reports in Oracle Order Management.

Suggested Action: Determine if obsolete reports are running. There are currently five reports related to Oracle Advanced Pricing:

- QPPRCST.rdf for price lists
- QPXPRFOR.rdf for pricing formulas
- QPXPRQFS.rdf for qualifier grouping
- QPXPRMLS.rdf for modifier details
- QPXPACRL.rdf for accrual details

All other reports related to pricing are no longer used.

To add a pricing report to the Oracle Order Management responsibility request group, login under System Administrator responsibility. Navigate to:

Security > Responsibility > Request

Query your request group.

To determine which request group is attached to the Oracle Order Management responsibility, navigate to:

Security > Responsibility > Define

Query the Oracle Order Management responsibility. Add the request from the Application Oracle Advanced Pricing.

Problem

Problems occurs while running copy price list, adjust price list, add items to price list, or update formula prices, through Standard Request Submission.

Suggested Action: Each of the mentioned operations is a concurrent program that has its own window that submits a request. Requests must be submitted through those forms and not Standard Request Submission. The forms have checks for mandatory parameters which are not found in the Standard Request Submission Submission window. You can locate these forms in the sub menus for Oracle Advanced Pricing.

Problem

LOV for product attribute value on the price list window does not display any values (items).

Suggested Action: Verify that the value for profile option QP: Item Validation Organization (Oracle Order Management SuperUser responsibility, Setup > Profiles) is Vision Operations. Also verify under Setup > Parameters that organization Vision Operations.

Problem

Product attribute values (when product attribute is item category) change to X automatically on list line in price list/modifier window after querying a price list/modifier window.

Suggested Action: The view mtl_categories_kfv is not properly regenerated. Re-compile the flex field for item categories.

Promotional Limits Troubleshooting

Problem

Limit Balance record not created.

Suggested Action: Check if the limit setup has the each organization check box checked. This feature is not currently supported and therefore, balance records are not created. The Limit with Limit Id as indicated in the message has multiple balances records. Keep the Organization box selected or change the limit setup. Check if modifier list for which the limit is setup is active and automatic.

For any other issues please look at the engine debug file and investigate in the limits engine code.

Promotional Limits Integration Messages

Problem 1

Limit Exceeded for Promotion Number &PROMOTION_NUMBER and Limit Number &LIMIT_NUMBER by the amount of &LIMIT_EXCEEDED_BY units.

Probable Cause: The soft limit setup for the Modifier List as indicated in the message has been exceeded and current limit balance is negative. This is an informational message.

How to Debug: User can increase the Limit available for the Modifier List in the Limits setup window.

Problem 2

Limit Exceeded for Modifier Number &MODIFIER_NUMBER and Limit Number &LIMIT_NUMBER by the amount of &LIMIT_EXCEEDED_BY units.

Probable Cause: The soft limit setup for the Modifier as indicated in the message has been exceeded and current limit balance is negative. This is an informational message.

How to Debug: User can increase the Limit available for the Modifier in the Limits setup window.

Problem 3

Modifier &OPERATOR &OPERAND for Limit Number &LIMIT_NUMBER and Promotion Number &PROMOTION_NUMBER adjusted to &PERCENT.

Probable Cause: The hard limit setup for the Modifier List as indicated in the message has been adjusted and current limit balance is zero. This is an informational message. This means that the modifier to which this limit is attached will no longer be available unless the limit is increased.

How to Debug: User can increase the Limit available for the Modifier List in the Limits setup window.

Problem 4

Modifier &OPERATOR &OPERAND for Limit Number &LIMIT_NUMBER and Modifier Number &MODIFIER_NUMBER adjusted to &PERCENT.

Probable Cause: The hard limit setup for the Modifier as indicated in the message has been adjusted and current limit balance is zero. This is an informational message. This means that the modifier to which this limit is attached will no longer be available unless the limit is increased.

How to Debug: User can increase the Limit available for the Modifier in the Limits setup window.

Problem 5

Limit Id &LIMIT has multiple balance records. A limit with no limit attributes or specific limit attributes (not 'Each' type) must have one and only one limit balance record.

Probable Cause: The Limit with Limit Id as indicated in the message has multiple balances records.

How to Debug: User must delete dubious or duplicate balance records and leaving only the correct balance record in database.

Problem 6

The variable QP_PREQ_GRP.G_ORDER_PRICE_REQUEST_CODE cannot be null. This can corrupt Promotional Limit Balances. So limits will not be consumed. Please investigate.

Probable Cause: The variable QP_PREQ_GRP.G_ORDER_PRICE_REQUEST_ CODE is being passed null value by the OM Integration code.

How to Debug: Please investigate OM/QP integration code with the help of the engine debug file.

Pricing Formula Problems

Problem

Get custom price function in a pricing formula returns null.

Suggested Action: Add customized code for the get custom price function in the QP_CUSTOM package body and not in another package. The profile option QP: Get Custom Price Customized must be set to yes (using System Administrator responsibility) if get custom price function has been customized and used in any formula.

Problem

When setting up a formula with price list line formula line, querying all values in the LOV for price list lines and scrolling through it causes the server to disconnect.

Suggested Action: This LOV has a large number of values. Issuing a query for all values in the LOV and scrolling through the values causes the server to disconnect. Use a more specific or reduced search.

Problem

When entering an order line with an item that has a formula-based price, an error message is received: use NVL around potential null formula components.

Suggested Action: This error occurs if any component in a formula evaluates to a null when the pricing engine determines the price of an item during order entry. A formula component may have a null value when it is a pricing attribute type and the value for the pricing attribute must be entered on the sales order. In this case, enter pricing attributes on the order and save. Always enter pricing attributes that are expected in the formula before proceeding to use NVL.

Pricing Organizer Problems

Scenario 1

User wants to query on modifiers that are effective from March 1, 2002 and enters the following query criteria:

Table 16–21 Modifier header tab: field names and values

Field Name in Header tab	Value
Type	Discount List
Exact Effective Date Match	Selected
Exact Effective Date Match	Selected
Effective From	01-MAR-2002
Effective To	Blank

Problem

The modifier lists on the Headers tab in the Modifiers Organizer are not returned.

Suggested Action: Since Exact Effective Date Match is checked, the query will match those modifiers with the exact effectivity dates as those in the modifier setup. The query will only return the modifiers lists (defined in the Modifier setup) which have Start Date =01-MAR-2002 and End Date = <Blank>. For Modifier Lines and

Qualifiers, the Exact Effective Date Match will compare the query criteria to the modifier setup.

Scenario 2

User wants to query on line level modifiers in the Lines tab and enters the following query criteria:

Table 16–22 Lines tab: field name and values

Field Name in Lines tab	Value
Level	Line
Formula	<ANY FORMULA>

Problem

The modifiers are not returned in query results even though there are modifier lines of Level 'Line'.

Suggested Action: By adding the query criteria of Formula=Any Formula, the query will only return those modifier lines that have any formula attached to the line AND is of Modifier Level =Line. These are 'AND' conditions. To query on just line level modifier lines, leave the Formula field blank in the query.

Scenario 3

User wants to query on modifiers in USD currency and has entered Product Attributes=Products as in the following example:

Table 16–23 Pricing Organizer tabs

Tab Name in Pricing Organizer window	Field Name	Value
Header tab	Currency	USD
Product Attributes tab	Product Attributes	Products

Problem

The modifiers returned in query results include all modifier lines that have a product attribute value.

Suggested Action: By giving Products Attribute=Product, this becomes an additional query criteria and the results will include modifier lines that have a

product attribute value. If Product Attributes=Not Specified, then the query will return all modifier lists with Currency=USD.

Scenario 4

User wants to query only on Promotions but has entered Qualifiers=No Qualifiers.

Table 16–24 Pricing Organizer tabs

Tab Name in Pricing Organizer window	Field Name	Value
Header tab	Type	Promotion
Qualifiers tab	Qualifiers	No Qualifiers

Problem

The modifiers returned in query results are those promotions that do not have qualifiers attached.

Suggested Action: If 'No Qualifiers' is selected for Qualifiers in the Qualifiers tab, only modifier lists that do not have header level qualifiers will be returned (on the Headers Tab of the Modifier Organizer.) The query will also return modifier lines that do not have any line level qualifiers attached (on the Lines Tab of the Modifier Organizer). If Qualifiers=Not Specified, then the query will return all modifier lists that are promotions, and not modifier lines.

Scenario 5

Customer wants to query on discount where General Technologies is used as a customer name qualifier

Table 16–25 Pricing Organizer tabs

Tab Name in Pricing Organizer window	Field Name	Value
Header tab	Type	Discount
Qualifiers tab	Qualifiers	Qualifiers
Qualifiers tab	Qualifier Context1	Customer
Qualifiers tab	Qualifier Attribute1	Customer Name
Qualifiers tab	Operator1	=
Qualifiers tab	Value From1	General Technologies

Problem

The query results show only Modifier Lists and not Modifier Lines.

Suggested Action: The qualifier Customer Name=General Technologies is only attached as a list level qualifier, thus the query will return modifiers list with Modifier Lists=Discount. If this qualifier is used as a line level qualifier, then the query will show all the modifier lines that have this qualifier (in the Lines tab of the Modifier Organizer).

Scenario 5

When viewing an order level charge in Order Management, the Charges window shows only the Charge Name, Type, and the charge value. The Modifier Name or modifier number is not on the window. The following example demonstrates a query for modifier lines with this Charge Name:

Table 16–26 Pricing Organizer tab

Tab Name in Pricing Organizer window	Field Name	Value
Header tab	Type	Freight and Special Charges

Problem

The query returns all modifier list=Freight and Special Charges and no modifier lines. The User still cannot find the modifier line.

Suggested Action: In order to get the specific modifier lines (in the Lines tab of the Modifier Organizer), you have to include Charge Name as a additional query criteria.

Multi-Currency Scenarios

Scenario 1

For conversion type as 'TRANSACTION', Base Currency of the Price List and Functional Currency must be same.

In Price List window:

Name = PL1

Currency = AUD

Multi-Currency Conversion = MC1

In Multi-Currency Conversion window:

Base Currency Code = AUD

Name = MC1

To Currency Code = CAD

Conversion Type = TRANSACTION

In Order Management Sales Order pad:

Price List = PL1

Currency = CAD

Conversion Type = User

Conversion Rate = 1.522

Set of Books Currency (functional currency) = USD

Problem

While placing an order, user is getting the error - For TRANSACTION conversion type, Base Currency AUD and Functional Currency USD are not same.

Suggested Action: Use a price list which has currency as USD and multi-currency conversion attached to it has a record for To Currency Code CAD and Conversion Type TRANSACTION. This occurs because the Base currency of the price list and functional currency must be same for conversion type 'TRANSACTION'.

Scenario 2

Set up the currency conversion rate in Oracle General Ledger before using conversion type of Oracle General Ledger.

In Price List window:

Name = PL2

Currency = USD

Multi-Currency Conversion = MC2

In Multi-Currency Conversion window:

Base Currency Code = USD

Name = MC2

To Currency Code = CAD

Conversion Type = TRANSACTION

In Order Management Sales Order pad:

Price List = PL2

Currency = CAD

Conversion Type = Corporate

Pricing Effective Date = 25-JUN-2002

Problem

While placing an order, user is getting the error - No conversion rate found: From Currency USD, To Currency CAD, Conversion Date 25-JUN-2002, Conversion Type Corporate.

Suggested Action: Set up the conversion rate in Oracle General Ledger for the From Currency USD, To Currency CAD, Conversion Date 25-JUN-2002 and Conversion Type Corporate. As the conversion type "Corporate" is being used in Sales Order pad, which is one of conversion types defined in Oracle General Ledger, the necessary set up must be done before placing an order.

Scenario 3

Conversion type must be passed from Sales Order pad to use multi-currency conversion of type TRANSACTION .

In Price List window:

Name = PL2

Currency = USD

Multi-Currency Conversion = MC2

In Multi-Currency Conversion window:

Base Currency Code = USD

Name = MC2

To Currency Code = CAD

Conversion Type = TRANSACTION

In Order Management Sales Order pad:

Price List = PL2

Currency = CAD

Conversion Type =

Conversion Rate =

Note: Conversion type, rate and date can be entered in sales order only when the functional currency is the same as the price list base currency.

Problem

While placing an order, user is getting the error. No conversion type is passed from OM.

Suggested Action: Provide the value for Conversion Type/Conversion Rate in Sales Order pad. As the order currency CAD is set up as conversion type TRANSACTION in multi-currency conversion list MC2, it is mandatory to pass either conversion type or conversion rate from Sales order pad.

Scenario 4: Formula Calculation Failure

In Price List window:

Name = PL2

Currency = USD

Multi-Currency Conversion = MC2

In Multi-Currency Conversion window:

Base Currency Code = USD

Name = MC2

To Currency Code = GBP

Conversion Type = FORMULA

Formula = GBPconversion

In Pricing Formulas window:

Header

Name = GBPconversion Formula = 1*2

Formula Lines:

Table 16–27 Formula Type

Formula Type	Pricing Attribute Context	Pricing Attribute	Component	Step
Pricing Attribute	Pricing Attribute	Export Cost	--	1
Numeric Constant	--	--	1.2	2

In Order Management Sales Order pad:

Price List = PL2 Currency = GBP

Value of Pricing Attribute "Export Cost" passed to pricing engine = Null

Problem

While placing an order, user is getting the error - Formula calculation failure.

Suggested Action: Pass a numeric value for Pricing Attribute "Export Cost". While using formula, make sure all the necessary information is available to correctly evaluate the formula.

Scenario 5: Formula Calculation Failure

How the effective dates work for multi-currency setup.

In Price List window:

Name = PL2

Currency = USD

Multi-Currency Conversion = MC2

Item = AS54888

Unit Price = 1000

In Multi-Currency Conversion window:

Base Currency Code = USD

Name = MC2

Table 16–28 In Multi-Currency Conversion window

To Currency Code	Effective From	Effective To	Conversion Type	Fixed Value
GBP	01-JAN-2002	31-MAR-2002	TRANSACTION	--
GBP	01-APR-2002	--	FIXED	.667

In Order Management Sales Order pad:

Price List = PL2Currency = GBP

Conversion Rate = .7

Pricing effective date = 25-JUN-2002

Item = AS54888

Problem

How the effective dates work for multi-currency setup?

Suggested Action: As per scenario V above, the unit price returned by pricing engine for item AS54888 will be 667 (1000 * 0.667) because pricing effective date is 25-JUN-2002. Conversion type .7 passed from Sales Order pad is ignored by pricing engine because the effective dates of TRANSACTION conversion type in multi-currency setup does not satisfy the pricing effective date passed from Sales order pad. Instead, the pricing engine chooses the FIXED conversion type record from multi-currency setup as it satisfies the pricing effective date.

Scenario 6

The sequence of conversion, markup and rounding operations.

In Price List window:

Name = PL2

Currency = USD

Multi-Currency Conversion = MC2

Item = AS54888

Unit Price = 1000

In Multi-Currency Conversion window:

Base Currency Code = USD

Name = MC2

Table 16–29 In Multi-Currency Conversion window

To Currency Code	Conversion Type	Fixed Value	Markup Operator	Markup
GBP	FIXED	.667	AMT	Value

In Order Management Sales Order pad:

Price List = PL2

Currency = GBP

Item = AS54888

Problem

How the markup works for multi-currency setup?

Suggested Action: From the preceding scenario, the unit price returned by pricing engine for item AS54888 will be 673 ((1000 * 0.667) + 6) because the markup will also be applied after conversion. In the process of multi-currency conversion, the list price is first converted to the order currency, then the markup is applied (if applicable), and finally, the rounding is done.

Other Technical Considerations

Pricing Engine

Verify that all the prerequisite (including server technology) patches are applied.

The pricing engine uses temporary tables; on-line patching may create a deadlock and the patch may fail.

Temporary tables are created in TEMP table space, hence TEMP table space must be sized based on size of the largest sales order data. Temporary tables are not sharable.

Troubleshooting ADPATCH Errors

Log files are written to APPL_TOP/admin/<db_name>/log, where <db_name> is the value of your ORACLE_SID or TWO_TASK variable.

For NT, the file is placed in%APPL_TOP%\admin\<db_name>\log, where <db_name> is the value of your local variable.

Review the log file for error messages after you run the utility. There may be one or more worker files if you are running steps that operate in parallel mode.

Review these adwork<number>.log files (adwork01.log, adwork02.log) for more detailed information about the errors.

Oracle 8*i* Temporary Table Locking/Patching Issues

> **Note:** While running adpatch, an attempt to alter, create, or drop an index on a temporary table already in use results in an error. If any Oracle Order Management or Oracle iStore user is pricing a line, the temporary tables are in use and adpatch encounters this error. Apply these patches only when users are not in Oracle Advanced Pricing.

Oracle Advanced Pricing patches attempt to drop and create Oracle8i temporary tables. Refer to the following instructions to verify that there are no processes accessing these tables before starting to apply the patch. The following script reveals which temporary tables are in use or locked (although the database session does not exist).

Before starting to apply the patch, the following two SQL statements should return no rows. Execute the following statement:

```
select a.sid,a.serial#,c.object_name
from all_objects c , v$lock b, v$session a
where c.object_name in
('QP_PREQ_LINES_TMP','QP_PREQ_LDETS_TMP','QP_PREQ_LINE_ATTRS_TMP',
'QP_PREQ_RLTD_LINES_TMP','QP_PREQ_QUAL_TMP')
and c.object_type = 'TABLE'
and c.object_id = b.id1
and b.sid = a.sid;
```

If the above SQL returns rows, the sessions must be ended. It is possible that the session is ended but reference to that session still exists in v$lock table. Execute the following statement:

```
select a.sid
from  v$lock a
where a.id1 in ( select b.object_id
    from all_objects b
    where b.object_name in
('QP_PREQ_LINES_TMP','QP_PREQ_LDETS_TMP','QP_PREQ_LINE_ATTRS_TMP',
'QP_PREQ_RLTD_LINES_TMP','QP_PREQ_QUAL_TMP'))
and not exists (select 'x'
from v$session c
where a.sid = c.sid);
```

If the above statement returns rows, the database must be brought down. Once the database is brought up, run the above SQL statements and verify that no rows are selected before beginning to apply the patch.

A

Windows and Navigator Paths

This appendix displays the default navigation paths for Oracle Pricing application.

Windows and Navigator Paths

Refer to the following sources for other window and application information:

- *Oracle Order Management User's Guide*
- *Oracle Application User's Guide*

The following table lists the navigation path for each window or HTML page. The term SSWA (Self Service Web Application) identifies a navigation path in the HTML user interface. Square brackets indicate a [button name or link] that must be clicked to display the window or HTML page.

Table A–1 Window Names and Navigation Paths

Window Name	Navigation Path
Add Items to Price List	Price Lists > Add Items to Price List
Adjust Price List	Price Lists > Adjust Price List
Advanced Pricing - Define Modifier	Modifiers > Modifier Setup
Advanced Pricing - Define Modifier (alternate)	Modifiers > Modifier Incompatibility Setup > [Modifiers]
Advanced Pricing - Price Lists	Price Lists > Price List Setup
Advanced Pricing - Pricing Formulas	Pricing Formulas > Formulas Setup
Advanced Search	SSWA > Oracle Pricing User >Price List Maintenance > [Advanced Search] Alternate: SSWA > Oracle Pricing Administrator > Price List Maintenance > [Advanced Search]
Assign Attributes	Setup > Attribute Management > Attribute Linking and Mapping > [Assign Attributes]
Attribute Defaulting Rules	Setup > Attribute Mapping > [Defaulting Rules...]
Bulk Create Privileges	SSWA > Oracle Pricing Administrator Responsibility > Security> Privileges > [Bulk Create Privileges]
Bulk Update Entity Usage	SSWA > Oracle Pricing Administrator Responsibility > Security> Entity Usage > [Bulk Update Entity Usage]
Context Setup	Setup > Attribute Mapping > Context and Attributes
Copy Modifiers	Modifiers > Copy Modifiers

Table A–1 Window Names and Navigation Paths

Window Name	Navigation Path
Copy Price List	Price Lists > Copy Price List
Create Entity Set	SSWA > Oracle Pricing Administrator Responsibility > Security> Entity Sets > [Create Entity Set]
Defaulting Condition Validation Templates	Setup > Attribute Mapping > [Defaulting Condition Templates...]
Defaulting Setup	Setup > Attribute Mapping
Define Limits	Modifiers > Modifier Setup > Define Modifier (W) > [List Limits] >
Define Modifier - Define GSA price	Price Lists > GSA Pricing Setup
Define Modifier Details	Modifiers > Modifier Setup > [Define Details*]
Descriptive Flexfield Segments	Setup > FlexFields
Entity Set Details	SSWA > Oracle Pricing Administrator Responsibility > Security> Entity Sets > [Set Id]
Entity Sets	SSWA > Oracle Pricing Administrator Responsibility > Security> Entity Sets
Entity Usage	SSWA > Oracle Pricing Administrator Responsibility > Security> Entity Usage
Event Phases	Setup > Event Phases
Exclude Items	Modifiers > Modifier Setup > [Exclude]
Express Create Privilege	SSWA > Oracle Pricing Administrator Responsibility Security> Privileges > [Express Create Privilege]
Factors	Pricing Formulas > Formulas Setup > [Factors]
Find Personal Profile Values	Setup > Profiles
Find Modifiers	Pricing Organizer
Find Requests	View Concurrent Requests
Find Requests (alternate)	View Concurrent Requests > [Find] > [Find Requests]
GSA Qualifiers	Price Lists > GSA Pricing Setup > [List Qualifiers]
Incompatibility Groups	Modifiers > Modifier Incompatibility Setup
Log file: request id	View Concurrent Requests > [Find] > [View Log...]

Table A–1 Window Names and Navigation Paths

Window Name	Navigation Path
Modifier Organizer	Pricing Organizer
More Pricing Attributes	Modifiers > Modifier Setup > [Pricing Attributes]
Multi-Currency Conversion List	Price Lists > Multi-Currency Conversion Setup
Oracle Pricing Lookups	Setup > Lookups
Personal Profile Values	Setup > Profiles > [Find]
Price Breaks	Pricing Agreements > Price Breaks
Price Breaks (alternate 1)	Modifiers > Modifier Setup > [Define Details*] > Price Breaks
Price Breaks (alternate 2)	Price Lists > Price List Setup > [Price Breaks]
Price List Maintenance	SSWA > Oracle Pricing Administrator Responsibility > Price List Maintenance Alternate: Oracle Pricing User > Price List > Maintenance
Pricing Agreements	Pricing Agreements
Pricing Attributes	Pricing Agreements > [Pricing Attributes]
Pricing Attributes (alternate)	Price Lists > Price List Setup > [Pricing Attributes]
Pricing Engine Request Viewer	Pricing Engine Request Viewer
Pricing Formulas	Pricing Formulas > Formulas Setup
Pricing Organizer	Pricing Organizer
Pricing Transaction Entity-Attribute Linking	Setup > Attribute Management > Attribute Linking and Mapping
Privileges	SSWA > Oracle Pricing Administrator Responsibility > Privileges
Privileges Summary	SSWA > Oracle Pricing Administrator Responsibility > Privileges > [Bulk Create Privileges] > Complete Bulk Create Privileges Step 1 to 3 > [Submit]
QUALIFIER - Header Level Qualifiers	Modifiers > Modifier Setup > [List Qualifiers] > [Cancel]
QUALIFIER - Line Level Qualifiers	Modifiers > Modifier Setup > [Line Qualifiers] > [Cancel]

Table A–1 Window Names and Navigation Paths

Window Name	Navigation Path
Qualifier Group	Qualifier Setup
Qualifier Groups - List	Modifiers > Modifier Setup > [List Qualifiers]
Qualifier Groups - Line	Modifiers > Modifier Setup > [Line Qualifiers]
Redeem Accruals	Modifiers > Accrual Redemption
Report: request id	View Concurrent Requests > [Find] > [View Output]
Request Detail	View Concurrent Requests > [Find] > [View Details...]
Request Diagnostics	View Concurrent Requests > [Find] > [Diagnostics]
Requests	View Concurrent Requests > [Find]
Segments Summary (Attachment context)	Setup > FlexFields > [Segments]
Segments Summary (alternate 1)	Setups > FlexFields > [Segments] > [New]
Segments Summary (alternate 2)	Setups > FlexFields > [Segments] > [Open]
Source Systems	Setups > Source Systems
Submit a New Request	Reports
Submit a New Request (alternate)	View Concurrent Requests > [Submit a New Request...]
Submit Request	Reports > OK
Value Sets	Setups > FlexFields > [Segments] > [Value Set]
Value Sets (alternate 1)	Setups > FlexFields > [Segments] > [New] > [Value Set]
Value Sets (alternate 2)	Setups > FlexFields > [Segments] > [Open] > [Value Set]
Update Formula Prices	Pricing Formulas > Update Formula Prices

B

Optimal Performance

This appendix provides information for optimizing Oracle Advanced Pricing performance. The following topics are discussed:

Overview

This appendix addresses implementation and setup considerations, and provides specific recommendations that improve performance from Oracle Advanced Pricing.

Oracle Advanced Pricing is designed to deliver maximum flexibility when pricing customer transactions. In e-business, transactions may be entered:

- By consumers using Oracle iStore.
- By telephone sales personnel using Oracle Telesales.
- By EDI orders received electronically into Oracle Order Management
- From other sources.

These Oracle applications integrate with Advanced Pricing in Oracle Release 11i, and use the Advanced pricing engine to price these transactions.

Each time a customer transaction is entered, the pricing engine is called to search through the applicable pricing rules - called qualifiers - that apply to this transaction. Then these rules are used to select the correct set of pricing actions including price lists, formulas, discounts, and promotions required to correctly price the transaction.

Potentially thousands of available actions may have been set up in Advanced Pricing's internal tables, so the task of the Pricing Engine to provide maximum flexibility while delivering fast performance for e-businesses is very challenging.

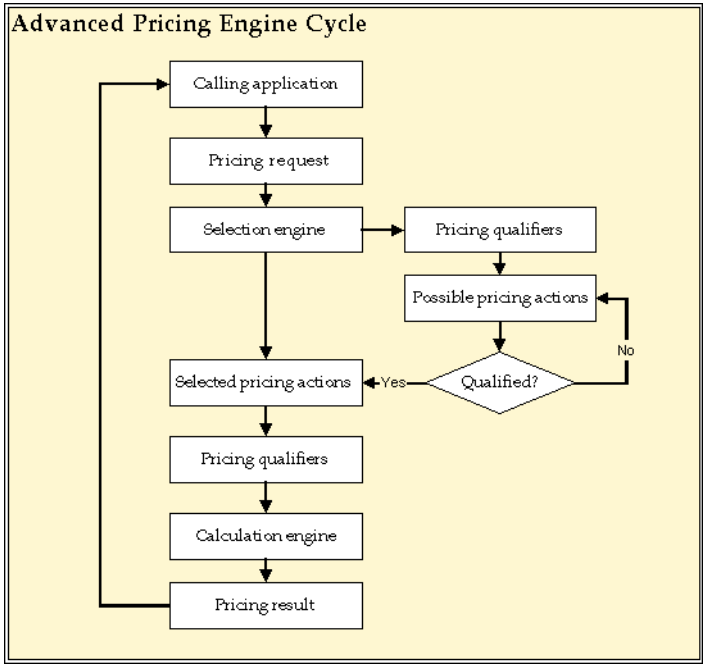
The pricing engine cycles each time a calling application makes a pricing request to the engine. Calling applications in Release 11*i* include Oracle iStore, Oracle Order Management, Oracle Contracts, Oracle TeleSales, Oracle Inventory, and Oracle Quoting.

Pricing engine performance is the amount of cycle time that elapses between when the pricing request is submitted to the engine and when the pricing result is returned.

The pricing engine runs through two types of processing activities each time it executes. First, the engine evaluates user-established qualifier rules. Based on these rules, the engine selects qualified pricing actions the calling application may need to apply to the transaction being processed. Second, the calculation engine performs the calculations necessary to compute selling price.

The following image depicts the Oracle Advanced Pricing engine cycle:

Figure B–1 Advanced Pricing Engine Cycle



Analysis of pricing engine performance characteristics reveals that the selection process is subject to more execution time variability, due to the number of records that may be selected. This chapter addresses the issue of optimizing selection engine performance.

Meeting the demanding performance requirements of e-businesses has been a major design goal. To achieve this goal, the Pricing Development team strives to optimize product performance particularly of the pricing engine.

Choices you make when selecting settings or setting up your pricing data can substantially improve the performance of the Advanced Pricing Engine. For this purpose, the implementation recommendations and considerations in the *Oracle Advanced Pricing Implementation Manual* are designed to assist you in optimizing the performance you receive from Oracle Advanced Pricing.

Oracle Advanced Pricing Setup Considerations

Analyzing your pricing data setups is the best way to improve Oracle Advanced Pricing engine performance. Removing any unnecessary qualifiers and inactivating any unnecessary, price lists, and modifiers can substantially improve pricing engine performance.

There are distributions of data in the pricing data setup, however, that can slow the pricing engine execution. The first of these is qualifier selectivity.

Qualifier Selectivity

As the pricing engine finds qualifiers that apply to a transaction, it selects all active price lists or modifier actions that the qualifier pertains to. When qualifiers identify a narrow range of pricing actions, the majority of which should be applied to the transaction, then that qualifier has high selectivity. When a single qualifier is linked to many pricing actions, the majority of which cannot be simultaneously applied to a transaction, then that qualifier is said to have low selectivity.

The new search optimizer contains the latest performance capabilities. Search optimizer introduces the ability of the pricing engine to tag the most selective qualifier within a group of qualifiers attached to the same modifier. For example, a 2% discount modifier has two qualifiers: Price list = Corporate and Customer = XYZ. Price List = Corporate is a non-selective qualifier (it is attached to many modifiers). Customer = XYZ is a selective qualifier (occurrence is low). The pricing engine first matches the most selective qualifier within the qualifier group and then matches the less selective qualifiers for the selected modifier. Search Optimizer uses the qualifier number of occurrences as criteria to tag selectivity. The pricing engine distinguishes selectivity of the qualifiers Price List = Common Price List and Price List = Customer Specific Price List.

If a modifier has qualifier as well as the product attached to it, then the pricing engine is qualifier driven rather than product driven. For example, 2% discount modifier has a qualifier attached: Price list = Corporate. It also has a product attached: Item=ABC. The pricing engine matches the qualifier first and then matches the product. At least one qualifier should be selective within the qualifier group.

Low Qualifier Selectivity Impact

Low selectivity has an adverse impact on pricing engine performance. The effect of low selectivity depends on the distribution of pricing data. The larger the data

volume when qualifier selectivity is low, the greater the negative impact on performance.

Qualifier Selectivity Examples

High Qualifier Selectivity

To illustrate the effects of selectivity, use the following business example consider how different pricing setups with high and low qualifier selectivity can be structured. Pricing engine performance implications are examined.

Qualifier Group has four qualifiers namely customer name, order type, price list, region. Customer Name is occurring five times. Therefore, it is tagged as most selective qualifier in the group. Although the other qualifiers are not highly selective, pricing engine will perform better because most selective qualifier is searched first. The following illustrates the setup with high qualifier selectivity:

Table B–1 High qualifier selectivity

Qualifier Group	Qualifier	Qualifier Count	Selectivity (Search Indicator)	Exclusivity Group	Precedence
1	Customer Name = 'ABC'	5	1	1	100
1	Order Type= Standard	300	2	1	100
1	Price List = Corporate	200	2	1	100
1	Region=Western	50	2	1	200

Low Selectivity Due to Historical Records

A company has low setup selectivity because of a large number of modifier and price list records with effectivity dates in the past. This company has been in business for several years. It has several price lists and discount records in its system, many of which are outside their effectivity dates but still in the system for historical purposes. Using the same base as in the previous example, the setup data for this example could be as follows:

Table B–2 Low Selectivity Due to Historical Records

Qualifier (Customer Class)	Qualifier Effective Dates	Price List	Effective Dates	Modifier	Exclusivity Group	Precedence
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/2001- 3/31/2001	5% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/2000 - 3/31/2000	4% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1999 - 3/31/1999	6% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1998- 3/31/1998	5% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1997 - 3/31/1997	4% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1996 - 3/31/1996	6% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1995- 3/31/1995	5% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1994 - 3/31/1994	4% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1993 - 3/31/1993	6% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1992 - 3/31/1992	4% discount	1	100
Wholesale	2/15/1991 - present	First Quarter Wholesale	2/15/1991 - 3/31/1991	5% discount	1	100

Assume that the other classes, Retail and Other, have similar data. Now examine customer All, where management has experimented with many different discount structures over time.

Table B–3 Low selectivity due to historical records

Qualifier (Customer Class)	Qualifier Effective Dates	Price List	Effective Dates	Modifier	Exclusivity Group	Precedence
All	1/01/1991 - present	Corporate	1/01/2001 - 12/31/2001	2% discount	1	200
All	1/01/1991 - present	Corporate	1/01/2000 - 12/31/2000	1.9% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1999 - 12/31/1999	1.8% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1998 - 12/31/1998	1.7% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1997 - 12/31/1997	1.6% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1996 - 12/31/1996	1.5% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1995 - 12/31/1995	1.4% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1994 - 12/31/1994	1.3% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1993 - 12/31/1993	1.2% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1992 - 12/31/1992	1.1% discount	1	200
All	1/01/1991 - present	Corporate	1/01/1991 - 12/31/1991	1% discount	1	200

In the previous example, the qualifier selectivity is very low, negatively impacting pricing engine performance. When the pricing engine executes against this data, it will find that all the historical records (those with effectivity dates that are already past) will be preliminarily qualified and selected by the Pricing Engine for further processing, even though only one price list and modifier will be finally selected to apply to the transaction. Specifically, for the historical records, the pricing engine will be forced to compare the exterior pricing date passed to it from the calling application to the effectivity date range of each record to determine if whether the selection process should proceed to the next step of considering the precedence. Since the effectivity date evaluation is a record-by-record process, large number of historical records will have an adverse impact on Pricing engine performance.

Correcting Low Qualifier Selectivity Due to Historical Records

Pricing engine performance can be improved by properly handling historical records. While the most direct route to improving engine performance is to eliminate historical records from the system, many companies must retain historical records.

Oracle Advanced Pricing provides a flag on both price list and modifier records that informs the engine whether the record is active. Because the pricing engine determines record status during the qualifier scan process, records set to inactive are automatically excluded from further consideration.

Low Qualifier Selectivity Due to Release 10.7/11 Pricing

Oracle Advanced Pricing provides more flexibility in setting up your pricing data than Release 10.7 or Release 11.

In Release 10.7 and in Release 11.0, discounts had to be associated to a price list because a price list was a mandatory qualifier for a discount. Users had to create different discounts because only one discount could be linked to one price list. In turn, price lists could be linked to either the customer, the Order Entry order type, or manually overridden on the order.

Therefore, in releases 10.7 and 11.0, many customers could potentially have large numbers of discount records, even when the number of different discrete discounts used was low.

Correcting Low Qualifier Selectivity due to Release 10.7/11 Upgrade

Reducing the number of discount records and making the qualifiers specific will help you optimize 11*i* pricing engine performance. Consider merging these discount records into as few 11*i* modifier records as possible, and then use 11*i* Pricing to tie them to customer groups.

If you examine your qualifiers and your business pricing requirement, you may find that you can qualify pricing either through the use of price lists or modifiers at the higher level of product hierarchy. For example, suppose you are selling greeting cards and have only 15 distinct prices but 100,000 items. By grouping the items into item categories, and using item category as the qualifier, you only have to create 15 price list lines rather than 100,000 lines. This results in the pricing engine searching through price list lines which substantially improves pricing engine performance.

If your business needs do not require a price list to act as a qualifier to a discount, you should delete any records that have been created as part of your upgrade process.

Low Selectivity When Qualifiers are Used as Constraints

If, for example, you have modifier lists, out of which 200 lists have Price List = Corporate as the only qualifier, this results in low selectivity of the qualifier because the pricing engine must process every list that satisfies this qualifier.

Correcting Low Selectivity When Qualifiers are Used as Constraints

If your business requires non-selective qualifiers as the only qualifiers, consider combining lists so that there are fewer lists for the engine to scan.

Redundant Qualifiers

Pricing engine performance can be boosted by avoiding qualifiers that are redundant. Here is an example of a redundant qualifier:

Customer = XYZ AND Customer Site = ABC

Because it is more specific than customer, customer site is sufficient for selection of the appropriate price list or modifiers. Adding the customer qualifier causes the unnecessary evaluation of the customer condition. By eliminating such redundant qualifiers, pricing engine performance is optimized.

Blind Modifiers

Modifiers without any qualifier or any product attached are blind modifiers. These modifiers are processed by the engine for every request line. Engine performance is negatively affected as more blind modifiers are defined in the system.

Use of ALL_ITEMS as a Product

Modifiers defined for ALL_ITEMS are processed by the engine for every request line. Engine performance is negatively affected as more ALL_ITEMS are defined in the system.

Use of Exclusions and the NOT= Operator

The pricing engine needs additional processing time to evaluate NOT = Operator in qualifiers, as well as EXCLUDE in the product hierarchy. If you have a high volume of setup data, use caution when implementing these operators.

Additional Tips for Better Performance

To improve processing, here are additional tips:

1. Always pass the price list that needs to be used for pricing, unless your business demands otherwise.
2. Try to avoid multiple price list lines from triggering incompatibility processing which eliminates ineligible list lines - this prevents unnecessary post selection processing.
3. Avoid using the same attribute for both the header and line level qualifiers. For example, if Customer = Joe's Diner is a Header level qualifier, it is not necessary to define it again as a line level qualifier.

Analyzing your Data Distributions: A Script

Oracle Advanced Pricing provides a script to analyze data distribution. The script is delivered as part of the product at $QP_TOP/patch/115/sql/qpperf.sql. If the customer is facing any performance problems then this script must be run at the SQL*Plus prompt using APPS login and the results must be provided to the pricing team in case of any performance bug logged by the customer.

The script also has hints to help customer identify the performance problems.

Note: This script is subject to change. Always check with Oracle Support to verify that you have latest version of this script.

Technical Improvements

When you implement Oracle Advanced Pricing, there are several technical measures that can be taken to ensure the best response time from the pricing engine. These measures are related to implementation-time activities.

Attribute Mapping

Oracle Advanced Pricing enables you to perform attribute mapping. Use caution when writing your own attribute sourcing. The code you write is executed for every pricing engine call. If the code is not tightly written, this can negatively impact pricing engine performance.

Pricing provides you with an option to source all the setup attributes or source attributes of only active price list, modifiers. Performance will be improved if the profile QP: Build Attributes Mapping Options is set to active only.

Phases and Events

11*i* Oracle Advanced Pricing enables pricing engine execution to be broken up into phases, and enables each phase to be associated with an event in the calling application.

If there is no need to fetch or view the discounts at the time of entering the order lines, you can modify the event phases records to execute the discounting phases at the save. This approach causes the engine to perform the price list line selections as each line is entered, while preventing the engine from doing the selection or calculation of modifiers until the save event causes the modifier selection to cycle. For users who must not view the discounts line by line as the ordered is entered, this technique can enhance pricing engine performance.

If you have unused pricing event phases, set the end date to a date in the past. This prevents pricing engine from attempting to select when the event that triggers the pricing phase occurs.

Temporary Tablespace

Oracle Advanced Pricing uses the temporary tablespace feature of Oracle8i extensively. Proper sizing of temporary tables is important to obtaining optimal performance. Depending on the size of the transaction to be priced, the initial extent for temporary tables should be sized at between 64K and 256K. The temporary tablespace should be defined as locally managed.

Memory Considerations

Because the pricing engine is frequently called during the order entry process, pricing packages must always be loaded into memory. Keep the following Pricing packages in memory:

- QP_BUILD_SOURCING_PVT
- QP_CALCULATE_PRICE_PUB
- QP_CLEANUP_ADJUSTMENTS_PVT
- QP_CUSTOM
- QP_FORMULA_PRICE_CALC_PVT
- QP_PREQ_GRP
- QP_PREQ_PUB
- QP_RESOLVE_INCOMPATIBILITY_PVT

Warning: Ensure that your shared pool size is calculated based on system use!

Performance in the Pricing Setup window

The following performance-related improvements in the setup window have been made:

- Price List window: To query a list price by product, a new find window is provided.
- Agreements window: A new find window is provided.
- Modifiers window: A list of values Customer Name, site_use, ship_to is provided.

Note: The user is responsible for the performance of the list of values, validation of the user-extended qualifiers, and pricing attributes. Therefore, ensure that the where clause in the user-defined value set is properly tuned.

Upgrade performance from Release 10.7/11 TO 11.I

Performance in upgrading the 10.7/11 pricing has been improved by providing:

- Parallel threads
- Bifurcation which enables you to upgrade the price lists and modifiers of active orders first and other data to be upgraded later at your convenience later. You may want to purge unneeded pricing data prior to the upgrade to simplify the upgrade process and improve the performance of the pricing engine.

If you purge the data, ensure that the data integrity of the transaction system is maintained. For example, you may want to purge the price list lines of the obsolete price list and upgrade just the price list header to maintain the data integrity.

Performance in applying Pricing Denormalization Patches

If you have a high volume of price list and modifier data, it may take from 5 minutes to an hour to apply pricing patches. A denormalization script is provided for the existing data. The denormalized columns are important for the pricing engine to select the list price or modifiers, and large denormalization patches have been provided with parallel threading to improve the performance.

Pricing Engine Redo Log

Users have experienced significant increase in the redo log while pricing the transactions. In the latest release, this redo is minimized by avoiding the delete operation on the temporary tables. Please check with support to ensure that you have the redo reduction patch applied.

Performance of Asked for Promotions

Users have experienced significant performance whenever asked for promotions were defined in the system. In the latest release, these issues are resolved by pre-matching the asked for Promotions. Please check with support to ensure that you have applied the latest performance patches.

Summary of Recommendations

The pricing engine is a resource intensive process which uses Oracle 8*i* temporary tables. The performance of pricing engine statements are significantly affected by inaccurate CBO settings, memory settings or any other significant high-load process. Many of the pricing performance bugs were resolved by tuning the database parameters or correcting other high-load statements. Following table lists certain common causes of performance issue.

Table B–4 Summary of Recommendations

Symptom	Probable Cause	How to fix it
1) Performance is slow and OM debug file is getting created in the debug directory	OM debug or QP debug is causing very high pl/sql processing	Make sure that Profile ONT_DEBUG_LEVEL is set to 0 (metalink Note 130511.1 has a script) And profile QP:QP_DEBUG is set to No.
2) Trace File is showing a big difference between CPU and Elapsed time	Pricing Engine Temporary tables are being thrown out of memory.	1) Some other non-pricing statement may be taking a huge amount of Logical Reads (These high-load statements would deteriorate Pricing engine performance). Execute the following statement to find high-load sql. Select sql_text, buffer_gets from v$sql where buffer_gets in (select max(buffer_gets) from v$sql); 2) Change db_block_buffers to a higher number.3--There may be CPU contention. Make sure that there are no runaway processes. Check your hardware resources.
3) Significant amount of time is spent during Pricing	Pricing Performance patches may not have been applied	Contact your support analyst to obtain any know performance fixes applicable to the specific release.
3a) Huge amount of redo is seen while pricing the transaction	Delete operations on Pricing temporary tables are generating redo	Redo is reduced by eliminating deletes. Please contact support to get the appropriate patch.
4) Trace File is showing Misses in library cache during parse	Not enough Shared Pool	Increase the value of Shared Pool parameter. Pin the frequently used Pricing Engine packages.
5) Trace file shows huge volume in the pricing temporary table qp_preq_qual_tmp	Qualifiers may be unselected which causes large number of records being selected by the engine	run qpperf.sql to see if there are any unselected qualifiers. Restructure the setup either by moving these qualifiers to Pricing attribute or by moving the qualifier from line to header. Use qpperf.sql Stmt # 10,16
6) Trace File is showing obj$ and other sys statements	SYS/SYSTEM schema is analyzed.	Do not analyze SYS/SYSTEM schema.
7) CBO is causing full table scans	CBO parameters in Init.ora may not be set properly.	Use/fnddev/fnd/11.5/admin/sql/AFCHKCBO .sql to check this).
8) Excessive calls made to the pricing engine at booking and scheduling	Pricing may be turned on at scheduling.	OM profile "OM: Deactivate Pricing at Scheduling" should be set to yes.
9) Form hangs	ST patches for temporary table may not have been applied	Check metalink Note 119542.1

Table B–4 Summary of Recommendations

Symptom	Probable Cause	How to fix it
10) unnecessary Pricing call is made at Booking	Pricing Event Phase is active for Book Event	Use Pricing Manager responsibility, invoke Event-Phase screen, Query Book event, set the end date to some prior date.
11) Unnecessary Modifiers are being fetched with every line	Automatic or Manual Modifiers are created without any qualification.	Analyze the price adjustments data to check if certain modifiers are selected for every line and are unnecessary. Inactivate these modifiers. Use qpperf.sql Stmt # 17, 27
12) Sales Order processing slow at Booking	OM performance patches may not have been applied	Check metalink for recommended performance patches.
13) Many obsolete modifier headers are being queried by Pricing Engine	Upgraded obsolete Pricing data is still active.	Inactivate the obsolete List headers by unchecking the active flag. Use qpperf.sql Stmt# 5.
14) Trace file is showing custom code getting executed many time as well as taking huge time	Inefficient Extension /custom code is being called by Attributes mapping or by get custom price. Caching not implemented in the custom code	Implement caching, tune the custom code. If you are not able to implement cache for order attributes at line level which change for every call then please look at the Order Amount sourcing as an example of caching such attributes.
15) Pricing debug screen is showing unnecessary qualifiers/ attributes being passed to the pricing engine.	Somebody has setup prototype modifies using the qualifiers and attributes which are not used by the production application	Pricing engine will get all the qualifiers, pricing attributes even if those are setup for prototype purpose. If you have setup such modifiers then inactivate these modifiers in the production instance. Then change the value of the profile 'QP_BUILD_ATTRIBUTES_MAPPING_ OPTIONS' to 'Y' and run build context concurrent program.

Table B–4 Summary of Recommendations

Symptom	Probable Cause	How to fix it
16) Many modifiers, price list lines coming into the mix in the pricing engine.	1) multiple "Not=" qualifiers without any selective qualifier in the qualifier group.	Too many lines of the same product element or same qualifier will reduce the selectivity of the engine. Evaluate alternate setup.
	2)Excessive use of "ALL_ ITEMS" product element.	If you have many "Not = " qualifiers without having any other selective qualifier in the group then performance will be affected. You can change your sourcing rule to convert the "Not = " qualifier to Equal to. For example instead of saying Customer Not= cust1 or Customer = cust2 , you can evaluate the condition in Attributes mapping if customer not in (cust1, cust2) then cust_code = 1 and use cust_ code as a qualifier.
17) Temporary Table space growing significantly. Causing huge overheads.	Initial extent of the Temp table space may be high. Pricing engine temp tables are created with big initial extents.	Change the storage clause of Temporary table space with a minimal setting of initial extent. Change the temporary table space to be locally managed.
18) Performance is slow when profile is enabled to save requests in Pricing Engine Request Viewer.	Huge data is being written into the request viewer tables.	Change the profile QP_Debug to Not store debug Log into request viewer tables. Also Purge data from Request viewer tables by using Purge Concurrent program.

Pricing Setup window

The following are performance-related improvements:

- Price List Setup window, query list price by product: A new find window is provided.
- Agreements Setup window: A new find window is provided.
- Modifiers window: A list of values customer name, site_use, ship_to is provided.
- Calling Modifier Organizer from Modifier windows is provided for faster search.

> **Note:** The user is responsible for the performance of the list of values/validation of user-extended qualifiers/pricing attributes. Verify that the where clause in the user-defined value set is properly tuned.

Global Engine Flag for Caching Attribute Mapping

On repricing, pricing engine is called once for each order. For multiple lines order, the build_sourcing is executed for every line to source all the attributes such as order amount and customer information. These attributes are therefore repeatedly sourced multiple times.

A pricing engine global flag, G_NEW_PRICING_FLAG is introduced to indicate that the attributes has been sourced for the first line so that sourcing is not required for all the following line(s). The flag has an initial value of 'Y' and build sourcing is only executed when this value is 'Y'. At the end of initial build sourcing call, this value is set to 'N' so that subsequent build sourcing is not necessary. The flag is reset back to 'N' at the end of the pricing engine call. This implementation helps to avoid unnecessary processing and hence improve the performance.

Customers who have any customized attributes can use the flag in a similar way to improve the pricing engine response time. This flag is set internally as described above. Developer can use the value of this flag to decide if re-sourcing is necessary.

Flag: QP_PREQ_GRP.G_NEW_PRICNG_CALL

Value: 'Y' or 'N'

Example of caching the order amount

```
PROCEDURE Get_Order_AMT_and_QTY (p_header_id IN NUMBER) IS
```

```
    orders_total_amt      NUMBER;
    orders_total_qty      NUMBER;
    returns_total_amt     NUMBER;
    returns_total_qty     NUMBER;

BEGIN

    SELECT SUM(nvl(pricing_quantity,0)*(unit_list_price)),
        SUM(nvl(pricing_quantity,0))
        INTO  orders_total_amt, orders_total_qty
    FROM oe_order_lines
    WHERE header_id = p_header_id
        AND (cancelled_flag = 'N' OR cancelled_flag IS NULL)
        AND (line_category_code <>'RETURN' OR line_category_code IS NULL)
            GROUP BY header_id;
        G_Order_Info.header_id := p_header_id;
    G_Order_Info.order_amount := FND_NUMBER.NUMBER_TO_CANONICAL(NVL(orders_
    total_amt,0)-NVL(returns_total_amt,0));
    G_Order_Info.order_quantity := FND_NUMBER.NUMBER_TO_CANONICAL(NVL(orders_
    total_qty,0)-NVL(returns_total_qty,0));
EXCEPTION
    WHEN no_data_found THEN
        OE_DEBUG_PUB.ADD('NO Data Found');
END;


FUNCTION Get_Order_Amount(p_header_id IN NUMBER) RETURN VARCHAR2 IS


BEGIN
    IF qp_preq_grp.g_new_pricing_call = qp_preq_grp.g_no THEN
        RETURN G_Order_Info.order_amount;
    ELSE
        Get_Order_AMT_and_QTY(p_header_id);
        RETURN G_Order_Info.order_amount;
    END IF;
END Get_Order_Amount;
```

Custom Applications Integrating with Oracle Applications

All the custom applications integrating with Oracle Applications need to integrate with the QP_PREQ_PUB.PRICE_REQUEST API instead of QP_PREQ_GRP.PRICE_REQUEST for the latest features and improved performance. For more information, see: Integrating with Oracle Advanced Pricing in *Oracle Advanced Pricing Implementation Manual*.

Performance Patches (subject to changes)

Following patches have been provided to improve the Performance of the Pricing

Engine. You should always check with support for any changes in this list of patches.

Table B–5 Performance Patches

Pricing Patch Number	Pack E	Pack F	Pack G	11*i*5	Comments
1508982	-	-	-	-	Dec 2000 - in 11i.4
1806021	X	-	-	X	May 2001
2043597	X	X	-	X	Oct 2001 - in G

C

Case Study: Pricing Scenarios in the High-Tech Industry

This case study illustrates how Oracle Advanced Pricing breaks down complex pricing scenarios into various pricing actions and rules for the high-tech industry. The following topics are covered:

Company Background

Tech Emporium is a fictitious manufacturer of high-tech gadgets for the networking industry. Their two most popular products are Brainglo and Infratimers. Both Brainglo and Infratimers are sold in a product grouping for tools items. Brainglo also belongs to the infrastructure product grouping.

Tech Emporium sells its products to two classes of customers: OEM companies and emerging growth companies. In this case study we examine orders placed from National OEM (an OEM company) and an emerging growth company called HTG (Hoping to Grow).

The following tables illustrate Tech Emporium's pricing situation. The first table depicts the price lists, discounts by customer are depicted in the next table, and discounts by products are depicted in the third table. These tables are for reference throughout this case study.

Table C–1 Example: Tech Emporium's pricing situation

Products	Corporate Price (default)	National OEM Price	HTG Price
Brainglo	$200	$175	Not applicable
Infratimers	$160	Not applicable	$150

Note: The Corporate Price List contains all items. If a product is not on a price list (marked not applicable), then the price defaults to Corporate.

Table C–2 Example: Tech Emporium's pricing situation (continued)

Customer	Customer Class	Price List	Discount	Discount Name	Exceptions
National OEM	OEM	National OEM	3%	VIP discount	Customer class is OEM
HTG Ltd.	Emerging growth	HTG	Null	Null	Null

Table C–3 Example: Tech Emporium's pricing situation (continued)

Product	Product Grouping	Discount	Exception
Brainglo	Infrastructure	5%	Customer specific price list
Brainglo	Tools	10%	Null
Infratimers	Tools	10%	Null

Problem Definition

The following section introduces several pricing scenarios.

Price List Scenario

Pricing Requirement 1

Tech Emporium wants certain customers to receive special pricing treatment for some products and standard pricing treatment for other products.

National OEM and HTG wish to place orders for Brainglo and Infratimers.

Pricing Actions and Rules: To receive the special and standard prices, pricing actions and rules are defined. Break down the pricing action into two parts.

Table C–4 Pricing requirement and rules

Pricing Requirement	Pricing Rules
Receive special price for product base on an applicable price list.	Price list is not specified on the order. Customer is eligible for a specific price list. Ordered item must appear on the specific price list.
Receive standard treatment price for other products.	Ordered item must not be on the customer's specific price list.

Discount Scenario

Tech Emporium offers discounts based on customers and product groupings.

National OEM is entitled to receive a special VIP discount of 3% for all of their orders. HTG belongs is in the emerging growth customer class; it is not eligible for the VIP customer discount.

There is a 5% discount on products in the infrastructure product group and a 10% discount on the tools product group (Brainglo is in both the Infrastructure and

Tools product group and Infratimers is only in the Tools product group). If a customer is eligible for multiple discounts based on product groupings, then they will receive the lesser of two discounts. For example, a customer can not receive a 5% and 10% discount for Brainglo.

Pricing Requirement 2

In certain cases, multiple discounts cannot be applied at the same time to the order. If a customer is eligible for both a 5% and 10% discount, then they should only get the 5% discount.

Pricing Requirement 3

Applying discounts depends on if an item has received special pricing promotions. Products in the infrastructure category get a 5% discount if the price is derived from the customer specific price list. Products in the tools category are entitled to a 10% discount.

Pricing Requirement 4

Discounts can be given based on a the structure of the customer hierarchy. Customers in the OEM customer class are entitled to 3% discount on all orders, whereas customers in HTG customer class are not entitled to any discounts.

Pricing Requirement 5

There are discounts that look at individual lines on the order. Other discounts may look at the entire order. For example, Customers in the OEM customer class are entitled to a order level VIP discount. Therefore The 5% discount and 10% discount can only be applied to the order and not individual lines.

Table C–5 Pricing actions and rules

Pricing Action	Pricing Rules
Give 5% discount	Only for the order line. Applies when user leaves the line. Applies when ordered item is from Infrastructure product group. Only when customer specific price list is selected. If there are conflicting discount, this one is selected.
Give 10% discount	Only for the order line. Applies when user leaves the line. Only when ordered item is from the Tools product group.
Give 3% discount	Only the entire order. Applies when customers are in the OEM customer class (in this case, only National OEM will receive the discount).

Applying Oracle Advanced Pricing

Now that each requirement has been divided into individual pricing actions and pricing rules, we will show you a method of implementing these rules and actions using Oracle Advanced Pricing.

Price List Setup

Setup requires two customer specific price lists and one corporate price list. Attaching a qualifier of customer name makes the price list customer specific. Customers do not specify a price list when ordering. This allows the pricing engine to search for a price list. In Oracle Advanced Pricing qualifiers direct the pricing engine towards a specific price list.

A secondary price list must be set up for the products that do not have special pricing. For this example, the Corporate Price List, which includes all of Tech Emporium's products, is the secondary (or default) price list. When the pricing engine does not find an ordered item on the primary price list it searches the secondary price list.

Discount Setup

Modifiers and Qualifiers: Other pricing actions, such as discounts, are implemented through the use of modifiers and qualifiers. Each pricing rule can be mapped to a section on the modifier form. You can also attach qualifiers to modifiers to specify the eligibility conditions for a discount.

Incompatibility Groups: A customer may be eligible for a 5% and a 10% discount. A rule is set that the customer cannot receive both discounts; the customer should only receive the 5% discount. By grouping discounts into the same incompatibility group, discounts no longer conflict. The pricing engine examines modifiers within the same incompatibility grouping level and selects only one modifier. Implementers select whether precedence or best price is used for incompatibility resolution for each pricing phase. Precedence has been selected for this case study. Both the 5% and 10% discounts are in the same incompatibility group; the pricing engine will select the modifier line with the lowest precedence value.

Pricing Phase: Incompatibility works only within a pricing phase. Discounts can be applied at different times in the order cycle. Tech Emporium applies three discounts at different times within the order cycle. The first two discounts apply when you leave the order line; this pricing phase is list line adjustment. The third discount is applied when the order is saved: this pricing phase is called all lines

adjustment. For all lines adjustment, all the lines on the order must be evaluated by the pricing engine.

Discount Example

One of Tech Emporium's pricing rules is that customers associated with the OEM customer class are entitled to a 3% order level discount. National OEM is eligible for this discount. There is no need to define an incompatibility group for this during setup because it applies in addition to other discounts.

Some discounts are at the line or group of lines level, while others are order level discounts. The 3% VIP discount is an order level discount, while the 5% and 10% product discounts are line level discounts.

5% Discount

To ensure that the discount applies when there are conflicting discounts, the precedence value must be smaller than that of the other discount. Incompatibility is set to Level 1 Incompatibility Group.

10% Discount

This discount is applied after the user leaves the order line. The incompatibility is Level 1 Incompatibility Group. The precedence value is a higher number (lower precedence).

3% Discount

The order level discount must be defined and a customer class qualifier attached. The discount always applies when the customer meets the requirement. An incompatibility group must not be defined. Customer class is the qualifier.

Results

Based on the price list and modifier setups discussed in this case study, the following tables depict how orders from National OEM and HTG appear.

The following table depicts a National OEM order:

Table C–6 National OEM order

Item	Quantity	Price List	List Price	Selling Price	Price Adjustment
Brainglo	1	National OEM	175.00	161.00	3% OEM discount 5% infrastructure products discount
Infratimers	1	Corporate	160.00	139.20	3% OEM discount 10% tools discount

The following table depicts an HTG order for the same items:

Table C–7 HTG order for the same items

Item	Quantity	Price List	List Price	Selling Price	Price Adjustment
Brainglo	1	Corporate	200.00	180.00	10% tools discount
Infratimers	1	HTG	150.00	135.00	10% tools discount

D

Case Study: Using Oracle Advanced Pricing Formulas for Healthy Fast Food

This case study illustrates how the fictitious Healthy Fast Food company uses the price list and formula capability of Oracle Advanced Pricing to price their Fresh-from-the-garden Burger. The following topics are discussed:

Introduction

Healthy Fast Food (HFF) is a chain of fast food restaurants that offers healthy fast food alternatives to traditional fast food. There are over two thousand Healthy Fast Food restaurants nationally. HFF provides eat in, take out, or home delivery of fresh vegetarian burgers and side dishes. HFF is well known for its Fresh-from-the-garden Burger: a vegetarian burger with an unlimited amount of toppings.

HFF is dedicated to freshness and quality. Every day fresh ingredients are shipped across the country to six distribution centers and then rushed to local stores.

Problem Definition

HFF sells primarily one item, the Fresh-from-the-garden Burger (the Burger). HFF wants to maintain one price list and one item on their price list for the Burger, which has no fixed price. The selling price is based upon several factors:

- Size of the Burger
- Type of bun
- Type of cheese
- Number of toppings
- Location of the store from the distribution center

The cost of toppings varies according to season and availability. The price of the Burger must change to reflect new topping prices. To minimize fluctuations in price, HFF only makes adjustments to the cost components on a monthly basis.

The final price a customer is charged for the Burger depends on the burger size, type of bun, and type of cheese. There is an extra charge for lettuce, tomato, onion, pickles, mustard, or other toppings. Each topping selected will add one cent to the final price.

There are three different burger sizes: single, double, and triple. The final price charged varies depending on the size ordered. A single adds 99 cents, a double adds $1.49, and a triple adds $1.99 to the final price.

Buns come in several types, including white, wheat, whole grain, honey wheat, and organic grain. Each of these has a different add-on value ranging from two cents for white to as high as ten cents for organic grain.

Cheese options include cheddar, jack, and gouda. These add two cents, five cents and ten cents respectively to the final price.

The customer is charged an add-on adjustment factor depending on the geographic region of the store serving the customer. For purposes of computing the store add-on, there are six different geographic areas. This can contribute 17 to 27 cents of the final price.

For example, the price of Double Fresh-from-the-garden Burger on a whole grain bun with lettuce, tomato, and gouda cheese in store A (which is in New York City) would be calculated as follows:

> Price = $1.49 (burger size) + $0.02 (two toppings at one cent each) + $0.05 (bun) + $0.10 (cheese) + $0.25 (store location charge) = $1.91

Pricing the Burger

HFF can use Oracle Advanced Pricing to price their Fresh-from-the-garden Burger.

Identify Pricing Actions

The first step in this pricing process is to identify the pricing actions that will be used for pricing. HFF wants to create a single item for the burger and have the price changed based upon the components of the burger. In Oracle Advanced Pricing, HFF can use the price list functionality and create a single item for the Burger. The item can be priced using a dynamic formula with user entered and sourced pricing attributes for input to determine the various pricing components. HFF is using two pricing actions: price lists and dynamic formula. Each one of these actions needs to be set up.

A. Set Up Price List for the Burger

1. Navigate to the price list setup form using the Oracle Pricing Manager responsibility.
2. Create a price list with the following information:
   - Price list name: HFF Corporate Price List
   - Currency: USD
3. Navigate to the list lines and enter item information for the Burger:
   - Product context: Item
   - Product Attribute: Item Number
   - Product value: Burger
   - Unit of measure: Each

- Line type: Price list line
- Application method: Unit price
- Value: null
- Dynamic formula: Fresh-from-the-garden Burger Calculation

B. Identify and Setup Pricing Components

HFF wants to maintain one item for the Burger, yet the price of the Burger is dependant on many input variables. HFF will use the formula feature to dynamically price the Burger at the time that the burger is ordered. All of these variables need to be identified and the source of their input needs to be determined.

Pricing Attributes

1. Identify all of the attributes that are used in the formula calculation:

- Size of the Burger
- Type of bun
- Type of cheese
- Number of toppings
- Store/distribution center matrix

2. Determine which attributes will be entered at the time of order and which attributes can be sourced from the order request. Pricing attributes are setup by navigating to the Pricing Context window under the Oracle Pricing Manager responsibility.

   See Appendix 10, "Attribute Management" for information on the setup for entered and sourced pricing attributes.

3. Setup entered attributes:

- Size of the Burger
- Type of bun
- Type of cheese

4. Setup sourced attributes:

- Number of toppings
- Distribution center location

- Store location

> **Note:** The sourced pricing attributes in this case study are for illustrative purposes only and do not suggest features that are available in Oracle Advanced Pricing or other Oracle Applications.

C. Set Up Formula for the Fresh-from-the-garden Burger

The Fresh-from-the-garden Burger formula calculation is HFF's most difficult pricing structure. At order entry the customer chooses the size of the burger, type of bun, type of cheese, and toppings. The store location where burger was purchased and the distribution center from where the Burger was shipped also determine the Burger's final price.

HFF uses the following equation to determine the price of the Burger:

Equation: Burger size + type of bun + type of cheese + number of toppings + distribution center/store location

1. Navigate to the pricing setup form from the Oracle Pricing Manager responsibility to create a formula for the Burger:

   Formula Name: Fresh-from-the-garden Burger Calculation

   Step: 1 + 2 + (NVL 3, 7) + (NVL 4, 75) + 6

2. Navigate to the formula lines and enter the formula detail:

Table D–1 Formula details for the Fresh-from-the-garden Burger

Formula Type	Pricing Attribute Context	Pricing Attribute	Component	Step
Factor	Null	Null	Burger Size	1
Factor	Null	Null	Type of Bun	2
Factor	Null	Null	Cheese	3
Pricing Attribute	Garden	Number of Toppings	Null	4
Numeric Constant	Null	Null	$0.01	5
Factor	Null	Null	Distribution Store Matrix	6
Numeric Constant	Null	Null	0	7

3. Navigate to Factor and fill in information for the burger size:

Table D–2 Factor details (burger size) for the Fresh-from-the-garden Burger

Base Pricing Attribute Context	Base Pricing Attribute	Operator	Value From	Value To	Adjustment Factor
Garden	Size	Equals	Single	Null	$0.99
Garden	Size	Equals	Double	Null	$1.49
Garden	Size	Equals	Triple	Null	$1.99

4. Navigate to Factor and fill in information for the type of bun:

Table D–3 Factor details (type of bun) for the Fresh-from-the-garden Burger

Base Pricing Attribute Context	Base Pricing Attribute	Operator	Value From	Value To	Adjustment Factor
Garden	Bun	Equals	White	Null	.02
Garden	Bun	Equals	Wheat	Null	.02
Garden	Bun	Equals	Whole Grain	Null	.05
Garden	Bun	Equals	Honey Wheat	Null	.05
Garden	Bun	Equals	Organic Grain	Null	.10

5. Navigate to Factor and fill in information for the type of cheese:

Table D–4 Factor details (type of cheese) for the Fresh-from-the-garden Burger

Base Pricing Attribute Context	Base Pricing Attribute	Operator	Value From	Value To	Adjustment Factor
Garden	Cheese	Equals	Cheddar	Null	.02
Garden	Cheese	Equals	Jack	Null	.05
Garden	Cheese	Equals	Gouda	Null	.10

6. Navigate to Factor and fill in information for the distribution costs:

Table D–5 Factor details (distribution costs) for the Fresh-from-the-garden Burger

Line	Attribute Context	Base Pricing Attribute	Operator	Value From	Value To	Adjustment Factor	Start Date	End Date
1	Distribution Costs	Region	Equals	Northeast		.21	01-Jan-2001	31-Jan-2001
2	Distribution Costs	Region	Equals	Northeast		.25	01-Jan-2001	31-Jan-2001
3	Distribution Costs	Region	Equals	Northeast		.27	01-Jan-2001	31-Jan-2001
4	Distribution Costs	Region	Equals	Northeast		.30	01-Jan-2001	31-Jan-2001
5	Distribution Costs	Region	Equals	Southeast		.11	01-Jan-2001	31-Jan-2001
6	Distribution Costs	Region	Equals	Southeast		.15	01-Jan-2001	31-Jan-2001
7	Distribution Costs	Region	Equals	Southeast		.17	01-Jan-2001	31-Jan-2001

Associate pricing attributes for line 1:

Table D–6 Associate pricing attributes

Associated Pricing Attribute Context	Associated Pricing Attribute	Operator	Value From	Value To
Store	Location	Equals	Baltimore	Null

Associate pricing attributes for line 2:

Table D–7 Associate pricing attributes

Associated Pricing Attribute Context	Associated Pricing Attribute	Operator	Value From	Value To
Store	Location	Equals	New York City	Null

Identify Pricing Rules

This pricing model applies to all Fresh-from-the-garden Burgers sold to any customer in any store.

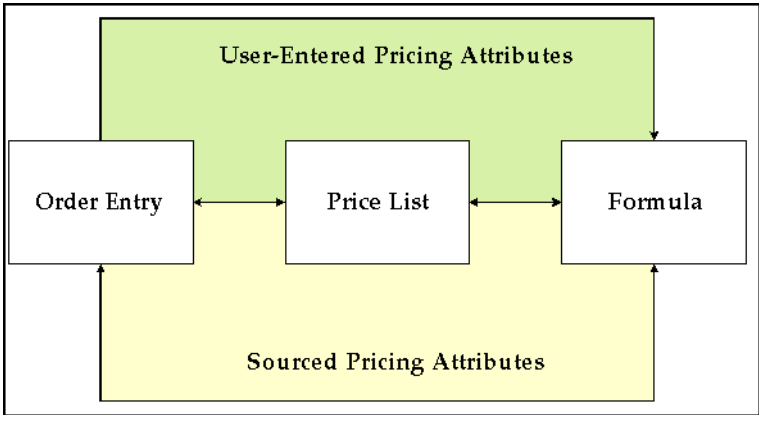
Identify Controls

HFF requires that the selling price of the Burger can change if the cost of the ingredients change. These price changes can only take effect on a monthly basis. HFF can use effectivity dates on the formula factors to control when these new costs go into effect.

Calculate the Burger Price

Now that HFF has set up the price list and formula, it can begin to use this formula to calculate the Burger. The following image illustrates the flow of information from the order entry system to price the Burger.

Figure D–1 User-entered Pricing Attributes



For example, a counter clerk in the New York City store enters the following information:

Table 16–30 Burger Order

Burger Details	Values
Size of burger	Double
Type of bun	Whole grain

Table 16–30 Burger Order

Burger Details	Values
Cheese	Gouda
Toppings	Lettuce, tomato

The Fresh-from-the-garden Burger is sent to the Oracle Advanced Pricing engine. The engine finds the price list for this item, HFF Corporate Price List, and determines that it is attached to the Fresh-from-the-garden Burger calculation. Additional information from the order entry system is necessary.

The following information is sent to the Oracle Advanced Pricing engine:

Size of burger:	Double
Type of bun:	Wheat
Cheese:	None
Number of toppings:	4

To finish completing the formula calculation, the engine needs to source the following information:

Store location:	New York City
Region:	Northeast

The engine now calculates the price of the Burger using the formula:

> Burger size (double = $1.49) + type of bun (wheat = $0.05) + type of cheese (gouda = $0.10) + number of toppings (2 = $0.02) + distribution center/store location (northeast/New York City = $0.25) = $1.91

The price of $1.91 is sent back to the order entry system.

E

Lookups

This appendix lists the predefined Oracle Pricing Lookups.

Lookups

Accessorial Charges (ACCESSORIAL_SURCHARGE)

Access Level: Extensible

Defines the seeded accessorial charges.

Table E–1 Accessorial Charges Lookup

Code	Meaning
BONDED STORAGE	Bonded Storage
BREAK-BULK	Break-Bulk
CONSOLIDATIONS	Consolidations
DOCUMENTATION	Documentation
ESD HANDLING	ESD Handling
HANDLING	Handling
IMPORT/EXPORT	Import/Export Compliance
INSPECTION	Inspection
LABELING	Labeling
LOT COMBINING	Lot Combining
MERGE	Merge
PUT-AWAY	Put-Away
RECEIVING	Receiving
SCHEDULING	Scheduling
STAGING	Staging
STOP-OFF	Stop-Off
STORAGE	Storage
TRANSLOADING	Transloading

Access Level (ACCESS_LEVEL)

Access Level: System

The Access Level lookups are used in pricing security to define the level of access granted to a price list or modifier (pricing entities).

Table E–2 Access Level Lookup

Code	Meaning
MAINTAIN	Maintain
VIEWONLY	View Only

Agreement Source Code (AGREEMENT_SOURCE_CODE)

Access Level: System

The Agreement Source code, which displays in the Pricing Agreements window, identifies whether the agreement displayed is a Pricing or Contract type of agreement.

- Contract type agreements (agreement_source_code = MCTR) are created through the Agreements public API (Business Object API or BOI) only. They can be viewed but not maintained in the Pricing Agreements window.
- Pricing type agreements (agreement_source_code = PAGR) are created and maintained in the Pricing Agreements window.

Table E–3 Agreement Source Code Lookup

Code	Meaning	Function
MCTR	Contract	Identifies Contract type agreements (agreement_source_code = MCTR) created through the Agreements public API. They can be viewed but not updated in the Pricing Agreements window.
PAGR	Pricing Agreement	Identifies Pricing type agreements (agreement_source_code = PAGR) which can be created, viewed, and maintained in the Pricing Agreements window.

Additional Service Charge (SERVICE_SURCHARGE)

Access Level: Extensible

Defines seeded service charges that can be applied.

Table E–4 Additional Service Charge Lookup

Code	Meaning
INSURANCE	Insurance
PACKAGING	Packaging
PALLETIZING	Palletizing
SHRINK WRAP	Shrink Wrapping

Agreement Type (QP_AGREEMENT_TYPE)

Access Level: Extensible

Enables the user to optionally categorize agreements by defining unique agreement types. For example, the user could set up an agreement type per contract type, or use the categorization for reporting purposes. An agreement type is optional on a pricing agreement.

The user can choose to use the seeded agreement types or add new types.

The following table lists the default (seeded) values for this lookup type.

Table E–5 Agreement Type Lookup

Code	Meaning	Function
GSA	Government Services Agreement	Used to categorize pricing agreements.
STANDARD	Standard Terms and Conditions	Used to categorize pricing agreements.
VPA	Volume Purchase Agreement	Used to categorize pricing agreements.

Arithmetic Operator (ARITHMETIC_OPERATOR)

Access Level: System

The method by which a price or modifier is calculated. Used in the Price List and Modifier Setup U.I.'s.

The following table lists the default (seeded) values for this lookup type:

Table E–6 Arithmetic Operator Lookup

Code	Meaning	Function
%	Percent	Modifier value is calculated as a per unit percentage of the List Price.
AMT	Amount	Modifier value is calculated as per unit amount +/- the List Price.
BLOCKPRICE	Block Price	Modifier value is calculated using a block price.
BREAKUNIT	Break Unit	Modifier value is calculated using a break unit which can be either a Point or Range break.
LUMPSUM	Lump Sum	Modifier value is a fixed amount, not per unit.
NEWPRICE	New Price	Modifier value overrides the selling price.
PERCENT_ PRICE	Percent Price	List Price is derived as a percentage of an associated item.
UNIT_PRICE	Unit Price	List Price is a per unit price.

Attribute Mapping Options (QP_ATTRIBUTE_MAPPING_OPTIONS)

Access Level: User

Describes the attribute mapping options.

Table E–7 Attribute Mapping Options Lookup

Code	Meaning
N	Map all attributes
Y	Map all attributes used

Calculate Price Flag (QP_CALCULATE_PRICE_FLAG)

Access Level: User

Indicates the degree to which the price is frozen.

Table E–8 Calculate Price Flag Lookup

Code	Meaning
N	Freeze price
P	Partial price
Y	Calculate price

column_type_code (COLUMN_TYPE_CODE)

Access Level: System

This lookup describes the allowed column type code.

Table E–9 column_type_code Lookup

Code	Meaning
DESC	Descriptive Flex Segment
KEY	Item Flex Segment
KEYFLEXFIELD	Descriptive Flex Segment
OEFORM	Field in OEFORM

Comparison Operator (COMPARISON_OPERATOR)

Access Level: System

Used when setting up Qualifiers and Pricing Attributes to define the rule as to how the search engine should evaluate the attribute on the request line.

The following table lists the default (seeded) values for this lookup type:

Table E–10 Comparison Operator Lookup

Code	Meaning	Function
=	Equals	Qualifier/Pricing Attribute value on the incoming request should match the Qualifier/Pricing Attribute value.
BETWEEN	Between	Qualifier/Pricing Attribute value on the incoming request should be in the range defined by the Qualifier / Pricing Attributes.
Not =	Not Equal	Qualifier Attribute value on the incoming request should NOT match the Qualifier Attribute value.

Conversion Date Type (CONVERSION_DATE_TYPE)

Access Level: System

Table E–11 Conversion Date Type

Code	Meaning
FIXED	Fixed Date Type

Table E–11 Conversion Date Type

Code	Meaning
PRICING_EFFECTIVITY_DATE	Pricing Effectively Date Type

Conversion Method (CONVERSION_METHOD)

Access Level: System

Table E–12 Conversion Method Lookup

Code	Meaning
FIXED	Fixed Conversion Method
FORMULA	Formula Conversion Method
TRANSACTION	Transaction Conversion Method

Currency Precision Type (CURRENCY PRECISION TYPE)

Access Level: System

Valid values for the profile option QP: Unit Price Precision Type. Indicates whether the currencies standard or extended precision should be used.

The following table lists the default (seeded) values for this lookup type.

Table E–13 Currency Precision Types Lookup

Precision Type	Rounding Factor
Extended	Rounding Factor is defaulted to the currencies extended precision
Standard	Rounding Factor is defaulted to the currencies standard precision

Effective Date Types (EFFECTIVE_DATE_TYPES)

Access Level: System

Effective date ranges of these types can optionally be defined on some types of modifier lists. The Search Engine will use these dates, if passed by the calling application, in addition to the pricing effective date to determine which Modifier Lists are eligible.

The following table lists the default (seeded) values for this lookup type.

Table E–14 Effective Date Types Lookup

Code	Meaning	Function
ORD	Order Date	Order Date must be within the date range.
SHIP	Requested Ship Date	Customer requested Ship Date must be within the date range.

Entity Quick Search Criteria (ENTITY_QUICK_SEARCH_CRITERIA)

Defines the entity search criteria used to query entities in pricing security.

Table E–15 Entity Quick Search Criteria

Code	Meaning
Name	Entity Name
OU	Owned By Operating Unit

Freight Charges Type (FREIGHT_CHARGES_TYPE)

Access Level: User

Table E–16 Freight Charges Type Lookup

Code	Meaning
MISCELLANEOUS	Miscellaneous Charges

Grantee Type (GRANTEE_TYPE)

Access Level: User

Grantee Type refers to the hierarchy of users to which privileges can be granted:

- Global: Includes all users with access to pricing menus.
- Operating Unit: Includes users of the named operating unit.
- Responsibility: Includes users with the specified or named responsibility.
- User: Specifies a specific named user.

Table E–17 Grantee Type Lookup

Code	Meaning
GLOBAL	Global

Table E–17 Grantee Type Lookup

Code	Meaning
NONE	None
OU	Operating Unit
RESP	Responsibility
USER	User

Incompatibility Groups (INCOMPATIBILITY_GROUPS)

Access Level: Extensible

Incompatibility Groups allow the user to define which modifiers cannot be applied to a request line with other modifiers (incompatible) and which modifiers cannot be applied to a request line with any other modifier (are exclusive).

All modifiers in a phase which are incompatible should be assigned to the same Incompatibility Groups, LVL1 - LVL3, and any modifier in a phase which is exclusive should be placed in the EXCL - Exclusive Group.

Users may define additional incompatibility groups, but only the seeded EXCL - Exclusive group is treated as "incompatible with ALL."

The following table lists the default (seeded) values for this lookup type.

Table E–18 Incompatibility Groups Lookup

Code	Meaning	Function
EXCL	Exclusive group	Incompatible with all other Modifiers in a Phase.
LVL1	Level 1 Incompatibility	Incompatible with other Modifiers in this incompatibility group in a Phase.
LVL2	Level 2 Incompatibility	Incompatible with other Modifiers in this incompatibility group in a Phase.
LVL3	Level 3 Incompatibility	Incompatible with other Modifiers in this incompatibility group in a Phase.

Incompatibility Resolution Code (INCOMPAT_RESOLVE_CODE)

Access Level: System

Methods of deciding which Modifier should be selected when multiple modifiers in the same incompatibility group are eligible to be applied to a request line in the same pricing phase. The method for resolving incompatibility is specified by

pricing phase when maintaining pricing phases in the Event to Phase Mapping Setup Up.

The following table lists the default (seeded) values for this lookup type.

Table E–19 Incompatibility Resolution Code Lookup

Code	Meaning	Function
BEST PRICE	Best Price	Search Engine selects the Modifier which gives the lowest price to the customer.
PRECEDENCE	PRECEDENCE	Search Engine selects the Modifier with the lowest precedence, i.e. the highest specificity.

Levels of Sourcing Rule for an Attribute (QP_ATTRIBUTE_MAPPING_LEVEL)

Access Level: Extensible

Define levels of the sourcing rule for an attribute.

Table E–20 Levels of Sourcing Rule for an Attribute Lookup

Code	Meaning
LINE	Order Line Level Sourcing Rule
ORDER	Order Header Level Sourcing Rule

Limit Attribute Type (LIMIT_ATTRIBUTE_TYPE)

Access Level: System

Indicates the entity of a limit dimension attribute.

The following table lists the default (seeded) values for this lookup type.

Table E–21 Limit Attribute Type Lookup

Code	Meaning
QUALIFIER	Qualifier Attribute
PRICING	Pricing Attribute
PRODUCT	Product Attribute

Limit Basis (QP_LIMIT_BASIS)

Indicates the basis on which a promotional cap limit is calculated. The following table lists the default (seeded) values for this lookup type.

Table E–22 Limit Basis Lookup

Code	Meaning
ACCRUAL	Accrual Units
CHARGE	Charge Amount
COST	Cumulative Discount
GROSS_REVENUE	Gross Revenue
QUANTITY	Item Quantity
USAGE	Usage

Limit Exceed Action (LIMIT_EXCEED_ACTION)

Access Level: System

Indicates the action to take if a applying a promotion or modifier to a sales order exceeds the promotional cap (available balance) of a promotional limit.

The following table lists the default (seeded) values for this lookup type.

Table E–23 Limit Exceed Action Lookup

Code	Meaning
HARD	Adjust the order benefit amount so that the order meets but does not exceed the promotional limit. Apply that adjusted amount to the order. Inactivate the modifier or modifier list.
SOFT	Apply the full benefit to the order and then place a promotional hold on the order.

Limit Level (LIMIT_LEVEL)

Access Level: System

Indicates how the pricing engine should maintain the promotional limit balance transactions.

The following table lists the default (seeded) values for this lookup type.

Table E–24 Limit Level Lookup

Code	Meaning
TRANSACTION	The pricing engine maintains a consumption record for each order that consumes the limit.

Table E–24 Limit Level Lookup

Code	Meaning
ACCROSS_ TRANSACTION	The pricing engine maintains one consumption record for all orders that consume the limit.

Line Type (QP_LINE_TYPE)

Access Level: User

Indicates the type of line within the pricing request.

Table E–25 Line Type Lookup

Code	Meaning
LINE	Line
ORDER	Order

List Type Code (LIST_TYPE_CODE)

Access Level: System

Used to categorize the type of list which groups price list lines or modifiers. Used for validation, including which types of lines can be included on the list, and reporting purposes.

The following table lists the default (seeded) values for this lookup type.

Table E–26 List Type Code Lookup

Code	Meaning
AGR	Agreement Price List
CHARGES	Freight and Special charge List
DEL	Deal
DLT	Discount List
PML	Price Modifier List
PRL	Standard Price List
PRO	Promotion
SLT	Surcharge List

Markup Operator (MARKUP_OPERATOR)

These values are used with multi-currency conversion lists to determine how the Markup Value (for example, 10) is applied against the base currency: either percent or amount.

Table E–27 Markup Operator Lookup

Code	Meaning
%	Percent
AMT	Amount

Miscellaneous Charges (MISCELLANEOUS)

Access Level: Extensible

Defines user-defined miscellaneous charges.

Table E–28 Miscellaneous Charges

Code	Meaning
MISC	Miscellaneous Charges
PENALTY	Charge for late payment
RESTOCKING	Restocking Fee
RETURN	Return Fee

Modifier Level Code (MODIFIER_LEVEL_CODE)

Access Level: System

Determines what qualifiers and pricing attributes are considered by the search engine when deciding if a request line qualifies for a modifier. This code also determines at what level, i.e. individual line or summary, a modifier should be applied to the request.

The following table lists the default (seeded) values for this lookup type.

Table E–29 Modifier Level Code Lookup

Code	Meaning	Function
Line	Line	Line Group

Table E–29 Modifier Level Code Lookup

Code	Meaning	Function
Line Group	Group of lines	The quantity, in the pricing UOM, and amount spent on an item is summed across all request lines. Hence the total item quantity and amount, on the request, or total quantity and amount at a level in the product hierarchy, is considered by the search engine when deciding if a modifier is qualified or not.Modifier application is at the request line level.
Order	Order	Only qualifiers or pricing attributes of the summary request line, or header, are considered by the search engine when deciding if a modifier is qualified. Note: it is not possible for a header level modifier to be qualified by a request line.Modifier application is at the summary request line, or header level.

Modifier List Line Type Code (LIST_LINE_TYPE_CODE)

Access Level: System

Defines the behavior of a List Line; a List Line maybe a Price List Line or a type of Modifier: for example, a price adjustment, benefit or charge.

The following table lists the default (seeded) values for this lookup type.

Table E–30 List Line Code Lookup

Code	Meaning	Function
CIE	Coupon Issue	Creation of a coupon which qualifies for a discount or promotional goods on a future request.
DIS	Discount	Reduces the list price, or selling of the previous pricing bucket, according to the calculation rules of the arithmetic operator.
FREIGHT_ CHARGE	Freight and Special Charges	Monetary charges which are calculated based on attributes of a request line, but which do not effect the selling price on the request line.
IUE	Item Upgrade	Substitution of one item for another on a request line, according to the pre-defined “promotional Upgrade” relationship between the two items.
OID	Other item Discount	A discount for which eligibility can be qualified by one or more request lines, but is applied to the same or different request line/s which are on the request.

Table E–30 List Line Code Lookup

Code	Meaning	Function
PBH	Price Break Header	A series of base price or price adjustments which are eligible for application to the pricing request according to a delimited break unit range and the rules of the break type.
PLL	Price List Line	Sets the base price of an item or level in product hierarchy.
PMR	Price Modifier	One or more pricing attributes, whose value or range of values is used to derive a factor on a formula line.
PRG	Promotional Goods	A discount for which eligibility can be qualified by one or more request lines, but for which a new request line is created for the discounted item.
SUR	Surcharge	Increases the list price, or selling of the previous pricing bucket, according to the calculation rules of the arithmetic operator.
TSN	Term Substitution	Changing value of qualifier attribute in terms context on request line. Seeded qualifier attributes in terms context are Freight, Shipping and Payment Terms.

Multi Currency Attribute Type (MULTI_CURR_ATTRIBUTE_TYPE)

Access Level: System

Table E–31 MULTI_CURR_ATTRIBUTE_TYPE Lookup

Code	Meaning
PRICING	Pricing Attribute
PRODUCT	Product Attribute
QUALIFIER	Qualifier Attribute ORGANIZER_FORMULA_TYPE

Organizer Formula Type (ORGANIZER_FORMULA_TYPE)

Access Level: System

Defines the formula types to be used in the Pricing Organizer.

Table E–32 Organizer Formula Type Lookup

Code	Meaning
ANY	<ANY FORMULA>

Table E–32 Organizer Formula Type Lookup

Code	Meaning
NO	<NO FORMULA> ORGANIZER_PRIC_ATTR_OPTION

Organizer Pricing Attributes Option (ORGANIZER_PRIC_ATTR_OPTION)

Access Level: System

Defines the pricing attributes option to be used in the Pricing Organizer.

Table E–33 Organizer Pricing Attributes Option Lookup

Code	Meaning
N	No Pricing Attributes
P	Pricing Attributes
S	Not Specified

Organizer Product Attributes Option (ORGANIZER_PROD_ATTR_OPTION)

Access Level: System

Defines the product attributes option to be used in the Pricing Organizer.

Table E–34 Organizer Product Attributes Option Lookup

Code	Meaning
N	No Products
P	Products
S	Not Specified

Organizer Qualifiers Option (ORGANIZER_QUAL_OPTION)

Access Level: System

Defines the qualifiers option to be used in the Pricing Organizer.

Table E–35 Organizer Qualifiers Option Lookup

Code	Meaning
N	No Qualifiers
Q	Qualifiers

Table E–35 Organizer Qualifiers Option Lookup

Code	Meaning
S	Not Specified

Price Break Type Code (PRICE_BREAK_TYPE_CODE)

Access Level: System

Rules which determine which delimited break unit range/s the qualifying break unit quantity falls into.

The following table lists the default (seeded) values for this lookup type.

Table E–36 Price Break Type Code Lookup

Code	Meaning	Function
POINT	Point	Volume break in which each volume of break unit gets price/discount in the break range into which it falls.
RANGE	Range	Volume break in which each volume of break unit gets base price/modifier in the break range within which the *total* volume falls.
RECURRING	Recurring	Volume break in which the modifier is given *for each* volume of break unit that falls into the break range. Used for modifiers only.

Price Formula Line Type Code (PRICE_FORMULA_LINE_TYPE_CODE)

Access Level: System

Defines the behavior of a formula line.Table C-12 are the defined lookups for basic pricing in OM, and Table C-13 are the lookups defined for Oracle Pricing.

The following tables list the default (seeded) values for this lookup type:

- Basic Pricing in OM
- Oracle Pricing Only

Table E–37 Price Formula Line Type Code Lookup (Basic Pricing in OM)

Code	Function	Meaning
ML	Factor List	Formula uses a price modifier list to derive the value for the formula line. A price modifier list is a grouping of price modifier lines, each line having one or more pricing attributes, whose value or range of values is used to derive a factor.
NUM	Numeric Constant	Fixed value
PRA	Pricing Attributes	Formula takes as input the pricing attribute for the item referenced by the formula line.

Table E–38 Price Formula Line Type Code Lookup (Oracle Pricing Only

Code	Function	Meaning
FUNC	Function	Formula uses a function to derive the value for the formula line
LP	Price List Line	Formula takes as input the list price of the price list line to which it is attached
ML	Factor List	Formula uses a price modifier list to derive the value for the formula line. A price modifier list is a grouping of price modifier lines, each line having one or more pricing attributes, whose value or range of values is used to derive a factor.
NUM	Numeric Constant	Fixed value
PLL	Price List Line	Formula takes as input the list price from the price list line (any price list line) referenced by the formula line.
PRA	Pricing Attribute	Formula takes as input the pricing attribute for the item referenced by the formula line.

Price Rounding (PRICE_ROUNDING)

Access Level: System

Defines the price rounding option selected. The following tables list the default values for this lookup type:

Table E–39 Price Rounding Lookup

Code	Meaning
Factor	Enforce Price List Rounding Factor
Precision	Enforce Currency Precision

These values are defined as follows:

- Enforce Price List Rounding Factor: Use the value of the Rounding Factor on the Price List for rounding the display of the value on the Price List Line. Use the value of the Rounding Factor on the Price List for rounding the engine result of the static formula calculation and displaying this value on the Price List Line
- Enforce Currency Precision: Use the currency precision for controlling the number of decimal place to display on the Price List and for rounding the engine result of the static formula calculation and displaying this value on the Price List Line.

If neither box is selected, then no limit on the number of decimal places entered on the price list and no rounding of the engine result of the static formula calculation and displaying this value on the Price List Line. Both boxes cannot be selected at the same time.

Pricing Control Flag (QP_PRICING_CONTROL_FLAG)

Access Level: User

Used by engine to identify whether the engine should just recalculate the selling price without retrieving new adjustments or calculate the selling price by retrieving new adjustments.

Table E–40 Pricing Control Flag Lookup

Code	Meaning
C	Calculate Engine
N	Search Engine
Y	Calculate & Search Engine

Pricing Events (PRICING_EVENTS)

Access Level: System

A pricing event is a "point" in the process flow of the transaction system/calling application when a call is made to the Pricing Engine (analogous to a Workflow Event).

The following seeded lookup codes are for Oracle Order Management integration with pricing; each event represents a stage in the order cycle at which pricing is performed. The information returned by pricing; base prices, price adjustments, promotions, freight charges etc. depends on the pricing phases which are processed for this event.

Note: in this release it is not possible to create new pricing events.

The following table lists the default (seeded) values for this lookup type.

Table E–41 Pricing Events Lookup

Code	Meaning	Function
BATCH	Batch Processing	Calls pricing engine when orders are processed in batch, replaces 'Line' and 'Order' events.
BOOK	Book Order	Calls pricing engine as order is booked.
FTE_APPLY_MOD	FTE: Apply Modifiers to Price	Calls pricing engine when a modifier is priced in Oracle Transportation.
FTE_PRICE_LINE	FTE:Price a Transportation Line	Calls pricing engine when a transportation line in Oracle Transportation is priced.
ICBATCH	INV: Batch Processing for Intercompany Transfer Pricing	Calls pricing engine when batch processing transaction is initiated in Oracle Inventory.
LINE	Enter Order Line	Calls pricing engine to get line level modifiers as user navigates out of a line or saves the order.
ORDER	Save Order Event	Calls pricing engine, as user saves order, to get order level modifiers and other benefits which depend on multiple order lines.
PRICE	Fetch List Price	Calls pricing engine to get base price as user enters item, quantity and unit of measure on the order line.

Table E–41 Pricing Events Lookup

Code	Meaning	Function
PRICE_LOAD	Price a Logistics Load	
REPRICE_LINE	Reprice Line	Pricing event which can be used to reprice an order line at any point during the order flow.
SHIP	Enter Shipments	Calls pricing engine as order is shipped.

Pricing Group Sequence (PRICING_GROUP_SEQUENCE)

Access Level: Extensible

A Pricing Group Sequence controls the application order of price adjustments and retrospective discounts such as accruals. The sequence of application of these modifiers becomes important when the adjustment or accrual value is derived from the selling price (the price resulting from applying prior price adjustments) rather than the list price. This is known as discounts on discounts or "cascading discounts". The sequence number of the group determines which order the calculation engine will apply the modifiers.

The pricing group sequence allows the user to place all price adjustments and retrospective discounts in a "pricing bucket"; all modifiers in a "bucket" are additive; this means that the adjustment amount for all modifiers in a bucket is calculated off the final selling price, or subtotal of the previous bucket.

The user can add additional pricing group sequences or buckets if they require further subtotals or cascading of modifiers. Pricing Group Sequence "0" is reserved for base price calculation.

The following table lists the default (seeded) values for this lookup type.

Table E–42 Pricing Group Sequence Lookup

Code	Meaning	Function
0	Base Price	Base Price calculation
1	Price Adjustments Bucket 1	First modifier subtotal
2	Price Adjustments Bucket 2	Second modifier subtotal
3	Price Adjustments Bucket 3	Third modifier subtotal

Pricing Engine Request Viewer Options (QP_REQUEST_VIEWER_OPTIONS)

Access Level: User

Used to control the options for Pricing Engine Request Viewer.

Table E–43 Request Viewer Options Lookup

Code	Meaning
N	Request Viewer Off
V	Request Viewer On, but Debug Log is not visible in Viewer
Y	Request Viewer On

Pricing Security Bulk Create Privileges Results Status (BULK_CREATE_STATUS)

Access Level: User

These values indicate the status of transaction when the Pricing Administrator creates a bulk grant in the Bulk Create Privileges page (pricing security).

Table E–44 Pricing Security Bulk Create Privileges Results Status Lookup

Code	Meaning
C	Privilege has existed and been updated correctly.
F	Failed to create the wanted privilege.
N	New privilege has been created successfully.
U	Privilege has existed and remains unchanged.

Pricing Status Code (QP_PRICING_STATUS)

Access Level: User

Indicates the returned status of the pricing engine.

Table E–45 Pricing Status Lookup

Code	Meaning
CALC	Error in calculation engine
DUPLICATE_PRICE_LIST	Duplicate price list
D_PBH	Delete in Price Break Processing
FER	Error processing formula
GSA	GSA violation
INVALID_BEST_PRICE	Could not resolve best price
INVALID_INCOMP	Could not resolve incompatibility
INVALID_UOM	Invalid unit of measure
INVALID_UOM_CONV	Unit of measure conversion not found
IPL	Invalid price list
N	New record created
OER	Other error
OTHER_ITEM_BENEFITS	Other Item Benefits
UOM	Failed to price unit of measure
UPDATED	Updated
X	Unchanged

Print on Invoice Flag (PRINT_ON_INVOICE_FLAG)

Access Level: System

This code tells whether to print a discount on an invoice or not.

Table E–46 Print on Invoice Flag Lookup

Code	Meaning
M	Print Message

Table E–46 Print on Invoice Flag Lookup

Code	Meaning
N	Don't Print Discount
Y	Print Discount

Process Code (QP_PROCESS_CODE)

Access Level: User

Used by the engine for selecting lines for calculation.

Table E–47 Process Code Lookup

Code	Meaning
D	Deleted
N	New
X	Unchanged

Proration Type (PRORATION_TYPE)

Access Level: System

Defines methods used to prorate discounts (none, category, all lines).

Table E–48 Process Code Lookup

Code	Meaning
C	Category
N	None
Y	All Lines

QP_INCREMENT_DECREMENT (QP_INCREMENT_DECREMENT)

Access Level: User

Defines the price list mass maintenance value change types.

Table E–49 QP_INCREMENT_DECREMENT Lookup

Code	Meaning
CV	Clear Value

Table E–49 QP_INCREMENT_DECREMENT Lookup

Code	Meaning
IP	Percent
IV	Amount
NV	Replace Value
XX	No Change

QP_MM_ACTION_TYPE (QP_MM_ACTION_TYPE)

Access Level: User

Defines the pricing mass maintenance action type.

Table E–50 QP_MM_ACTION_TYPE Lookup

Code	Meaning
NV	End Date and Create New
OV	Override

QP_MM_DATE_CHANGE (QP_MM_DATE_CHANGE)

Access Level: User

Defines the pricing mass maintenance date change type.

Table E–51 QP_MM_DATE_CHANGE Lookup

Code	Meaning
NO	No Change
YES	Change Date

QP_MM_FORMULA_CHANGE (QP_MM_FORMULA_CHANGE)

Access Level: User

Defines the pricing mass maintenance formula change criteria.

Table E–52 QP_MM_FORMULA_CHANGE Lookup

Code	Meaning
NO	No Change

Table E–52 QP_MM_FORMULA_CHANGE Lookup

Code	Meaning
REM_FOR	Remove Static and Dynamic Formula
REP_DYN	Replace All Formulas with Dynamic Formula
REP_STA	Replace All Formulas with Static Formula

Query Operator (QUERY_OPERATOR)

Access Level: System

Indicates the available values for a query.

Table E–53 Query Operator Lookup

Code	Meaning
=	Equal
BETWEEN	Between
LIKE	Like

Reason Code (QP_CHANGE_REASON_CODE)

Access Level: Extensible

Indicates the reason for an adjustment to a promotional limit balance. You adjust a balance by creating a consumption record against it.

The following table lists the default (seeded) values for this lookup type.

Table E–54 Reason Code Lookup

Code	Meaning
MISC	Miscellaneous

Rebate Payment Transaction Type Code (REBATE_TRANSACTION_TYPE_CODE)

Access Level: System

Defines the Rebate Payment Transaction Type Code.

Table E–55 Rebate Payment Transaction Type Code Lookup

Code	Meaning
CREDIT_MEMO	Credit Memo

Related Modifier Group Type (RLTD_MODIFIER_GRP_TYPE)

Access Level: System

Used by Oracle Pricing internally to identify relationships between, and functional groupings, of modifiers.

The following table lists the default (seeded) values for this lookup type.

Table E–56 Related Modifier Group Type Lookup

Code	Meaning	Function
BENEFIT	Benefit	Identifies those modifiers which are given as a benefit once the qualification criteria has been met.
COUPON	Coupon	Identifies the benefit which is given for a Coupon Issue.
PRICE BREAK	Price Break	Records which modifiers are price break lines for a price break.
QUALIFIER	Qualifier	Identifies those modifiers which the request must qualify for in order to get a benefit.

Relationship Type Code (QP_RELATIONSHIP_TYPE_CODE)

Access Level: User

Used in identifying the relation between the lines.

Table E–57 Relationship Type Code Lookup

Code	Meaning
BUY	Buy
DETAIL_TO_DETAIL	Detail to Detail
GENERATED_LINE	Generated Line

Table E–57 Relationship Type Code Lookup

Code	Meaning
GET	Get
LINE_TO_DETAIL	Line to Detail
LINE_TO_LINE	Line to Line
ORDER_TO_LINE	Order to Line
PBH_LINE	Price Break Header Line
RELATED_ITEM_PRICE	Related Item Price
SERVICE_LINE	Service Line

Request Type (REQUEST_TYPE)

Access Level: Extensible

A Request Type indicates to the pricing engine the type of transaction being priced. This is important to pricing, as the engine will use this information to only consider data created specifically to price this particular type of transaction.

The following seeded lookup codes are for Oracle Order Management integration with pricing. Any application which wishes to use Oracle Pricing should create a request type lookup code to identify their transaction.

The following table lists the default (seeded) value for this lookup type.

Table E–58 Request Type Lookup

Code	Meaning	Function
ASO	Order Capture	Used to price an Order Capture transaction.
FTE	Oracle Transportation Shipment	Used to price an Oracle Transportation Shipment.
IC	Inter Company Invoicing	Used to price Inter Company Invoicing.
MSD	Demand Planning	Used to price a Demand Planning transaction.
OKC	Oracle Contracts	Oracle Contracts Core
ONT	Order Management Order	Used to price an Order Management Order.

Revision Reason Code (QP_REVISION_REASON_CODE)

Access Level: User

Defines the reason for revising the agreement header and lines.

Table E–59 Revision Reason Code Lookup

Code	Meaning
NOREV	No Revision for this Agreement

Rounding Type (QP_ROUNDING_TYPE)

Access Level*:* User

Used by the engine for rounding the price.

Table E–60 Rounding Type Lookup

Code	Meaning
N	No Rounding
Q	Look at the QP Profile QP: SELLING PRICE ROUNDING OPTIONS
U	Round Price After Adding Unrounded List Price and Adjustments
Y	Round Selling Price and Adjustments

Security Control on/off (QP_SECURITY_CONTROL)

Access Level: User

Defines the available values for the profile option to turn security on or off.

Table E–61 Security Control on/off Lookup

Code	Meaning
OFF	Off
ON	On

Security Entity Type (SECURITY_OBJECT_TYPE)

Access Level: User

Defines the available entity types to which security privileges can be assigned.

Table E–62 Security Entity Type Lookup

Code	Meaning
AGR	Agreement Pricelist
MOD	Modifier
PRL	Standard Price list
SET	Pricing Entity Set

Selling Price Rounding Options (QP_ROUNDING_OPTIONS)

Access Level: User

Defines the available rounding options for the selling price.

Table E–63 Rounding Options Lookup

Code	Meaning	Description
NO_ROUND	No: unrounded listprice + unrounded adj	No Rounding: List Price and adjustments are not rounded. The selling price also is not rounded.
NO_ROUND_ADJ	Additive: round(listprice + adj); unrounded Freight	Round Selling Price after adding unrounded list price and unrounded adjustments. Freight charges are unrounded.
ROUND_ADJ	Individual: round(listprice) + round(adj)	Round Selling Price and adjustments

Source System (SOURCE_SYSTEM)

Access Level: Extensible

Defines the seeded source systems used when setting up Pricing Transaction Entities.

Table E–64 Source System Lookup

Code	Meaning
AMS	Oracle Marketing
ASO	Oracle Capture
FTE	Oracle Transportation
INV	Oracle Inventory
OKC	Oracle Contracts
QP	Oracle Pricing

Surcharges Type (SURCHARGES_TYPE)

Access Level: Extensible

Defines the seeded types of surcharges

Table E–65 Surcharges Type Lookup

Code	Meaning
ACCESSORIAL	Accessorial Charge
SERVICE_SURCHARGE	Additional Service Charge

Time UOM Conversion (QP_TIME_UOM_CONVERSION)

Access Level: System

Defines the the options to set Time UOM Conversion either by Standard or by Oracle Contracts.

Table E–66 Time UOM Conversion Lookup

Code	Meaning
ORACLE_ CONTRACTS	OKS Time Conversion
STANDARD	Standard

Types of Context (QP_CONTEXT_TYPE)

Access Level: Extensible

Defines the type of available contexts.

Table E–67 Types of Context Lookup

Code	Meaning
PRICING_ATTRIBUTE	Pricing Context
PRODUCT	Product Context
QUALIFIER	Qualifier Context

Types of Pricing Transaction Entities (QP_PTE_TYPE)

Access Level: Extensible

Indicates the type of pricing transaction entity.

Table E–68 Types of Pricing Transaction Entities Lookup

Code	Meaning
DEMAND	Demand Planning
INTCOM	Intercompany Transaction
LOGSTX	Logistics
ORDFUL	Order Fulfillment

Types of Segment Levels (QP_SEGMENT_LEVEL)

Access Level: Extensible

Indicates the level at which attribute can be sourced.

Table E–69 Types of Segment Levels Lookup

Code	Meaning	Description
BOTH	Order Header As Well As Order Line Sourcing	Attribute Sourced at Order Header as well as Order Line
LINE	Order Line Sourcing Only	Attribute Sourced at Order Line only
ORDER	Order Header Sourcing Only	Attribute Sourced at Order Header only

Usage Pricing Type (QP_USAGE_PRICING_TYPE)

Access Level: User

Indicates the usage pricing type.

Table E–70 Usage Pricing Type Lookup

Code	Meaning
REGULAR	Regular
BILLING	Billing
AUTHORING	Authoring

Ways of Sourcing an Attribute (QP_SOURCING_METHOD)

Access Level: User

The sourcing method indicates different ways an attribute can be sourced.

Table E–71 Source System Lookup

Code	Meaning
ATTRIBUTE MAPPING	Sourcing Rule Needs To Be Defined.
CUSTOM SOURCED	Actual Sourcing Code Provided: No Sourcing Rule Needed
USER ENTERED	Attribute Sourced While Pricing An Order

Yes No (YES_NO)

Access Level: System

Defines the seeded Yes and No values.

Table E–72 Yes No Lookup

Code	Meaning
No	No
Yes	Yes

Glossary

A

Access Level

Provides Maintain or View-Only access to a pricing entity:

- View-Only: Enables the user to view but not update the pricing entity.
- Maintain: Enables the user to view and update pricing entities. Not all of the entities support delete capabilities.

accrual

In Oracle Pricing are monetary or non-monetary units that are earned and accumulated for later remittance in a form of a monetary or non-monetary payment. Remittance settlement is based on a predefined basis for performance.

ask for modifier

A promotion modifier type that requires a customer to specifically request the promotion on the order.

attribute

A basic data element used by Oracle Pricing to control pricing activity. For example, Pricing uses attributes to define the eligibility of a customer order to receive a particular price or modifier. In Oracle Pricing, individual attributes are obtained from data sources that are called contexts. Pricing attributes may also be used as elements of a pricing formula.

automatic modifier

In Oracle Pricing, a control that allows you to specify that the Pricing Engine apply a modifier automatically to a transaction, assuming that the transactions meets the qualifier eligibility.

B

base price

The original price for an item obtained from the Price List; the price before any price adjustments are applied. Also known as List Price.

benefits

In Oracle Pricing, modifiers that result in non-monetary adjustments to an order. Types of benefits include: Item upgrade, free items, change in payment, or shipment terms. **See also Price Adjustments.**

best price

An alternative method to precedence which is used to determine which modifier should be selected when multiple modifiers in the same exclusivity or incompatibility group are eligible to be applied to the same pricing line within a pricing phase. The modifier which gives the lowest price or most advantageous price to the customer on the given pricing line will be applied. **See also Precedence.**

buckets - pricing

A feature in Oracle Pricing that determines how modifier price adjustments are applied to the list price of an item to calculate the selling price. Modifiers use the previous buckets sub-total for percentage calculation. Modifiers within the same bucket are Additive i.e. added together, and subtracted from the previous buckets total. The user can create an unlimited amount of buckets to calculate selling price. For example, discounts associated with bucket 0 use list price as their calculation basis. Bucket 1 prices use the subtotal resulting from subtracting bucket 0 discounts from list price as their calculation. Bucket 2 uses the subtotal remaining after subtracting bucket 1 discounts from the bucket 0 subtotal, and so forth.

C

calculation engine

In Oracle Pricing the calculation engine is a part of pricing engine. It returns the price of an item/service.

CTD (Capable to Deliver)

Refers to considering the transportation resources and transportation lead time to meet your customers delivery needs. In this release, only transportation lead time is being considered. Transportation resources will be added in a future release.

Chained Discounts

See Compound Discounts, also Buckets, Pricing.

compound discounts

Discounts that are applied on top of already discounted prices. **See buckets, pricing.**

context

A data source used by Oracle Pricing to obtain attributes. For example, Oracle Pricing considers the customer order itself to be a context. Various attributes from the order, such as order date, can be used by Pricing to control the selection of a price or modifier. Oracle Pricing does not limit the number of contexts that can be defined. Any single context can supply up to 100 pricing attributes. **See also Dimension.**

customer class

A method to classify and group your customers. For example, you could group them by their business type, size, or location. You can create an unlimited number of customer classes.

D

deal

A Modifier List type in Oracle Pricing that is a child of the modifier list type Promotion. A Deal is a group of modifiers that share the same header level qualifications and are reported together. A Deal is tied to a Promotion for reporting purposes only.

defaulting

Used in Oracle Pricing to identify the supply of a value for an attribute. **See sourcing.**

defaulting condition

Used in Oracle Pricing to refer to the context/dimension name as well as the source system. Each attribute is defaulted differently according to which context is being defaulted.

dimension sourcing

Provides the rules for finding attribute values that have been defined in qualifier and pricing dimensions. For a given transaction, the Pricing Engine must receive all the values of the attributes defined in the qualifier and pricing dimensions (contexts) in order to determine which prices and/or benefits the transaction is qualified for. **See context.**

discounts

Is a Modifier type in Oracle Pricing that creates Pricing Adjustments which allows Pricing Engine to extend a reduced price for an order, specific line item, or group of lines.

E

effective dates

Start date and end date that a price, discount, surcharge, deal, promotion, or change is active.

entity set

A set of pricing entities that can be used as an Entity Type to which you can grant privileges with Maintain or View-Only access levels.

entity type

A term used to describe one of the following pricing entities: Standard Pricelist, Modifier List, Pricing Agreement, and Entity Set.

entity usage

Grants the entity's usage to one or all operating units so it can be used during pricing engine calls.

exclusivity

A feature in Oracle Pricing that determines if an order qualifies for multiple modifiers on an order, and if one of the modifiers is exclusive, then only the exclusive modifier is applied.

F

factor list

In Oracle Pricing Formulas, factor lists may be linked to multiple pricing attributes or range of these attributes. For example, a glass with thickness between 0.1 to 0.3 mm will have a factor of 3 and a glass with thickness between 0.4 to 0.8 mm will have a factor of 5. The choice of factors from the factor list is made by the Pricing engine at the time the formula is computed. **See formula.**

formula

A mathematical formula used in Oracle Pricing to define item pricing or modifier adjustments. You create a pricing formula by combining pricing components and assigning a value to the components.

G

global usage

When Global Usage is set to Yes for a pricing entity, it can be used across all operating units for processing orders. If No is selected, the entity's usage is restricted to the operating unit that created or owns it.

When security is turned on, a Global box indicating Global Status is dynamically added to the header region of all price lists and modifiers. A user with Maintain access privileges can update the Global box. The Oracle Pricing Administrator can also update the Global Usage settings in the Entity Usage pages.

grantee

The specific user or users for a Grantee Type that are given permission to view or maintain a pricing entity. Used in combination with a Grantee Type.

grantee type

The level to which privileges are granted:

- Global: Includes all users with access to pricing menus.
- Operating Unit: Includes users within the named operating unit.
- Responsibility: Includes users within the named responsibility.
- User: Specifies a named user.

GSA (General Services Administration)

GSA (General Services Administration): a customer classification that indicates the customer is a U.S. government customer. For products on the GSA price list, a fixed price must be used, defined on the GSA contract. The items contained on the GSA price list cannot be sold to commercial customers for the same or less price than the government price. In other terms, the price offered to the government must be the minimum in the market.

I

Incompatibility

A feature in Oracle Pricing that allows you to define groups of modifiers where the modifiers in a group are incompatible with each other. Modifiers in the same incompatibility group may not be used together on the same transaction. **See Precedence.**

L

list price

In Oracle Pricing, the base selling price per unit of the item, item category or service offered. You define the list price on a price list. All price adjustments are applied against the list price.

list price (Pricing Formula)

A Formula type used in Oracle Pricing to notify the pricing engine to use the Price on the line that the formula is attached to for calculating the formula.

list price (Price List)

In Oracle Pricing, the base selling price per unit of the item, item category or service offered. You define the list price on a price list. All price adjustments are applied against the list price.

M

modifier

Defines the terms of how Oracle Pricing will make adjustments. For example, a modifier can take the form of: discounts, or surcharges. In Oracle Pricing, when you setup modifiers, you define the adjustments your customers may receive. You

control the application of modifiers by the pricing engine by also setting up rules that specify qualifiers and attributes governing their use.

modifier list

A grouping of modifiers in Oracle Pricing.

For the rest of the document, we will use 'Multilevel ATP' as a short form for this feature.

P

precedence

Used by Oracle Pricing to resolve Incompatibility. Precedence controls the priority of modifiers and price lists. If a customer qualifies for multiple modifiers that are incompatible with each other, precedence determines the discount that the customer is eligible for based on the precedence level of the modifier. **See also incompatibility.**

price adjustments

In Oracle Pricing, modifiers that result in monetary adjustments to an order. Types of benefits include: discount in %, amount or new price, discounts on other items, surcharges. **See benefits.**

price breaks

A quantity delimiters to which are associated prices or discounts by range of quantity and amount. These breaks may occur at either the line item level or at the total order level and associated with a item/ item group.

price list

A list containing the base selling price per unit for a group of items, item categories or service offered. All prices in a price list are for the same currency.

pricing date

In Oracle Advanced Pricing, the date according to which selling price will be calculated. Pricing Date can be Order Date, Ship Date or sysdate.

pricing entity

A pricing entity can be a price list, modifier list, or pricing agreement.

pricing entity security

The highest level of security administration for Oracle Pricing. This level of security is in addition to Functional Security and PTE plus Source System Code security. Functional Security is established for each user by responsibility set up. The Oracle Pricing Administrator is a new Responsibility which has complete access to all pricing entities without restriction and is used for global administration of secured access to pricing entities. This security is administered in the Oracle HTML user interface.

pricing event

A point in the process flow of a Pricing Line, e.g. Order, Contract etc. at which a call is made to the Pricing Engine to fully or partially price the Order or Contract. A pricing event is analogous to a Workflow event. Ex: Booking Order, Reprice Order, Shipping Order etc.

pricing phase

A user defined control that can be applied to Price Lists, Discount Lists, Promotions, Deals, etc. In Oracle Pricing, the Pricing Engine looks at the phase when deciding which lists should be considered in a Pricing Event. **See Pricing Event.**

pricing request structure

In Oracle Pricing the Pricing Request Structure is the parameter information that is passed to the Pricing Engine for calculating the final price which is inclusive of all the modifiers that the customer is eligible for.

product pricing attributes

In Oracle Pricing, these define the products referenced in a deal, discount, promotion, or price list.

promotion

A Modifier List type in Oracle Pricing. A Promotion is a group of modifiers that share the same header level qualifications and are reported together.

Q

qualifier

Used in Oracle Pricing to determine eligibility for a price, price adjustment or benefit.

R

request type

Used by Oracle Pricing to identify the transaction system which calls the Pricing Engine for calculation of the price. Ex: Order Management, Order Capture, Oracle Service, Contracts etc.

S

selection engine

In Oracle Pricing the selection engine is a part of pricing engine. Selection engine is responsible for selecting price lists and modifiers based on the Pricing Request.

selling price

Selling Price is defined as the price derived after applying price adjustments to the list price. The selling price is the unit cost for a particular item. Thus, if two of item A cost $10.00 each, the selling price is $10.00 for each unit.

source transaction system

The source application that populates the pricing tables with information using the Pricing API's. These include: Ex: Order Management, iMarketing, Trade Management, Contracts, etc.

sourcing

Used in Oracle Pricing to refer to the supply of a value for an attribute. **See defaulting and dimension sourcing.**

Suggested Order Date

The date that the planning process suggests an order for goods or services is entered. The earliest order date allowed is today and no compression days are allowed.

Index

A

B

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

W