

# Oracle® iStore

Implementation Guide

Release 11*i*

May 2001

Part No. A90344-01

Copyright © 2001, Oracle Corporation. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

**Restricted Rights Notice** Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark. Other names may be trademarks of their respective owners.

---

---

# Contents

<b>Send Us Your Comments</b> .....	xiii
<b>Preface</b> .....	xv
Intended Audience .....	xv
Related Documents.....	xv
Conventions.....	xvi
Documentation Accessibility .....	xvi
Accessibility of Code Examples in Documentation.....	xvii
<b>1 Oracle iStore 11i Overview</b>	
1.1 Product Overview .....	1-2
1.2 Architectural Overview .....	1-3
1.3 Hardware Requirements .....	1-4
1.4 Software Requirements .....	1-5
<b>2 Oracle iStore 11i Dependency Setups</b>	
2.1 Setup Process Flow Diagrams .....	2-2
2.2 Dependency Requirements.....	2-5
2.2.1 Mandatory Modules .....	2-5
2.2.2 Optional Modules .....	2-7
2.3 Setting Up the Mandatory Dependencies.....	2-9
2.3.1 Setting Up Languages in AOL .....	2-9
2.3.2 Setting Up Sales Assistance Prompts in AOL.....	2-10
2.3.3 Setting Up Service Items in AOL .....	2-10

2.3.4	Setting Up Oracle General Ledger .....	2-12
2.3.5	Setting Up Product Items in Oracle Inventory .....	2-12
2.3.6	Setting Up Serviceable Items in Oracle Inventory .....	2-14
2.3.7	Setting Up Service Items in Oracle Inventory .....	2-15
2.3.8	Setting Up Oracle Inventory for Regular Available to Promise (ATP) .....	2-16
2.3.9	Setting Up Oracle Receivables .....	2-17
2.3.10	Setting Up Oracle Order Capture .....	2-17
2.3.11	Setting Up Oracle Order Management .....	2-18
2.3.12	Setting Up Oracle Pricing .....	2-19
2.3.13	Setting Up Prices for Serviceable and Service Items in Oracle Pricing .....	2-21
2.3.14	Setting Up Oracle Human Resources .....	2-23
2.3.15	Setting Up Oracle Store Manager Users .....	2-25
2.4	Setting Up the Optional Dependencies .....	2-26
2.4.1	Setting Up Oracle Advanced Inbound .....	2-26
2.4.2	Setting Up Oracle Advanced Supply Chain Planning for Global ATP .....	2-27
2.4.3	Setting Up Oracle Configurator .....	2-27
2.4.4	Setting Up Oracle Contracts .....	2-29
2.4.5	Setting Up Oracle iMarketing .....	2-32
2.4.6	Setting Up Oracle iPayment .....	2-32
2.4.7	Setting Up Oracle Material Requirements Planning .....	2-32
2.4.8	Setting Up Oracle Shipping .....	2-33
2.4.9	Setting Up Oracle Workflow .....	2-33

### **3 Installation and Dependency Verification**

3.1	Checkpoints .....	3-1
-----	-------------------	-----

### **4 Oracle iStore 11i Setup**

4.1	Overview of Store Creation .....	4-2
4.2	Accessing the Merchant UI .....	4-3
4.3	Creating Specialty Stores .....	4-4
4.4	Creating the Hierarchy .....	4-8
4.5	Setting Up the Product Catalog .....	4-14
4.5.1	Using Seeded Relationship Types .....	4-18
4.6	Testing the Store .....	4-20

## 5 Oracle iStore 11i Customization

5.1	Overview of Store Customization.....	5-2
5.2	Customizing Multimedia .....	5-3
5.2.1	Creating Media Source Files.....	5-3
5.2.2	Cataloging Multimedia .....	5-3
5.3	Cataloging Multimedia Components.....	5-7
5.4	Customizing Templates.....	5-10
5.4.1	Creating Template Source Files .....	5-10
5.4.2	Cataloging Templates.....	5-14
5.5	Cataloging Display Styles .....	5-19
5.6	Modifying the Hierarchy.....	5-20
5.7	Creating Relationships.....	5-24
5.8	Customizing Product Presentation at the Category Level.....	5-27
5.9	Customizing Product Presentation at the Item Level .....	5-28
5.9.1	Modifying the Product Catalog .....	5-28
5.9.2	Creating Images for Products.....	5-33
5.9.3	Adding Item Descriptive Flexfields .....	5-34
5.10	Setting Up Product Searches.....	5-35
5.10.1	Setting Up Oracle Inventory for Product Search.....	5-36
5.10.2	Setting Search Profile Options.....	5-36
5.10.3	Populating the Oracle iStore 11i Search Table for Category Level Search .....	5-37
5.10.4	Populating the Oracle iStore 11i Search Table for Section Level Search.....	5-38
5.10.5	Changing Between Category Level Search and Section Level Search .....	5-39
5.10.6	Enabling a Fuzzy Search .....	5-40
5.10.7	Creating Search Index Tables .....	5-41
5.10.8	PL/SQL, Java, and JSP's Involved in Search.....	5-42
5.10.9	Customizing Search .....	5-43
5.10.10	Adding Stopwords to Searches.....	5-44
5.11	Customizing the Shopping Cart.....	5-45
5.11.1	Enabling Unit of Measure (UOM) Conversions .....	5-46
5.11.2	Allowing Decimal Quantities for Items .....	5-46
5.11.3	Specifying Flexfields At the Checkout Page .....	5-47
5.11.4	Setting Up Credit Card Payments in Oracle iStore 11i.....	5-48
5.12	Previewing Products and Sections.....	5-50

## 6 Oracle iStore 11i Administration

6.1	Overview of Store Administration.....	6-2
6.2	Roles and Permissions for Oracle iStore 11i Users .....	6-2
6.3	Setting Up Oracle iStore 11i Customer Types .....	6-7
6.3.1	Setting Default Customer Roles .....	6-8
6.3.2	Setting Default Customer Responsibilities.....	6-8
6.4	Setting Up B2B Users .....	6-9
6.4.1	Creating B2B User Roles.....	6-12
6.4.2	Approving B2B Users .....	6-15
6.4.3	Modifying B2B User Data .....	6-17
6.5	Managing the Cache.....	6-22
6.5.1	Purging the Entire Cache .....	6-23
6.5.2	Purging the Section Cache .....	6-23
6.5.3	Purging the Product Cache.....	6-24

## 7 Profile Options, Accounts, and Forms Settings

7.1	Before You Begin .....	7-2
7.2	Setting Up Store Manager User Accounts .....	7-2
7.3	Setting Up the Guest User Account .....	7-3
7.4	Setting Up JTF Properties .....	7-7
7.5	Setting Foundation (JTF) Profile Options.....	7-8
7.6	Setting Oracle iStore (IBE) Profile Options.....	7-9
7.7	Setting Profile Options for Language and Currency.....	7-20
7.8	Setting Order Capture (ASO) Profile Options.....	7-21
7.9	Setting Order Management (OM) Profile Options .....	7-23
7.10	Setting Multi Organization (MO) Profile Options.....	7-23
7.11	Setting Oracle Contracts Core (OKC) Profile Options .....	7-24
7.12	Setting Up an Oracle iMarketing User .....	7-25
7.13	Setting Up Concurrent Program Manager.....	7-25
7.13.1	iStore Search Insert.....	7-25
7.13.2	iStore Section Search Refresh.....	7-26
7.13.3	iStore - One Click Consolidation.....	7-26
7.14	Setting Up Order Management in Forms.....	7-28
7.14.1	Setting Up Web-Enabled Shipping Methods .....	7-29
7.15	Setting Up Site-Level Profile Options.....	7-29

7.16	Setting Up Order Capture in Forms .....	7-30
7.17	Understanding Cookies.....	7-31

## 8 Oracle iStore 11/ Catalog APIs

8.1	Catalog API Class Summary.....	8-2
8.2	Class Item .....	8-3
8.2.1	Variables for Class Item .....	8-4
8.2.2	Methods for Class Item .....	8-5
8.3	Class ItemFlexfield .....	8-38
8.3.1	Methods for Class ItemFlexfield .....	8-38
8.4	Class PriceObject .....	8-39
8.4.1	Methods for Class PriceObject .....	8-40
8.5	Class Section.....	8-41
8.5.1	Variables for Class Section.....	8-42
8.5.2	Methods for Class Section.....	8-42
8.6	Exception Classes for Package oracle.apps.ibe.catalog.....	8-58

## 9 Oracle iStore 11/ Shopping Cart Quote APIs

9.1	Shopping Cart Quote API Class Summary .....	9-3
9.2	Class CCTrxnOutRecord .....	9-4
9.2.1	Variables for Class CCTrxnOutRecord .....	9-4
9.2.2	Constructors for Class CCTrxnOutRecord .....	9-5
9.3	Class Contract .....	9-6
9.3.1	Variables for Class Contract .....	9-6
9.3.2	Constructors for Class Contract .....	9-7
9.3.3	Methods for Class Contract .....	9-7
9.4	Class ControlRecord .....	9-9
9.4.1	Variables for Class ControlRecord .....	9-10
9.4.2	Constructors for Class ControlRecord .....	9-11
9.5	Class FreightChargeRecord .....	9-11
9.5.1	Variables for Class FreightChargeRecord .....	9-11
9.5.2	Constructors for Class FreightChargeRecord .....	9-14
9.6	Class HeaderRecord.....	9-14
9.6.1	Variables for Class HeaderRecord.....	9-14
9.6.2	Constructors for Class HeaderRecord.....	9-22

9.7	Class LineAttributeExtRecord .....	9-23
9.7.1	Variables for Class LineAttributeExtRecord .....	9-23
9.7.2	Constructors for Class LineAttributeExtRecord .....	9-25
9.8	Class LineDetailRecord .....	9-25
9.8.1	Variables for Class LineDetailRecord .....	9-25
9.8.2	Constructors for Class LineDetailRecord .....	9-31
9.9	Class LineRecord .....	9-31
9.9.1	Variables for Class LineRecord .....	9-32
9.9.2	Constructors for Class LineRecord .....	9-37
9.10	Class LineRelationshipRecord .....	9-37
9.10.1	Variables for Class LineRelationshipRecord .....	9-37
9.10.2	Constructors for Class LineRelationshipRecord .....	9-39
9.11	Class OrderHeaderRecord .....	9-39
9.11.1	Variables for Class OrderHeaderRecord .....	9-39
9.11.2	Constructors for Class OrderHeaderRecord .....	9-40
9.12	Class PaymentRecord .....	9-40
9.12.1	Variables for Class PaymentRecord .....	9-40
9.12.2	Constructors for Class PaymentRecord .....	9-44
9.13	Class PriceAdjustmentAttributeRecord .....	9-44
9.13.1	Variables for Class PriceAdjustmentAttributeRecord .....	9-44
9.13.2	Constructors for Class PriceAdjustmentAttributeRecord .....	9-46
9.14	Class PriceAdjustmentRecord .....	9-46
9.14.1	Variables for Class PriceAdjustmentRecord .....	9-47
9.14.2	Constructors for Class PriceAdjustmentRecord .....	9-54
9.15	Class PriceAdjustmentRelationshipRecord .....	9-54
9.15.1	Variables for Class PriceAdjustmentRelationshipRecord .....	9-54
9.15.2	Constructors for Class PriceAdjustmentRelationshipRecord .....	9-56
9.16	Class PriceAttributeRecord .....	9-56
9.16.1	Variables for Class PriceAttributeRecord .....	9-56
9.16.2	Constructors for Class PriceAttributeRecord .....	9-68
9.17	Class Quote .....	9-68
9.17.1	Variables for Class Quote .....	9-69
9.17.2	Constructors for Class Quote .....	9-73
9.17.3	Methods for Class Quote .....	9-73
9.18	Class QuoteAccessRecord .....	9-88



9.18.1	Variables for Class QuoteAccessRecord .....	9-89
9.18.2	Constructors for Class QuoteAccessRecord .....	9-90
9.19	Class ShipmentRecord .....	9-90
9.19.1	Variables for Class ShipmentRecord .....	9-90
9.19.2	Constructors for Class ShipmentRecord .....	9-96
9.20	Class SubmitControlRecord .....	9-96
9.20.1	Variables for Class SubmitControlRecord .....	9-97
9.20.2	Constructors for Class SubmitControlRecord .....	9-97
9.21	Class TaxDetailRecord .....	9-97
9.21.1	Variables for Class TaxDetailRecord .....	9-97
9.21.2	Constructors for Class TaxDetailRecord .....	9-101
9.22	Exceptions for Package oracle.apps.ibe.shoppingcart.quote .....	9-101

## 10 Oracle iStore 11/ Postsales APIs

10.1	Postsales API Class Summary .....	10-2
10.2	Class AkCurrencyFormatter .....	10-2
10.2.1	Variables for Class AkCurrencyFormatter .....	10-3
10.2.2	Methods for Class AkCurrencyFormatter .....	10-3
10.3	Class AkDateFormatter .....	10-5
10.3.1	Variables for Class AkDateFormatter .....	10-5
10.3.2	Methods for Class AkDateFormatter .....	10-5
10.4	Class AkQuery .....	10-7
10.4.1	Variables for Class AkQuery .....	10-7
10.4.2	Constructors for Class AkQuery .....	10-7
10.4.3	Methods for Class AkQuery .....	10-7
10.5	Class AkQueryCondition .....	10-24
10.5.1	Variables for Class AkQueryCondition .....	10-24
10.5.2	Constructors for Class AkQueryCondition .....	10-24
10.5.3	Methods for Class AkQueryCondition .....	10-25
10.6	Class AkRegion .....	10-26
10.6.1	Variables for Class AkRegion .....	10-26
10.6.2	Methods for Class AkRegion .....	10-26
10.7	Class IbeAtpPvt .....	10-33
10.7.1	Variables for Class IbeAtpPvt .....	10-33
10.7.2	Constructors for Class IbeAtpPvt .....	10-33

10.7.3	Methods for Class IbeAtpPvt .....	10-33
10.8	Class Query .....	10-34
10.8.1	Variables for Class Query .....	10-35
10.8.2	Constructors for Class Query .....	10-35
10.8.3	Methods for Class Query .....	10-35
10.9	Class QueryCondition .....	10-49
10.9.1	Variables for Class QueryCondition.....	10-50
10.9.2	Constructors for Class QueryCondition .....	10-50
10.9.3	Methods for Class QueryCondition.....	10-50
10.10	Class QueryUtil.....	10-51
10.10.1	Variables for Class QueryUtil.....	10-51
10.10.2	Constructors for Class QueryUtil .....	10-51
10.10.3	Methods for Class QueryUtil.....	10-51
10.11	Class QueryValidatorException .....	10-53
10.11.1	Variables for Class QueryValidatorException .....	10-53
10.11.2	Constructors for Class QueryValidatorException.....	10-53
10.12	Examples of Customizations with the Postsales APIs .....	10-54
10.12.1	AkQuery.java .....	10-54
10.12.1.1	The connect Method .....	10-54
10.12.1.2	The addCondition Method .....	10-54
10.12.1.3	The addFormatter Method .....	10-55
10.12.1.4	The setOrderByColumn Method.....	10-56

## 11 Diagnostics and Troubleshooting

11.1	Java Applet Warning Workaround.....	11-2
11.2	Error ORA-29868 While Executing amvicn.sql .....	11-3
11.3	Display Manager Errors .....	11-4
11.3.1	Display Manager Error Messages .....	11-4
11.4	Catalog and Pricing Errors.....	11-5
11.4.1	Checkpoints.....	11-5
11.4.2	Specific Error Messages.....	11-6
11.5	Shopping List Errors .....	11-9
11.6	Search Errors .....	11-9
11.7	Postsales Errors .....	11-11
11.7.1	Order Summary Page Records Out of Sequence.....	11-11

11.8	Potential Issues Installing Oracle8 <i>i</i> interMedia Text Version 8.1.7.....	11-11
11.8.1	Manually Installing ctxsys Data Dictionary .....	11-11
11.8.2	Post-Installation Setup.....	11-12
11.9	Reporting Issues .....	11-14

## **A Oracle iStore 11*i* Roles and Responsibilities**

A.1	Oracle Forms Roles and Responsibilities.....	A-2
A.2	Oracle CRM Applications Roles and Responsibilities .....	A-5
A.3	Oracle iStore 11 <i>i</i> Customer UI Roles and Responsibilities.....	A-6

## **Index**



---

---

# Send Us Your Comments

## **Oracle iStore Implementation Guide, Release 11*i***

### **Part No. A90344-01**

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 in the following ways:

- Electronic mail: [eccontent\\_us@oracle.com](mailto:eccontent_us@oracle.com)
- FAX: (650) 654-6208 Attn: Oracle iStore Documentation
- Postal service:  
Oracle Corporation  
Oracle iStore Documentation  
500 Oracle Parkway, 60p4  
Redwood Shores, CA 94065  
USA

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

## Intended Audience

This document is intended for IT professionals who are tasked with implementing Oracle iStore 11i.

## Related Documents

Check the latest versions of the following documents for additional information on Oracle Applications, Release 11i:

- *Release Notes, Oracle Applications Release 11i*
- *Oracle Applications Release 11i Concepts*
- *Installing Oracle Applications Release 11i*
- *Oracle Applications System Administrator's Guide Release 11i*
- *Oracle Applications Implementation Wizard User's Guide*
- *Oracle Applications Product Update Notes, Release 11i*
- *Oracle iStore Concepts And Procedures, Release 11i*
- *Oracle General Ledger User's Guide*
- *Oracle Inventory User's Guide*
- *Oracle Marketing Online Concepts and Procedures, Release 11i*
- *Oracle Order Management User's Guide*
- *Oracle Pricing User's Guide*
- *Oracle Receivables User's Guide*

- *Oracle Master Scheduling/ Planning Guide*
- *Oracle Configurator and SellingPoint Administration Guide, Release 11i and 4.2.2*
- *Implementing Oracle CRM: ERP Functional Checklist, Release 11i* (available on Oracle MetaLink)
- *Implementing Oracle CRM: Foundation Functional Checklist, Release 11i* (available on Oracle MetaLink)

## Conventions

This manual uses the typographic conventions listed in the following table:

<b>Convention</b>	<b>Meaning</b>
<i>italic text</i>	Book titles
Courier text	User commands, file content examples, directory names
UPPERCASE	Structured Query Language (SQL) commands, initialization parameters, profile options, responsibilities, or environment variables
<b>boldface text</b>	Menu, button, keyboard, and form options, emphasis
< >	Angle brackets enclose user-supplied names. Note: Do not type the angle brackets.

## Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. 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 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 Java and SQL\*Plus code examples in this document. The conventions for writing Java and SQL\*Plus 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.



---

---

## Oracle iStore 11*i* Overview

This chapter provides an overview of the features and architecture of Oracle iStore 11*i*. Topics include:

- [Product Overview](#)
- [Architectural Overview](#)
- [Hardware Requirements](#)
- [Software Requirements](#)

## 1.1 Product Overview

Oracle iStore 11i provides businesses from all industries with the ability to establish business-to-business and business-to-consumer electronic commerce. The Oracle iStore 11i application provides an easy-to-use mechanism for merchants to set up Internet storefronts that capture and process customer orders and to integrate their storefronts with Oracle Enterprise Resource Planning (ERP) applications.

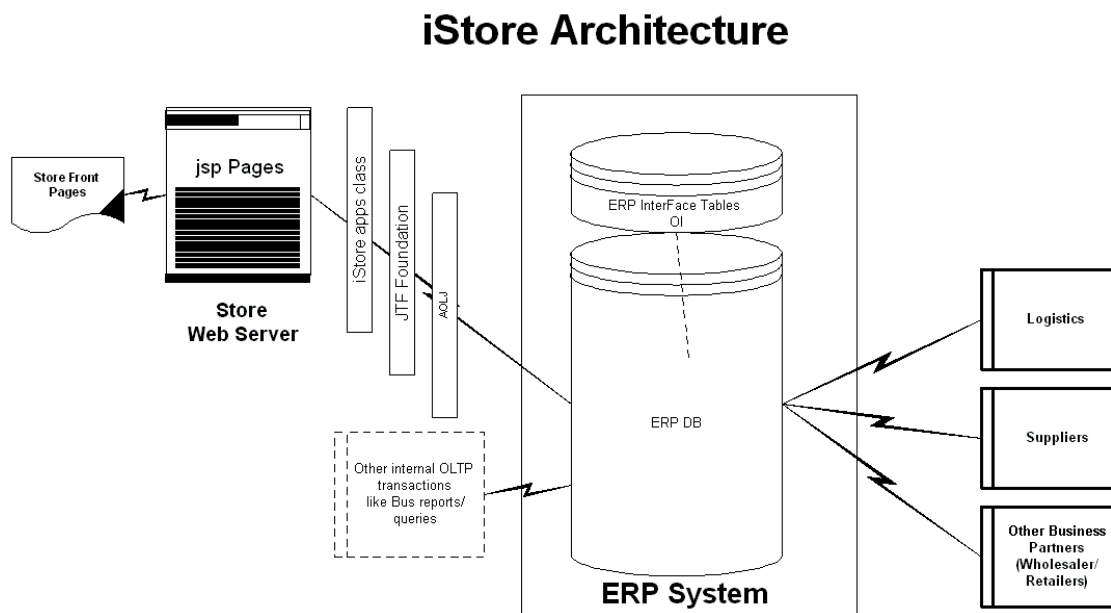
The key features and benefits of Oracle iStore 11i include the following:

- **Specialty stores** to create different stores within a single instance for serving different customer segments. Each store can have its own product selection, user interface, and process flows, while all the stores can utilize a unified, central merchant administration and repository of products and content.
- **A rich product catalog** that can display in multiple languages and currencies and is dynamically generated to reflect customer-specific product selection and pricing. The product catalog also supports a variety of relationships between products and product groups.
- **Oracle Configurator** for guided selling of complex products and configurable items
- **Sophisticated order capture** for customer-specific pricing, shipping and handling, and tax, and access to inventory availability
- **Channel integration** to leverage assets and processes, and to achieve a consistent customer experience across the contact points
- **Seeded roles and permissions** to offer personalized features for different customer segments and channel partners
- **Self-administration** that enables customers to manage store users and processes, thereby lowering operational costs
- **Personalization and recommendations** that cross-sell and up-sell items and improve the visits to purchase ratio for your Web stores
- **B2B functionality** that allows management of complex relationships with corporate customers in a self-service environment

## 1.2 Architectural Overview

In Oracle iStore 11i architecture, data is loaded and retrieved directly from ERP main tables. This functionality provides instant real time data status to Web stores and to other integrating systems and applications.

*Figure 1–1 Oracle iStore 11i Architecture*



## 1.3 Hardware Requirements

The suggested hardware configuration for Oracle iStore 11i is a series of Web servers in the front and a high performance database server machine in the back end. With global systems, the necessity for high performance database servers is even greater.

Oracle recommends the following server requirements:

- ERP database server machine - high throughput at fast speed (CPU)
- Web servers running Apache for external customers
- One forms server for administration

You can determine the actual sizing of the machines after completing capacity planning.

Specific hardware requirements depend on the particular installation that you perform. The hardware requirements listed in the following table are guidelines only, and assume a single-node environment.

**Table 1-1 Minimum Hardware Requirements**

<b>Hardware</b>	<b>Requirement</b>
CPU	2 CPUs minimum, 4 or more highly recommended
Memory	256 MB minimum, 1GB or more highly recommended
Disk Space	22GB, including 1GB in /tmp (plus an additional 9GB if installing from a staging area)

## 1.4 Software Requirements

The minimum software requirements are listed in the following table.

**Table 1–2 Minimum Software Requirements**

<b>Software</b>	<b>Requirement</b>
Database	8.1.7 version of Oracle8i
Middle-Tier	The middle-tier requirements are an Apache 3.0 version Web server and Oracle Forms 6.0 server (as a part of ERP 11i implementation). For faster page serving, you can also use a caching server in front of the Web server series. The caching server (for example, Calypso) then serves the static content through the cached TCP/IP packets (or cached pages). An invalidated cache results in service requests being directed to the Web servers.





---

---

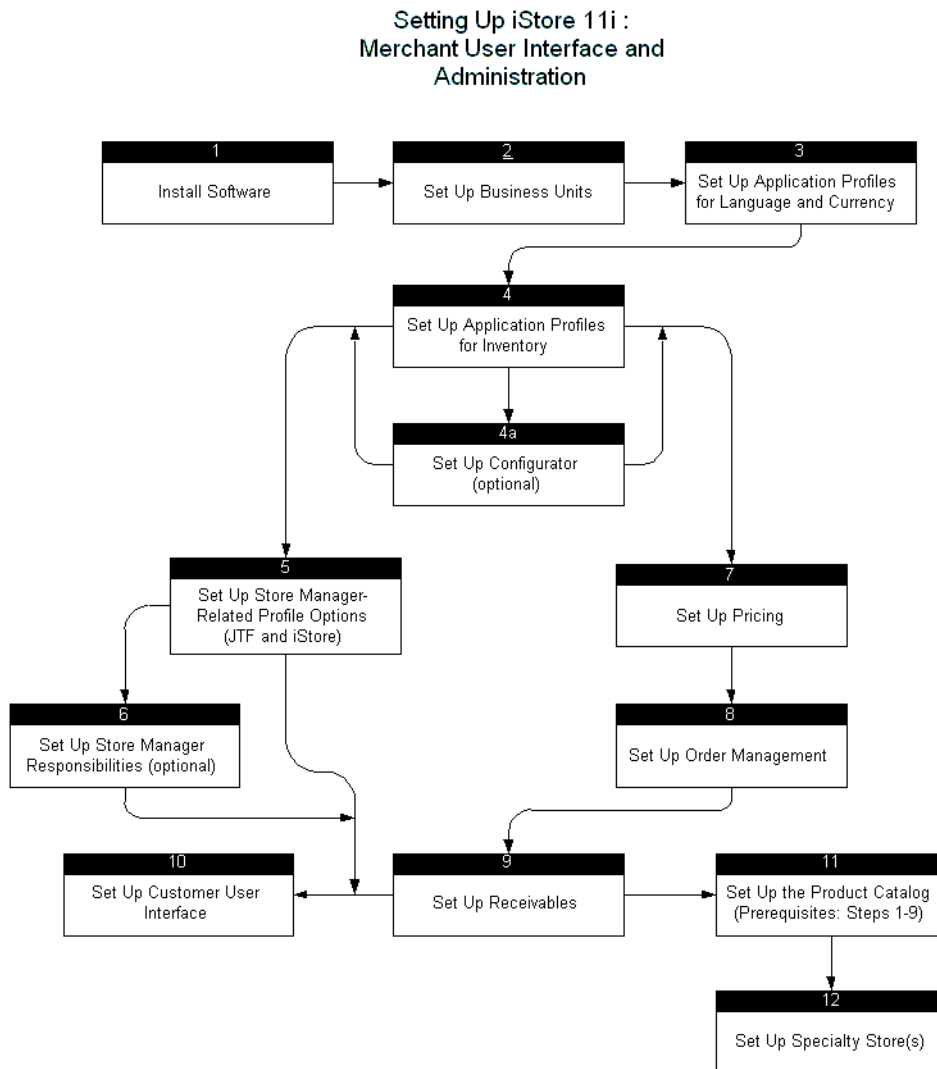
# Oracle iStore 11i Dependency Setups

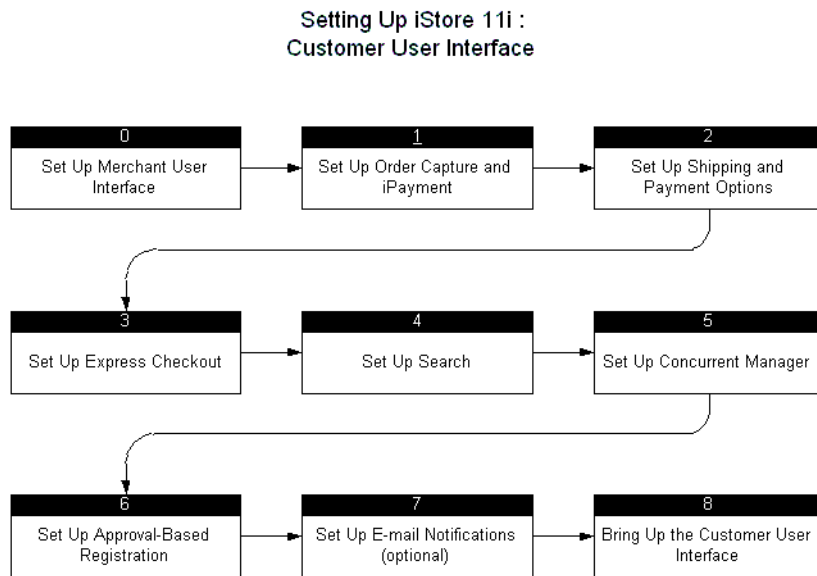
This chapter describes the setup of applications and other program functions on which Oracle iStore 11*i* depends. Topics include:

- [Setup Process Flow Diagrams](#)
- [Dependency Requirements](#)
- [Setting Up the Mandatory Dependencies](#)
- [Setting Up the Optional Dependencies](#)

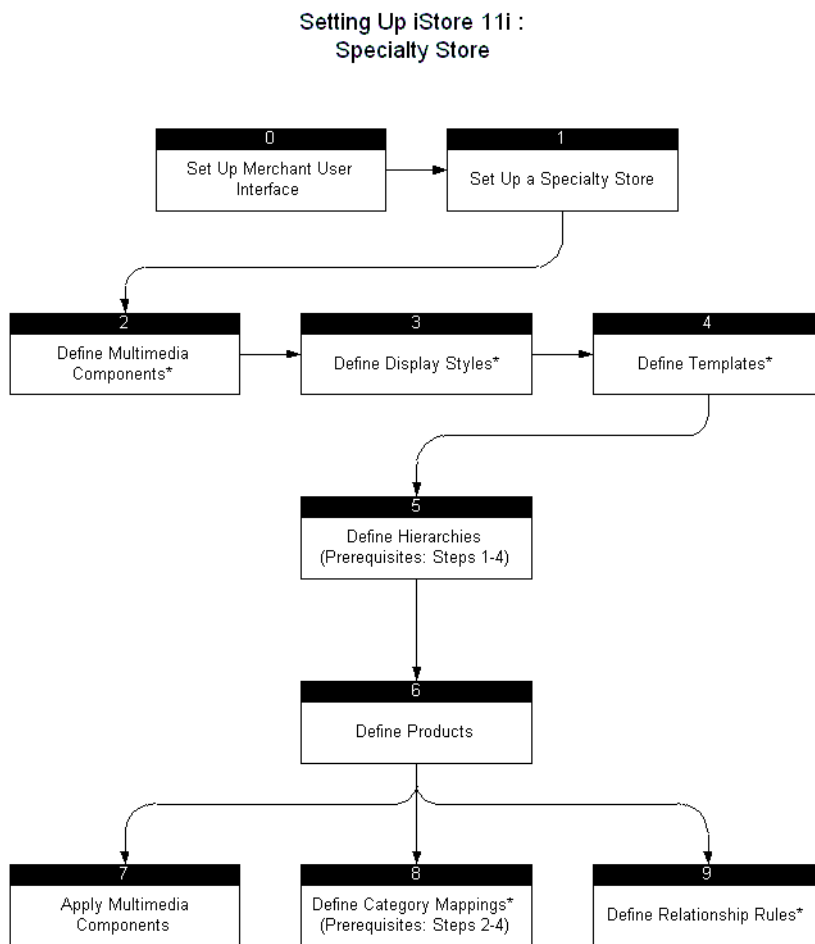
## 2.1 Setup Process Flow Diagrams

Figure 2-1 Setup Process Flow (Merchant UI)



**Figure 2–2 Setup Process Flow (Customer UI)**

**Figure 2-3 Setup Process Flow (Specialty Store)**



\* = Optional step. Seeded information supplied.

## 2.2 Dependency Requirements

Oracle iStore 11*i* integrates with many other Oracle application modules to provide and extend its functionality. You must set up the mandatory modules before Oracle iStore 11*i* can run. Setting up the optional modules is not required; however, if they are not set up, then the additional functionality provided by these modules will not be available.

### 2.2.1 Mandatory Modules

Modules listed in this section must be set up for Oracle iStore 11*i* to function properly.

#### **Oracle Application Object Library**

Oracle Application Object Library (AOL) is a required dependency of all Oracle applications modules. Oracle iStore 11*i* uses the AOL to manage responsibilities of store managers as well as customers. You also define new languages and manage prompts for your store here.

#### **Oracle General Ledger**

Oracle General Ledger provides business unit information to Oracle iStore 11*i*.

#### **Oracle Human Resources**

Oracle Human Resources stores information related to your organizations, such as permitted bill-to and ship-to countries.

#### **Oracle Inventory**

Oracle Inventory is a required dependency for Oracle iStore 11*i*. The following information is set up in Oracle Inventory:

- Category structure to set intelligent defaults for the ways products can be displayed, and default values for different multimedia components used to display products
- Item information
- Oracle Inventory itself requires at least one inventory organization to be set up and at least one business unit (organization) to be set up. In addition, it requires at least one product catalog group to be set up, even though you may not use it.

### **Oracle Order Capture**

Oracle Order Capture provides pricing, shipping, and tax information for Oracle iStore 11*i*, and transforms Oracle iStore 11*i* shopping carts into orders in Oracle Order Management.

### **Oracle Order Management**

Oracle Order Management processes orders from Oracle iStore 11*i* and provides order tracking, returns, and history. Order Management in turn depends upon Oracle Pricing and Oracle General Ledger. Oracle iStore 11*i* uses Order Management (OM) to keep records of orders placed by customers and pricing of those orders. It does so by using APIs provided by Order Capture (OC). Order Management in turn uses:

- Oracle Receivables for keeping records of customers and invoices, and capturing payments upon shipments
- Oracle Order Capture
- Oracle Pricing for determining prices of goods sold
- Oracle Shipping for shipping execution

---

---

**Note:** Order Management in turn requires you to set up the inventory and business units as well as financials-related parameters.

---

---

### **Oracle Pricing**

Oracle Pricing provides the prices of goods sold for Oracle iStore 11*i*. It enables complex, customer-specific pricing through price lists and pricing agreements.

### **Oracle Receivables**

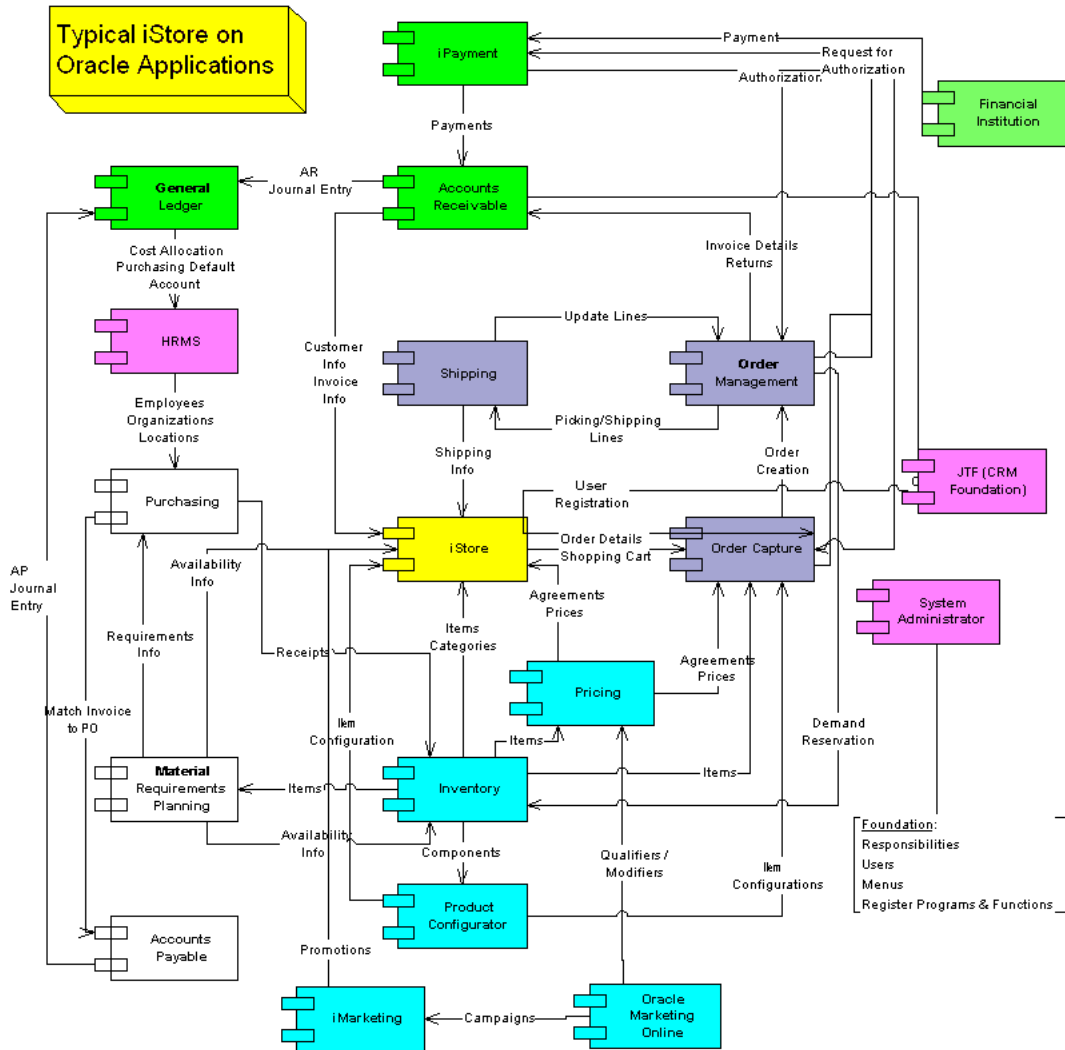
Oracle Receivables is a central data repository for customer information that uses the Trading Community Architecture (TCA) model. It also calculates taxes and generates invoices

## 2.2.2 Optional Modules

The following Oracle applications modules can be set up to provide additional functionality for your electronic store:

- **Oracle Advanced Inbound** to process call-me-back requests
- **Oracle Advanced Supply Chain (Global ATP Server)** to provide product availability information
- **Oracle Configurator** to enable customer configured products and provide guided selling as well as to perform some of the validations of the shopping cart
- **Oracle Contracts** for complete contract management and service agreements
- **Oracle CRM Business Intelligence** to assess the performance of the store
- **Oracle iMarketing** to define promotions and discounts
- **Oracle Incentive Compensation** for managing sales compensation across all channels
- **Oracle iPayment** to process payments with online credit card authorization
- **Oracle iSupport** to provide return authorizations and knowledge base integration
- **Oracle Marketing Online** to define, execute, and manage marketing campaigns, budgets, and segments across all channels
- **Oracle Material Requirements Planning** to provide product availability information
- **Oracle Shipping** to calculate shipping charges
- **Oracle Workflow** to send e-mail notifications and confirmations to customers

Figure 2-4 Typical Oracle iStore 11i Integration With Other Oracle Applications





## 2.3 Setting Up the Mandatory Dependencies

The following steps indicate the suggested task sequence for setting up Oracle iStore 11i's mandatory dependencies.

### Steps

1. Define and enable languages in AOL.
2. Define and set up your Business Units, Set of Books, in Oracle General Ledger.
3. Set up Oracle Inventory.
4. Set up Oracle Receivables.
5. Set up Oracle Order Capture.
6. Set up Oracle Order Management.
7. Set up Oracle Human Resources.
8. Optional: Define more users for store manager.
9. Set up JTF, ASO, and iStore profiles.
10. Create the product catalog.

### Using the Oracle Application Implementation Wizard

Use the Oracle Application Implementation Wizard (AIW) to coordinate dependency setups and identify the steps required to implement the Oracle iStore 11i application.

You can use the AIW to see the graphical overview of the steps involved, read online help on set up and open the appropriate forms. You can also document your actions for further reference and review.

Please refer to *Oracle Application Implementation Wizard User's Guide* for more details.

### 2.3.1 Setting Up Languages in AOL

Oracle Applications 11i enables you to have a multi-lingual set up against one instance. However the set of languages available is dependent upon the character set of the database implementation. The languages enabled determine the set of languages in which the store could be presented to the user.

For a given specialty store, you select languages from those languages enabled at the Oracle Application level. These will be the languages that are going to be

available in that specialty store. See [Chapter 4](#) for details of multilingual capabilities at the store level.

## 2.3.2 Setting Up Sales Assistance Prompts in AOL

Web store customers can request help from a sales representative through the Oracle iStore 11i Sales Assistance feature. When they do so, the application asks them to choose from a list of reasons why they need assistance, and to enter any comments that they may have. When the customers indicate why they need assistance, the application submits their shopping carts as orders with an Entered status, and the sales representative specified in the profile option IBE: Default Sales Assistant to Send Workflow Notification receives an e-mail notification. The sales representative can then contact the customer, provide the necessary assistance, and book the order.

You must use AOL to set up the list of reasons for needing assistance that customers choose from when using the Sales Assistance feature. You can do this by setting up lookup codes, with meanings, for the FND lookup type "IBE\_SALES\_ASSIST\_REASONS\_LK," using the AOL Lookups Form. See *Oracle Applications System Administrator's Guide, Release 11i* for more information.

After setting up the list of reasons, you must set the profile option IBE: Default Sales Assistant to Send Workflow Notification to activate the Sales Assistance feature. See [Chapter 7](#) for more information.

## 2.3.3 Setting Up Service Items in AOL

Selling service items in your Oracle iStore 11i Web stores requires preliminary setup in AOL. Use the following procedure to create lookups for Oracle iStore 11i and set up the Oracle iStore 11i shopping cart pull-down menu for technical support.

### Prerequisites

Service items have been set up in Oracle Inventory. See [Section 2.3.7, "Setting Up Service Items in Oracle Inventory"](#) for more information.

Prices for service items have been set up in Oracle Pricing. See [Section 2.3.13, "Setting Up Prices for Serviceable and Service Items in Oracle Pricing"](#) for more information.

## Steps

1. In SQL\*Plus, query the `mtl_system_items_b` table to find the `inventory_item_id` for the service item, as shown below:

```
$ sqlplus app/<apps_pwd>@<connect_string>
SQL> select inventory_item_id from mtl_system_items_b where organization_id
= <enter the organization ID value> and segment1 = 'support_item';
```

SQL\*Plus returns the `inventory_item_id` value.

2. Launch Oracle Forms by navigating to:

```
http://<host>:<apache port>/
```

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

3. Log in with the Application Developer responsibility.
4. Choose **Application > Lookups > Application Object Library**.

The Application Object Library Lookups form opens.

5. Query by example with `Type = IBE_SUPPORT_FOR_SHOP_CART`.

The Application Object Library Lookups form opens with the record for lookup type `IBE_SUPPORT_FOR_SHOP_CART`.

6. Add a line with the following values:
  - a. Code = Enter the `inventory_item_id` value returned by the SQL\*Plus query.
  - b. Meaning = Support
  - c. Description = Support
  - d. From = Enter the current date.
  - e. Enabled = Check this checkbox.
7. Add a line with the following values:
  - a. Code = NONE
  - b. Meaning = None
  - c. Description = No Service is Selected
  - d. From = Enter the current date.
  - e. Enabled = Check this checkbox.

8. Click the Save icon in the toolbar to save your changes.
9. Next, set the IBE: Use Support and IBE: Use Support Cart Level profile options to **Yes**. See [Chapter 7](#) for more information.

### 2.3.4 Setting Up Oracle General Ledger

Because Oracle Order Management and Oracle Inventory require at least one Multi Org and associated set of books, you need to create at least one business unit in Oracle General Ledger. See *Oracle Applications Release 11i Concepts* for more information on business units and multi-org. See *Oracle General Ledger User's Guide* for steps involved in setting up business units.

### 2.3.5 Setting Up Product Items in Oracle Inventory

Oracle Inventory serves as the repository of product items that can be sold through Oracle iStore 11i. Use the Oracle Inventory forms to create new items and then create additional content for the Web through the Store Manager. Before you can create items in the Oracle Inventory system, you must set up and define the structure around it. Refer to *Oracle Inventory User's Guide* for details of inventory set up.

#### Guidelines

While making decisions about set up, please keep the following guidelines in mind:

- Products must have their Web Status set to either Published or Unpublished to appear in the Oracle iStore 11i Merchant UI. The Web Status can be changed from Oracle Inventory or Oracle iStore 11i.
- Items that will be sold through the Web must be set with the flags Web Status = Published, and Orderable on Web, in the Web Option tab of the Inventory Master Item form. They must also be set with the flag Customer Orders Enabled, in the Order Management tab of the Inventory Master Item form.
- Oracle iStore 11i uses the category structure to keep track of default ways of displaying products as well as default values for multimedia components associated with the product. All items belonging to a category in specified category sets are treated similarly from display perspective. In Oracle iStore 11i setup, you set the IBE: Category Set profile option to a category set value that helps Oracle iStore 11i differentiate between items based on the categories they belong to within this category set. Items in the same category in this category set are treated as homogenous from display and multimedia default perspectives.

If you are planning to sell items of different types (e.g., books vs. computers) and need to display them differently (i.e., use different templates) then you should create two different categories within this category set. If you do not have the flexibility to do so, then you may specify the display properties at the item level. Refer to [Chapter 5](#) for more details.

- Set up flexfields according to your needs. Flexfields are used to capture additional information about items.
- Oracle iStore 11i requires one Inventory Organization to be identified against which the product catalog is built. Typically this would be the Master Inventory Organization.
- In a multiple operating unit environment, the main Inventory Organization should consist of all the items from all the operating units. If you need to separate the items to be sold from each operating unit into different Inventory Organizations, create a separate Inventory Organization for each operating unit. These operating unit Inventory Organizations should exist only as subsets of the main Inventory Organization. Separate items from different operating units in the Web stores by setting up each operating unit with its own Web store sub-hierarchy within the main Oracle iStore 11i hierarchy. See [Section 4.4, "Creating the Hierarchy"](#) for more information about setting up the Oracle iStore 11i hierarchy.

## Steps

The `MTL_SYSTEM_ITEMS` table in Oracle Inventory is where product information resides. The following minimum setups are required:

1. Set the following parameter values:
  - Set the value for the `WEB_STATUS` flag parameter.
  - Set the value for the `CUSTOMER_ORDER_ENABLE_FLAG` parameter.
  - Set the value for the `INV_ORDERABLE_ON_WEB` parameter.These settings are required to display products in the Merchant UI.
2. In Inventory, map inventory items to organizational IDs (organizational IDs are set in General Ledger). This mapping determines the inventory items that specific organizational IDs can view.
3. Enter product descriptions.

## 2.3.6 Setting Up Serviceable Items in Oracle Inventory

Follow this procedure to create serviceable items in Oracle Inventory that you can sell through Oracle iStore 11i.

### Steps

1. Launch Oracle Forms by navigating to:  
`http://<host>:<apache port>/`  
and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in with the Inventory responsibility.
3. Choose **Inventory > Items > Master Items**.  
The Organizations window opens.
4. Search for and highlight the master inventory organization, and click **OK**.  
The Master Item window opens.
5. In the Item field, enter the item number.
6. In the Description field, enter the item description.
7. Click the flexfield icon [ ] next to the Description field, and fill in the flexfields in the Items pop-up window that appears. When you are finished, return to the Master Item window.
8. In the Main tab, set Primary Unit of Measure to **Each** and Item Status to **Active**.
9. In the Inventory tab, check the Reservable checkbox.
10. In the Web Option tab, set Web Status to **Published** and check the Orderable On the Web checkbox.
11. In the Order Management tab, check the Customer Ordered, Customer Orders Enabled, and Shippable checkboxes.
12. Repeat the above steps for any subset inventory organizations in a multiple operating unit environment that should contain this serviceable item.
13. Next, create prices for this item in Oracle Pricing. See [Section 2.3.13, "Setting Up Prices for Serviceable and Service Items in Oracle Pricing"](#) for more information.

## 2.3.7 Setting Up Service Items in Oracle Inventory

Follow this procedure to create service items in Oracle Inventory that you can sell through Oracle iStore 11i.

### Steps

1. Launch Oracle Forms by navigating to:  
`http://<host>:<apache port>/`  
  
and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Verify the UOM Mappings for Period value (MTH) to be used when setting up a service item in Oracle Inventory, as follows:
  - a. Log in with the Corporate Contracts Manager responsibility.
  - b. Choose **Setup > Contract > Units of Measure > UOM > UOM**.  
The Organizations window opens.
  - c. Search for and highlight the master inventory organization, and click **OK**.  
The Units of Measure window opens.
  - d. Verify and/or create a record with the following values:
    - Name = Month
    - UOM = MTH
    - Description = Month
    - Class = **Time**
  - e. In the Navigator - Corporate Contracts Manager window, choose **Setup > Contract > Units of Measure > Time Units of Measure**.  
The Map Time Units window opens.
  - f. Verify and/or create a record with the following values:
    - User Unit = **Month**
    - Base Unit = **Month**
    - Conversion = 1
3. Switch to the Inventory responsibility.

4. Choose **Inventory > Items > Master Items**.  
The Organizations window opens.
5. Search for and highlight the master inventory organization, and click **OK**.  
The Master Item window opens.
6. In the Item field, enter the item number.
7. In the Description field, enter the item description.
8. Click the flexfield icon [ ] next to the Description field, and fill in the flexfields in the Items pop-up window that appears. When you are finished, return to the Master Item window.
9. In the Main tab, set Primary Unit of Measure to **Month** and Item Status to **Active**.
10. In the Inventory tab, check the Reservable checkbox.
11. In the Service tab, check the Support Service checkbox.
12. In the Service Duration region of the Service tab, set Value to 12 and Period to **MTH** (Month).
13. In the Web Option tab, set Web Status to **Published** and check the Orderable On the Web checkbox.
14. In the Order Management tab, check the Customer Ordered, Customer Orders Enabled, and OE Transactable checkboxes.
15. Repeat the above steps for any subset inventory organizations in a multiple operating unit environment that should contain this service item.
16. Next, create prices for this item in Oracle Pricing. See [Section 2.3.13, "Setting Up Prices for Serviceable and Service Items in Oracle Pricing"](#) for more information.

### 2.3.8 Setting Up Oracle Inventory for Regular Available to Promise (ATP)

Oracle iStore 11i can provide regular available to promise (ATP) information on inventory items without customization. Oracle iStore 11i checks the ON\_HAND\_QTY field in the Oracle Inventory ATP columns to determine the availability of items requested in the Web stores. You must set up ATP in Oracle Manufacturing, define ATP sourcing rules in Oracle Inventory, and enable product items for ATP by setting their ATP and ATP component flags. See *Oracle Inventory User's Guide* for instructions on setting up ATP rules in Oracle Inventory for regular ATP.



If you want to enable global ATP for Oracle iStore 11i, you must install Oracle Advanced Supply Chain Planning and Oracle Material Requirements Planning. These modules will create global ATP rules that populate the ATP columns in Oracle Inventory. See [Section 2.4.2, "Setting Up Oracle Advanced Supply Chain Planning for Global ATP"](#) and [Section 2.4.7, "Setting Up Oracle Material Requirements Planning"](#) for more information on enabling global ATP.

### 2.3.9 Setting Up Oracle Receivables

Oracle iStore 11i uses the Oracle Receivables module to record customer information and tell customers about their invoices and payments made. See *Oracle Receivables User's Guide* for information on setting up Oracle Receivables.

Customer registration information is maintained in the TCA/Oracle Receivables schema. At a minimum, you need to set up the following:

- Address Validation
- Tax Codes and the Default Tax Code

### 2.3.10 Setting Up Oracle Order Capture

Oracle iStore 11i uses Order Capture to integrate with Oracle Order Management and Oracle Receivables. Oracle iStore 11i takes carts saved and orders placed in the Web stores and sends them to Order Capture. Order Capture saves the carts as quotes and passes the orders to Order Management.

When an order is placed in Oracle iStore 11i, Order Capture passes the order to Order Management with a status of either Entered or Booked, depending on the type of order. Oracle iStore 11i controls the order status for all orders placed from the Web stores. Orders with the status Entered can be modified in Order Management. Orders with the status Booked cannot be modified in Order Management.

When the profile option IBE: Use CABO UI is set to **Yes**, Oracle iStore 11i purchase orders are always passed by Order Capture to Order Management with an Entered status, and Oracle iStore 11i credit card and invoice orders are always passed by Order Capture to Order Management with a Booked status. Orders are sent to Order Management with these statuses regardless of the Order Capture profile settings, because Oracle iStore 11i sets the book\_order\_flag based on these rules.

---

---

**Note:** Credit card orders can be passed by Order Capture to Order Management with an Entered status if the users have chosen to fax their credit card information in the Web store billing page. The option to fax credit card information is only available on the billing page when the profile option IBE: Use CABO UI is set to **Yes**.

---

---

See [Chapter 7](#) for more information about the profile option IBE: Use CABO UI.

Order Capture saves the Oracle iStore 11*i* carts as quotes. The Order Capture quote name becomes the same as the Oracle iStore 11*i* cart name once the Oracle iStore 11*i* cart is saved. If you place the quote as an order in Order Capture, instead of from Oracle iStore 11*i*, Order Capture passes the order to Order Management with its status set according to the Order Capture profile options.

---

---

**Note:** Quotes and orders created in Order Capture can also be viewed in Oracle iStore 11*i*.

---

---

See *Oracle Order Capture Implementation Guide* for information on how to set up Order Capture. See *Oracle Order Capture Concepts and Procedures* for information on using Order Capture and viewing Oracle iStore 11*i* carts and orders in Order Capture.

### 2.3.11 Setting Up Oracle Order Management

Oracle iStore 11*i* uses Oracle Order Management to record customer orders, set up payment options and shipping options, and provide order status and shipping information to customers. Please see *Oracle Order Management User's Guide* for details of setting up Oracle Order Management.

Once you have set up Oracle Receivables and Oracle Order Management you only need to set up certain profile values. Those profile values are described in [Chapter 7](#).

## Restricting Items Based on Operating Units

In a multiple operating unit environment, you need to set up the associations between the operating units and the Inventory Organizations in Order Management. Each of these Inventory Organizations will be a subset of the main Inventory Organization.

Oracle iStore 11*i* uses these associations to limit the items that customers can access in the operating units' Web stores. You create a separate IBE customer responsibility for each operating unit in Oracle Forms. See [Section 7.10, "Setting Multi Organization \(MO\) Profile Options"](#) for more information about associating responsibilities with operating units.

Each customer name is assigned one of these responsibilities when the customer name is approved. When a customer enters a Web store, Oracle iStore 11*i* notes the customer's responsibility and the operating unit to which it is assigned, then uses the Inventory Organization associated with each operating unit to restrict customers to the items in the Inventory Organization. See [Section 4.3, "Creating Specialty Stores"](#) for instructions on choosing the responsibilities that are supported by a specialty store.

Specify the Inventory Organization (MASTER\_ORGANIZATION\_ID) associated with each operating unit (ORG\_ID) in the OE\_SYSTEM\_PARAMETERS\_ALL table. Oracle iStore 11*i* uses the OE\_SYSTEM\_PARAMETERS\_ALL table to be consistent with Order Management and Order Capture.

## 2.3.12 Setting Up Oracle Pricing

Setting up Oracle Pricing is one of the required steps for Order Management but is described here separately. For each item that you plan to sell, you must specify the price in at least one price list and make that price list available to customers. For walk-in users, Oracle iStore 11*i* picks a default price list to show the price of items on the product catalog pages in the store, and to price the shopping cart. The default price list to be used for different currencies is specified in the Oracle iStore 11*i* Merchant UI (see [Chapter 4](#) for more details).

Using Pricing, you can also set up advanced promotions and discounts. Oracle iStore 11*i* uses the Pricing engine to determine the best price that the customer can get based on the items in the shopping cart and the customer. You set up your pricing rules in the Pricing module. Oracle iStore 11*i* supports pricing attributes and customer-requested qualifiers (promocodes) that are set up in Oracle Pricing.

You must also create pricing agreements in Oracle Order Management or Oracle Pricing to enable pricing of quotes created by users in your organization with the

quote creator role. Pricing agreements set up the billing specifications that allow a quote to be priced. The attributes of a pricing agreement are price list, purchase order number, bill-to address, bill-to contact, invoicing terms, payment terms, and shipment terms. These attributes set up and apply default pricing rules in Oracle Order Management. Each pricing agreement can have only one price list assigned to it, but one list can be linked to multiple agreements. You can create universal and customer-specific pricing agreements.

At a minimum, you must set up Qualifiers and Modifiers in Oracle Pricing to support Oracle iStore 11i.

### Steps

1. Set Qualifiers in Oracle Pricing. Qualifiers are used to calculate item prices.
2. Set Modifiers in Oracle Pricing. Modifiers determine discounts or can be used to calculate shipping costs.

When setting up Oracle Pricing sourcing rules, the sourcing rules for Oracle Order Capture and Oracle Order Management must be identical. The qualifiers for these sourcing rules must also be identical. See *Oracle Pricing User's Guide* for more information.

### Pricing Setup Example

This procedure shows how to set up simple fixed-amount freight charges in Oracle iStore 11i:

1. Launch Oracle Forms, log in as SYSADMIN, and choose the Oracle Pricing Manager responsibility.
2. Choose **Modifiers > Define Modifier**. (If you don't have the Oracle Pricing Manager responsibility, grant it to the SYSADMIN user first.)
3. Click LOV for Type.
4. Click **Freight and Special Charge List**.
5. Enter some identifier for the modifier in the Number field (e.g., IBEFR01).
6. Enter some description in the Name field (e.g., IBE Freight Modifier).
7. Click in **Modifiers Summary > Modifier No**, and enter a number (e.g., 1).
8. Click in **Level**, select modifier level (e.g., Line).
9. Click in **Modifier Type**, select **Freight/Special Charge**.
10. Enter a Start Date and an optional End Date.

11. Scroll right. Click in **Pricing Phases**, choose **Line Charges** from the list of values (LOV).
12. Click the Discounts/Charges tab.
13. Click in **Charge Name**, select **Freight Costs** from the LOV.
14. Click **Application Method**, select **Amount**. Enter a per-line freight cost dollar amount (e.g., 3.00).
15. Save the form.

### 2.3.13 Setting Up Prices for Serviceable and Service Items in Oracle Pricing

Use this procedure to set up prices for serviceable and service items in Oracle Pricing.

#### Prerequisites

The serviceable and service items have been created in Oracle Inventory.

#### Steps

1. Launch Oracle Forms by navigating to:  

```
http://<host>:<apache port>/
```

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in with the Oracle Pricing Manager responsibility.
3. Choose **Price Lists > Price List Setup**.  

The Price Lists form opens.
4. Query by example for Name = the price list used in the specialty store that offers the item.  

The price list appears in the Price Lists form.
5. Place your cursor in the Product Context column of the List Lines tab and click the New icon in the toolbar to create a new line for the item record.
6. For a serviceable item, create the item record with the following values in the specified columns:
  - a. Product Context = **Item**

- b. Product Attribute = **Item Number**
  - c. Product Value = Enter the item number from Oracle Inventory, which appears in the Item field of the Master Items form for the item.
  - d. UOM = **Ea**
  - e. Line Type = **Price List Line**
  - f. Application Method = **Unit Price**
  - g. Value = Enter your desired value.
  - h. Start Date = Enter the current date.
  - i. Precedence = Enter the item precedence.
  - j. [ ] = When you click in this field, the Additional Info for List Lines flexfield window opens. In the Web field, enter *Y* to show that the item is Web-enabled. Click **OK** to return to the Price Lists form.
7. For a service item, create the item record with the following values in the specified columns:
- a. Product Context = **Item**
  - b. Product Attribute = **Item Number**
  - c. Product Value = Enter the item number from Oracle Inventory, which appears in the Item field of the Master Items form for the item.
  - d. UOM = **MTH**
  - e. Line Type = **Price List Line**
  - f. Application Method = **Percent Price** or **Unit Price**
  - g. Value = Enter your desired percentage or value.
  - h. Start Date = Enter the current date.
  - i. Precedence = Enter the item precedence.
  - j. [ ] = When you click in this field, the Additional Info for List Lines flexfield window opens. In the Web field, enter *Y* to show that the item is Web-enabled. Click **OK** to return to the Price Lists form.
8. Click the Save icon in the toolbar to save the record.
9. Next, for a service item, follow the instructions in [Section 2.3.3, "Setting Up Service Items in AOL"](#) to create lookups for Oracle iStore 11i and set up the Oracle iStore 11i shopping cart pull-down menu for technical support.

For a serviceable item, add the item to the specialty stores that should carry it, as outlined in [Chapter 4](#) and [Chapter 5](#).

### 2.3.14 Setting Up Oracle Human Resources

Installation and setup of Oracle Human Resources is a prerequisite to the Oracle iStore 11i globalization functionality. Oracle iStore 11i supports the implementation of a global store in a single instance. Each specialty store can support multiple languages and currencies. Globalization is also supported in an environment with multiple business units or organizations.

Every registered Web store customer has a preferred language and currency associated in his or her user profile. The Web store customer can change the preferred language or currency by modifying his or her account profile.

Globalization depends on setups for your organizations in Oracle Human Resources. Use Oracle Human Resources to specify the permitted bill-to and ship-to countries for each of your operating units. You can associate each of these operating units to different customer user responsibilities, as [Section 7.10, "Setting Multi Organization \(MO\) Profile Options"](#) explains.

In Oracle iStore 11i, you can then set up each of your specialty stores to support specific responsibilities. You can also set up each of your specialty stores to include or exclude B2B users by their own organization affiliations, for a higher degree of control over the extent of globalization available to each customer.

The Oracle Human Resources bill-to and ship-to country setups for each of the operating units associated with these responsibilities will determine the extent of globalization of each specialty store for each user.

Use the following procedure to set up bill-to and ship-to country information for an organization. See *Using Oracle HRMS - The Fundamentals* for more information.

#### Steps

1. Launch Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

2. Log in with the Human Resources responsibility.
3. Choose **Work Structures > Organization > Description**.

The Find Organization window opens.

4. In the Find Organization window, enter the search criteria for your merchant organization and click **Find**.

The Organization form opens with the organization's record.

5. In the Organization Classifications region, select **Operating Unit** in the Name fields.

6. Click **Others**.

The Additional Organization Information window opens.

7. In the Additional Organization Information window, enter %Country in the Find field and click **Find**.

8. In the search results, choose **Bill to Country** and click **OK**.

The Additional Organization Information window for Bill to Country opens.

9. Place your cursor in a Bill to Country field.

The Bill to Country window opens.

10. Add countries to the Additional Organization Information window for Bill to Country as follows:

- a. In the Bill to Country window, click the Bill to Country LOV button.

The Bill to Country search window opens.

- b. Search for a country, highlight it, and click **OK**.

The Bill to Country field in the Bill to Country window is populated with the value of this country.

- c. Click **OK** to add the country to the Additional Organization Information window for Bill to Country.

- d. Repeat for each country that you want to allow as a bill-to country for this merchant organization.

11. When you are finished adding countries, click **OK** in the Additional Organization Information window for Bill to Country.

You return to the Organization form with the organization's record.

12. In the Organization Classifications region, select **Operating Unit** in the Name fields.

13. Click **Others**.



The Additional Organization Information window opens.

14. In the Additional Organization Information window, enter %Country in the Find field and click **Find**.

15. In the search results, choose **Ship to Country** and click **OK**.

The Additional Organization Information window for Ship to Country opens.

16. Place your cursor in a Ship to Country field.

The Ship to Country window opens.

17. Add countries to the Additional Organization Information window for Ship to Country as follows:

- a. In the Ship to Country window, click the Ship to Country LOV button.

The Ship to Country search window opens.

- b. Search for a country, highlight it, and click **OK**.

The Ship to Country field in the Ship to Country window is populated with the value of this country.

- c. Click **OK** to add the country to the Additional Organization Information window for Ship to Country.

- d. Repeat for each country that you want to allow as a ship-to country for this merchant organization.

18. When you are finished adding countries, click **OK** in the Additional Organization Information window for Ship to Country.

You return to the Organization form with the organization's record.

### 2.3.15 Setting Up Oracle Store Manager Users

Oracle iStore 11i is seeded with the responsibility IBE\_ADMINISTRATOR. Assign this responsibility as the default responsibility for Oracle iStore 11i store manager users. The user gets all the required menus and privileges to manage the store through this responsibility. The menu assigned to this responsibility is called IBE\_ADMIN\_MENU. Follow the instructions in [Section 7.2, "Setting Up Store Manager User Accounts"](#) to create store manager users.

If you need to modify the functions that the user can perform, create a new responsibility for the user in the Application Object Library (AOL) module. For the new responsibility you can assign the default menu (IBE\_ADMIN\_MENU) and still remove access to some of the tabs ("functions" in AOL terminology), or you can

create a new menu using the Oracle iStore 11i functions. All Oracle iStore 11i functions can be found by searching for `IBE_%` in the Form Functions window.

See *Oracle Applications System Administrator's Guide, Release 11i* for more details on AOL and managing responsibilities.

## 2.4 Setting Up the Optional Dependencies

In addition to mandatory dependencies, Oracle iStore 11i also depends upon the following modules to provide additional functionality:

- Oracle Advanced Inbound
- Oracle Advanced Supply Chain (Global ATP Server)
- Oracle Configurator
- Oracle Contracts
- Oracle CRM Business Intelligence
- Oracle iMarketing
- Oracle Incentive Compensation
- Oracle iPayment
- Oracle iSupport
- Oracle Marketing Online
- Oracle Material Requirements Planning
- Oracle Shipping
- Oracle Workflow

### 2.4.1 Setting Up Oracle Advanced Inbound

Oracle iStore 11i provides call-me-back functionality whereby customers can access a call being placed in the call center. Please see *Oracle iSupport Concepts and Procedures* for details.

## 2.4.2 Setting Up Oracle Advanced Supply Chain Planning for Global ATP

Set up Oracle Advanced Supply Chain Planning for Global ATP if you want to provide global inventory availability information to your customers. As part of the rules for determining availability, you can provide sourcing rules that encompass orders already placed, open purchase orders, and other availability factors.

To set up the dependencies for ATP, set up ATP in Oracle Manufacturing, define ATP sourcing rules in Oracle Inventory, and enable product items for ATP by setting their ATP and ATP component flags. For more details, refer to *Oracle Advanced Planning and Scheduling and Oracle Global ATP Server Implementation Manual*, *Oracle Advanced Supply Chain Planning and Oracle Global ATP Server User's Guide*, *Oracle Inventory User's Guide*, *Oracle Order Management User's Guide*, and *Oracle Master Scheduling/Planning Guide*.

You must also set up Oracle Material Requirements Planning. See [Section 2.4.7, "Setting Up Oracle Material Requirements Planning"](#) for more information.

## 2.4.3 Setting Up Oracle Configurator

Oracle Configurator is used to create product models and to help the buyer assemble related and dependent products in the shopping cart.

The Configurator developer UI helps create the product models for dependent and related products, and build rules around the products.

The product models are imported initially from BOM models. The Configurator developer UI can be used to create a tree structure for the product model. For example, if a customer wants to build their own laptop from a store of electronic items, then the developer UI of the Configurator can be used to help define the structure and the dependent (mandatory and optional) products (e.g., a 15" or 17" screen with 16MB or 32MB RAM and 6GB or 8GB drive).

When the customer is browsing items in the featured section, a **Configure** button appears on the store UI if the item has the MODEL flag set in Oracle ERP applications. For every item, Oracle iStore 11i sends a call to Configurator to find whether a Configurator UI is associated with the item. If a Configurator UI is associated with an item, the store UI adds the **Configure** button or a link.

When the user clicks the **Configure** button, the store sends another message to populate the Configurator UI with item ID. The Configurator UI appears in the frame and the related or dependent products can be assembled for placing the order. There is a top menu bar with buttons such as **Save** and **Done**. Once the customer has finished building the list of selected items for the order, clicking the **Save** button places all the items in the shopping cart.

For more information, see *Oracle Configurator and SellingPoint Administration Guide, Release 11i and 4.2.2* for detailed setup documentation.

**Changing the Configurator UI** The Configurator UI appears in the frame provided on the Oracle iStore 11i featured section. This Configurator window is created with DHTML or a Java Applet. The look and feel are similar to the Oracle iStore 11i UI. If you want to change the Configurator window, you can modify the HTML templates for Configurator. These templates are loaded in Oracle iStore 11i's `html` directory, on the same server, by default. The Configurator data is stored in the CZ tables in the APPS schema.

**Profile Setup for the Configurator UI** The only setting required for Oracle iStore 11i to get the Configurator UI is a URL that handles all interaction between the client and the server. For example, the URL could be

```
http://apps-server-host/apps/cz/oracle.apps.cz.servlet.UiServlet.
```

The server and directory structure are installation information that the calling application must read, but the `oracle.apps.cz.servlet.UiServlet` portion is always the same. This URL is read from the `CZ_UIMGR_URL` profile value.

The setup needed to run the Configurator UI from the Oracle iStore 11i Customer UI is the setting of the above mentioned profile variable.

### Setup Steps

Once the rapid install has been done, the Configurator servlet is set up and tested, and the UI built, complete the following steps:

1. Using the Forms application, every item that can be shown for this configured item (from the root model item, every option class, and every leaf node) must be:
  - Web Status = PUBLISHED and Orderable on the Web = Y (checked). Using the Inventory responsibility, set these flags in the **Items > Master Items** (choose the org) > **Web Option** tab.
  - Added to the price list that will be used by the store and customer (even if it is there with a zero price)
  - For the IBE\_CUSTOMER responsibility, the profile option BOM: Configurator URL of UI Manager must be set to the following:

```
http://<machine>:<port>/servlets/oracle.apps.cz.servlet.UiServlet
```

2. Using the Merchant UI, the model item must be added to some part of the catalog to be displayed.
3. The .dbc file to connect to the database must be in place in the secure directory under the directory defined as `fnktop`.
4. The `jserv.properties` file must have the template URL defined as follows (these URLs must be able to be resolved when entered into a browser):

```
wrapper.bin.parameters=-Dcz.uiservlet.templateurl=http://<machine>:<port>
/OA_HTML/US/czFraNS.htm
wrapper.bin.parameters=-Dcz.uiservlet.templateurl.ie=http://<machine>:<port>/OA
_HTML/US/czFraIE.htm
```

### Testing the Configurator Setup

1. Enter the following URL in the browser:

```
http://<machine>:<port>/servlets/oracle.apps.cz.servlet.UiServlet?test=version
```

The browser should return the following statement:

```
Using configuration software build: 11.5.1.14.27 Expecting schema: 14c
```

This informs the merchant whether or not the Configurator middle tier servlet is up and running.

2. Edit the fields in the Configurator Standalone Test page, open it in a browser, and click the **Launch DHTML** button. The UI should appear.

## 2.4.4 Setting Up Oracle Contracts

You can integrate Oracle iStore 11*i* with Oracle Contracts to provide contract negotiation and agreement functionality in Web stores. This contract functionality enables communication through Oracle iStore 11*i* between the merchant and the customer regarding the terms and conditions of a contract.

When you set up Oracle iStore 11*i* to have contract negotiation functionality through Oracle Contracts, you must first set up a standard contract template with Oracle Contracts for use with all initial quotes.

When customers proceed to checkout with a shopping cart, Oracle iStore 11*i* displays the terms and conditions of the standard contract in the order review page and asks the customers to agree or disagree.

If the customers agree with the terms and conditions, they can place the order. At this point, a standard contract in the signed state is created if the profile option IBE: Create Standard Contract is set to **Yes**. The standard contract is associated with the quote number.

If the customers disagree with the terms and conditions, the following sequence of events takes place:

1. Oracle iStore 11i forces the customers to save the shopping cart.
2. The customers must enter a reason for their disagreement in a text box. These comments are associated with the contract created by the application.
3. The standard contract is associated with the quote.
4. Oracle iStore 11i sends an e-mail to the sales representative, contract specialist, and customer with the quote number and contract ID. The sales representative's e-mail is specified in the Oracle Human Resources Forms for each of your organizations.
5. The contract specialist looks at the quote and the customer's reason for disagreeing with the standard terms and conditions. The customers cannot checkout the cart unless the requested contract changes are approved.
6. If the contract specialist approves the requested change in terms and conditions, the customer is notified and can then retrieve the shopping cart. At this point, the customer can either place the order, which displays the revised contract, or disagree with the terms and conditions again, which repeats the process of notifying the contract specialist and sales representative. The customer cannot otherwise modify the cart.
7. If the contract specialist rejects the requested change in terms and conditions, the customer is notified and will not be able to retrieve the shopping cart.

### Prerequisites

You have created a contract template in Oracle Contracts with an Oracle Contracts logical name. See *Oracle Contracts Core Concepts and Procedures* for more information.

### Steps

1. Set up the IBE profile options associated with Oracle Contracts integration:
  - Set the profile option ASO: Enable Use Contracts to **Yes**.
  - Set the profile option OKC: Contract Template Name For Terms as the Oracle Contracts logical name of the contract template.

- Set the profile option IBE: Create Standard Contract according to your business needs.

See [Chapter 7](#) for more information about setting profile options.

2. Set up the default sales representative user who will receive the contract-related notifications, for each of your organizations, as follows:
  - a. Launch Oracle Forms by navigating to:  
`http://<host>:<apache port>/`  
  
and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
  - b. Log in with the Human Resources responsibility.
  - c. Choose **Work Structures > Organization > Description**.  
The Find Organization window opens.
  - d. In the Find Organization window, enter the search criteria for your merchant organization and click **Find**.  
The Organization form opens with the organization's record.
  - e. In the Organization Classifications region, select **Operating Unit** in the Name fields.
  - f. Click **Others**.  
The Additional Organization Information window opens.
  - g. In the Additional Organization Information window, choose **Notification User** and click **OK**.  
The Additional Organization Information - Notification User window opens.
  - h. In the Additional Organization Information - Notification User window, place your cursor in the Notification User field.  
The Notification User window opens.
  - i. In the Notification User window, in the Contracts field, use the LOV to enter the user name of the sales representative who should receive notifications of terms change requests for the organization, and click **OK**.
  - j. In the Additional Organization Information - Notification User window, click **OK**.

- k. Click the Save icon in the toolbar to save the record.

## 2.4.5 Setting Up Oracle iMarketing

You can use Oracle iMarketing to personalize the store and make recommendations. You create postings in Oracle iMarketing, create rules that determine the content for a given posting, and then modify Oracle iStore 11i templates to make reference to posting tags.

See *Oracle Marketing Online Implementation Guide, Release 11i* for more details.

## 2.4.6 Setting Up Oracle iPayment

If you are planning to provide credit card payment options, then you must set up Oracle iPayment to perform credit card authorization and fund capture. You can set up authorization to be done when the order is being placed or to be deferred to a later time. Refer to [Section 5.11.4, "Setting Up Credit Card Payments in Oracle iStore 11i"](#) for instructions on configuring the behavior.

Also see *Oracle iPayment Implementation Guide, Oracle Order Management User's Guide*, and *Oracle Account Receivables User's Guide* for additional setups required for Oracle iPayment. In addition to Oracle iPayment setup, you will need to set up the Oracle iPayment system itself to talk to the provider networks.

## 2.4.7 Setting Up Oracle Material Requirements Planning

Set up Oracle Material Requirements Planning for global ATP if you want to provide global inventory availability information to your customers. As part of the rules for determining availability, you can provide sourcing rules that encompass orders already placed, open purchase orders, and other availability factors.

To set up the dependencies for ATP, set up ATP in Oracle Manufacturing, define ATP sourcing rules in Oracle Inventory, and enable product items for ATP by setting their ATP and ATP component flags. For more details, refer to *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning User's Guide, Oracle Inventory User's Guide, Oracle Order Management User's Guide*, and *Oracle Master Scheduling/Planning Guide*.

You must also set up Oracle Advanced Supply Chain Planning. See [Section 2.4.2, "Setting Up Oracle Advanced Supply Chain Planning for Global ATP"](#) for more information.



## 2.4.8 Setting Up Oracle Shipping

The Oracle Shipping module must be in place to enable post-order tracking and shipping detail views in Oracle iStore 11i. At a minimum, you must specify shipping methods and descriptions in Oracle Shipping. See *Oracle Shipping Execution User's Guide, Release 11i* for more information.

## 2.4.9 Setting Up Oracle Workflow

Oracle Workflow sends e-mail confirmations to customers upon registration and order submission. Oracle Workflow also sends e-mail alerts to parties that share a shopping cart. The iStore Alerts Workflow in Oracle Workflow calls the PL/SQL package IBEVWFB.pls.

At a minimum, you must set up message text in Oracle Workflow for Oracle iStore 11i e-mail notifications. If you want to customize the e-mail confirmations that are sent to customers when they register or place an order, open the Oracle Workflow message template IBENOTIF.wft. If you want to customize the e-mail alerts that are sent to customers when they are named as sharees of a shared shopping cart, open the Oracle Workflow message template IBEMAIL.wft.

See *Oracle Workflow Guide* for more information.

The following table lists the notifications that are available in Oracle iStore 11i.

**Table 2–1 Oracle Workflow Notifications Available in Oracle iStore 11i**

Notification	Description
Account Registration Notification	Customers receive this notification after registration.
Customer Notification on Request for Sales Assistance	Customers receive this notification as an acknowledgment of their requests for sales assistance.
Notification for Customer Quote	Customers receive this notification upon their requests to change contract terms.
Order Confirmation Notification	Customers receive this notification as a confirmation of their orders.
Order Confirmation for Faxed Orders	Customers receive this notification as a confirmation of their faxed orders.
Orders Not Booked Notification	Customers receive this notification when their orders are not booked.

**Table 2–1 Oracle Workflow Notifications Available in Oracle iStore 11i**

<b>Notification</b>	<b>Description</b>
Sales rep Notification on Request for Sales Assistance	Sales representatives receive this notification regarding customers' requests for assistance.
Notification to Sales rep for Customer Quote	Sales representatives receive this notification regarding customers' requests for contract term changes.
Cart Sharee Email	Sharees of shared shopping carts receive this notification.

After setting up Oracle Workflow, you must set the profile options IBE: Use Workflow Features in iStore and IBE: Use CABO UI to **Yes**. You must also set the profile options IBE: Default Order Admin to Send Workflow Notification and IBE: Default Sales Assistant to Send Workflow Notification. See [Chapter 7](#) for more information about setting profile options.

---

---

# Installation and Dependency Verification

Before you start configuring the profile options and creating the product catalog in Oracle iStore 11i, check that you have correctly completed the following prerequisites.

## 3.1 Checkpoints

1. Verify that the installation and middle tier setup have been done correctly.  
Once the Rapid Installer Wizard finishes the installation, verify that the proper installation and configuration of the following components:
  - **Apache Server:** Go to `http://<host>:<apache port>/apachedocs/`. You should see the Apache Server documentation page.
  - **Apache JServ:** Go to `http://<host>:<apache port>/servlets/IsItWorking`. You should see a message reassuring you that Apache JServ is working.
2. Verify that the ERP applications are installed and functioning properly. Refer to the *Oracle CRM: ERP Functional Checklist* document available on Oracle MetaLink for a description of the tasks required.
3. Verify that the setup dependency has been done correctly.

Place an order through Sales Order bench. If it goes through correctly, then your core dependency has been done correctly. To find how to place an order, refer to *Oracle Order Management User's Guide*.



---

---

## Oracle iStore 11*i* Setup

This chapter describes the tasks required to set up Oracle iStore 11*i* using application default settings and components, after you have verified your installation and dependency setup. Topics include:

- [Overview of Store Creation](#)
- [Accessing the Merchant UI](#)
- [Creating Specialty Stores](#)
- [Creating the Hierarchy](#)
- [Setting Up the Product Catalog](#)
- [Testing the Store](#)

---

---

**Note:** When using the Oracle iStore 11*i* Merchant UI, ensure that cookies are enabled for your browser. See the relevant browser documentation for information on enabling cookies.

---

---

## 4.1 Overview of Store Creation

Oracle iStore 11i ships with templates and defaults that allow you to develop a basic store. You can customize and add functionality as required. The Merchant UI enables you to perform the Oracle iStore 11i setup tasks outlined in this chapter and [Chapter 5](#) to set up the Customer UI.

The following prerequisites must be met prior to creating your initial store:

### Prerequisites

- Define JTF profiles. See [Chapter 7](#) for details.
- Define IBE profiles. See [Chapter 7](#) for details.
- Run concurrent jobs. See [Chapter 7](#) for details.
- Set up shipping options. See *Oracle Order Management User's Guide* and *Oracle Shipping Execution User's Guide* for details.
- Set up payment options. See *Oracle Order Management User's Guide* and *Oracle iPayment Implementation Guide* for details.
- Set up price lists and currencies in Oracle Pricing. See *Oracle Pricing User's Guide* and *Oracle Order Management User's Guide* for details.

You must select a price list for each type of customer on the Oracle iStore 11i Currencies and Price Lists page. Only maximum order limit is optional.

- Set up languages in Oracle AOL. See *Oracle Applications Concepts* for details.  
Set a default language for the store.
- Set up business units in Oracle General Ledger. See *Oracle Applications Concepts* and *Oracle General Ledger User's Guide* for details.

---

---

**Note:** If the information for these prerequisites is unknown, you can continue with the setup now and revise this information later.

---

---

You can create a basic specialty store using Oracle iStore 11i as follows:

### Steps

1. Access the Merchant UI.
2. Set up a specialty store.
3. Set up an overall hierarchy for the store sections and products.

4. Build the product catalog.
5. Test the Customer UI.

## 4.2 Accessing the Merchant UI

After installing Oracle iStore 11i and setting up the Merchant UI as detailed in [Chapter 2](#), you can enter the Merchant UI by logging in to:

`http://<host>:<apache port>/OA_HTML/jtflogin.jsp`

with a user name that the system administrator has set up as an Oracle iStore 11i store manager user account. See [Section 7.2, "Setting Up Store Manager User Accounts"](#) for more information on creating a store manager user account.

**Figure 4–1 The Oracle iStore 11i Merchant UI**

The screenshot displays the Oracle iStore 11i Merchant UI interface. At the top left is the Oracle iStore logo. To the right are 'Sign Out' and 'Help' links. Below the logo is a navigation bar with tabs for 'Specialty Stores', 'Multimedia Components', and 'Display Styles'. Above the main content area are tabs for 'Setup', 'Category', 'Hierarchy', 'Relationship', 'Product', 'Templates', 'Multimedia', and 'Cache'. The main content area is titled 'Specialty Stores' and features a table with the following structure:

Remove	Specialty Store Name	Specialty Store Code	Description
<input type="checkbox"/>	<a href="#">Sample Specialty Store</a>		Sample Specialty Store

Buttons for 'Create', 'Update', and 'Restore' are located below the table.

## 4.3 Creating Specialty Stores

A specialty store is any Web store. You can create multiple stores, for example a main store, a store for one large customer, a holiday specials store, and a store that requires registered users. You must create at least one store.

Multiple currencies and languages can be selected for every specialty store. The customer's preference, as defined in his or her user profile, determines which currency and language is to be used for a store. Once a registered customer selects a preferred language, the store defaults to that preferred language each time the customer enters. When the user preference is not set, default language and currency settings will take effect.

Users can also change their display languages and currencies by choosing from a list of the languages and currencies supported by the store. When a customer changes the display language and currency, Oracle iStore 11i changes his or her preferred language and currency to match the newly selected display language and currency. This change enables consistency across customer sessions and Oracle applications. For example, this will ensure that the order confirmation alert in a German language store will be in German, even if the customer's previous preferred language was French.

Customers must also choose the responsibility with which they enter a specialty store. The Customer UI shows a list of available combinations of specialty stores and Oracle iStore 11i responsibilities. When the customer chooses one of these combinations, he or she enters that specialty store with that responsibility for the current session. The responsibility determines the operating unit against which any orders placed in the current session are booked. The responsibility also determines the values of the profile options set at the responsibility level.

You can set up specialty stores to check the customers' responsibilities when they log in. For every specialty store flagged to check the user's responsibility, the Customer UI excludes those specialty store-responsibility combinations containing responsibilities not associated with the user.

---

---

**Note:** If the customer can only access one specialty store-responsibility combination, he or she is automatically forwarded to that specialty store's home page with that responsibility and does not see a list of specialty store-responsibility combinations.

---

---



Use Oracle Forms to create responsibilities and assign them to customer user names. See *Oracle Applications System Administrator's Guide, Release 11i* and *Oracle CRM Foundation Concepts and Procedures, Release 11i* for more information.

Use the following procedure to create a basic specialty store.

---

---

**Note:** Specialty stores are also referred to as "minisites."

---

---

### Steps

1. Launch the Merchant UI.
2. In the Setup tab, choose **Specialty Stores**.
3. Click **Create**. The Specialty Store Detail screen appears.
4. Enter the basic information for the specialty store in the following fields. The fields marked with an asterisk in the Merchant UI are mandatory.
  - a. Specialty Store Name: Enter the name of the specialty store.
  - b. Specialty Store Code
  - c. Description: Enter a description for the specialty store.
  - d. Start Date: Enter the date when the specialty store should first be active and available to customers.
  - e. End Date: Enter the date when the specialty store should no longer be active and available to customers.
5. In the Languages section, check the Select checkbox next to each language that you want the specialty store to support.
6. In the Default Language pull-down menu, choose the default language for your specialty store.
7. Click **Continue**. The Store Flags screen appears.

This screen is used to select the root section for the specialty store from the Oracle iStore 11i hierarchy and to determine whether the specialty store will:

- Be ATP Enabled for Oracle Inventory
- Allow walk-in customers (customers who have not logged in or registered)
- Check user responsibility

8. From the ATP Enabled pull-down menu, choose **Yes** if you want this store to provide ATP inventory information to the customer.
9. From the Walkin Customers Enabled pull-down menu, choose **Yes** if you want this store to allow walk-in customers who are not registered or logged in.
10. From the Check User Responsibility pull-down menu, choose **Yes** if you want the store to check user responsibility when customers log in.
11. Click **Go** next to the Root Section field.

A pop-up window displays the Oracle iStore 11i hierarchy.

12. Search for and highlight the root section of the store, and click **Done**.

The pop-up window closes, and the Root Section field is populated with the name of the section you have chosen.

---

---

**Note:** A root section is required for each specialty store.

---

---

13. Click **Continue**.

The Supported Responsibilities screen appears.

14. Click **Add Responsibility**.

The Select Responsibility pop-up window opens.

15. In the Select Responsibility pop-up window, search for responsibilities that you want this specialty store to support by application and responsibility name, key, or description, using the wildcard character % if necessary. Check the Select checkbox next to the responsibilities, and click **Add**. When you are finished selecting responsibilities, click **Done**.

You return to the Supported Responsibilities page.

16. In the Display Name fields, enter user-friendly names by which each specialty store-responsibility combination will appear in the Customer UI.
17. In the Start Date and End Date fields, enter the dates when the specialty store will support each responsibility you have added.
18. In the Order fields, specify the order in which these responsibilities will appear on the customer login page.

19. Click **Continue**.

The Access Restrictions page appears.

**20. Click Add Organization.**

The Select Organization pop-up window opens.

- 21.** In the Select Organization pop-up window, search for organizations by name or account number, using the wildcard character % if necessary. Check the Select checkbox next to the organizations you want to add to the list, and click **Add**. When you are finished selecting organizations, click **Done**.

You return to the Access Restrictions page.

**22.** Highlight one of the three radio buttons:

- No Restriction, if you want this specialty store to allow users from any organization
- Include the following organizations, if you want this specialty store to allow only users from the organizations you specify in this page
- Exclude the following organizations, if you want this specialty store to deny access only to users from the organizations you specify in this page

- 23.** In the Start Date and End Date fields, enter the dates when the inclusion or exclusion of the listed organizations is effective.

**24. Click Continue.**

The Currencies and Price Lists page displays available currencies.

- 25.** Choose currencies by checking the Select checkbox next to the currencies that you want this specialty store to support.

- 26.** For each selected currency, choose the price lists for Walk-in Customer, Registered Customer, and Business Partner from the pull-down menus.

- 27.** Optional: Enter a maximum orderable limit for each selected currency.

- 28.** Choose the default currency for the store from the Default Currency pull-down menu.

**29. Click Continue.**

The new specialty store is saved.

To modify an existing specialty store, click on its name in the **Setup > Specialty Stores** section of the Merchant UI and change the information as needed. Click on **Update** instead of **Continue** in the specialty store information pages to save your changes.

## 4.4 Creating the Hierarchy

After saving the specialty store, you must define an overall hierarchy in the Oracle iStore 11*i* Merchant UI Hierarchy tab. This hierarchy determines the organization of your specialty stores and their sections and products in the Merchant UI. It also determines the organization and presentation of each store's sections and products in the Customer UI.

The overall hierarchy contains products from Oracle Inventory, grouped into sections. Associate a specialty store to a portion of the overall hierarchy or to the whole hierarchy itself by setting up its root node to point to a section. The hierarchy determines the browsing experience of the customer and what products are featured at different levels in the store. When users come to a specialty store, they see the hierarchy starting from the root node of the store. You can choose not to show a particular section in a specialty store even though the given specialty store might point to an ancestor of the section. In the templates shipped with Oracle iStore 11*i*, the top level appears as tabs while the lower level appears as browse bins on the store pages.

Set up the top level sections in the hierarchy first. For each top level section, create as many subsections or children as you wish. The levels of sections are driven by the design.

You can assign products to sections of the hierarchy from the Hierarchy or Product tabs. Similarly, you can create groups of featured products at any level in the hierarchy by creating a subsection of type Featured in that section. The products in a section are shown by using the display style that you specify for that section.

---

---

**Note:** In the Customer UI for Oracle iStore 11*i*, the minisite root sections are treated as virtual roots. The current minisite's root section will not appear in the menu tabs or the navigational hierarchy of the Customer UI. The minisite's home page will display the first navigational subsection under the root section, not the root section page itself. To present featured sections on the minisite's home page, make them subsections of this navigational subsection.

---

---

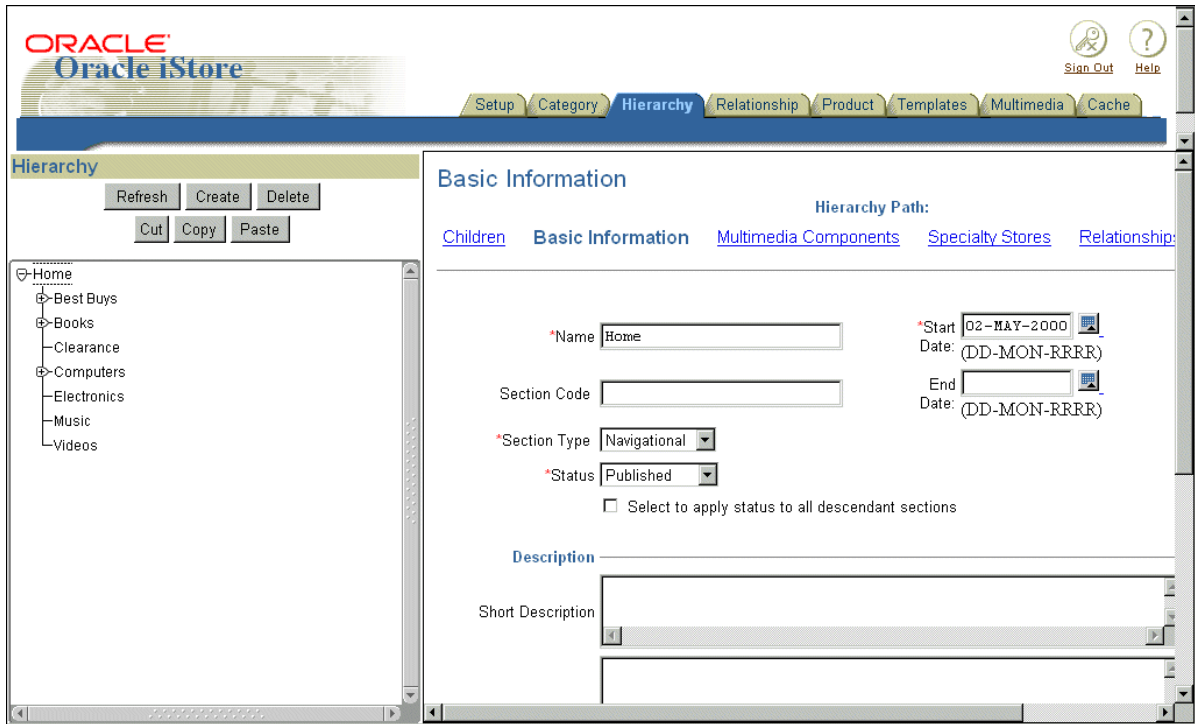
In the templates shipped with Oracle iStore 11*i*, the top level appears as tabs while the lower level appears as browse bins on the store pages. A section containing subsections shows the featured products in the middle and the subsections in the left browse bin. If a section contains only products, it lists the products in the middle. Featured sections cannot have subsections.

While working with the hierarchy, you can determine whether or not to publish sections. A published section is available in the Customer UI. An unpublished section is not available in the Customer UI, unless the user has the IBE\_ADMINISTRATOR responsibility. Thus, you can choose to keep a section unpublished until you have tested its appearance in the Customer UI.

You can also specify whether the descendant sections have the same published/unpublished status as the section on which you are currently working. If a section is unpublished, its descendant sections are effectively unpublished, since a user cannot navigate to the descendant sections in the Customer UI (unless he or she has the IBE\_ADMINISTRATOR responsibility). However, if a user knows the exact URL to access a descendant section, the user can access the descendant section if it is published.

If any section is not published, a user lacking the IBE\_ADMINISTRATOR responsibility cannot access the section even if he or she has the exact URL.

**Figure 4–2 The Oracle iStore 11i Merchant UI Hierarchy Tab**



### Prerequisites

Profile set up for IBE:Item Validation Organization (see definition for IBE\_ITEM\_VALIDATION\_ORGANIZATION profile).

### Steps

1. Launch the Merchant UI and enter the Hierarchy tab.
2. In the left frame of the Hierarchy tab window, the overall hierarchy tree appears. In the tree, select the node that will host the node you want to create, and click **Create**. (To modify existing information, select the node itself.)
3. The Basic Information Screen appears in the right frame of the Hierarchy tab window. Enter the following information:
  - a. Name: The name of the section.

- b. Section Code: Leave this blank. (To refer to the section in customized templates explicitly by name, use its Section Code.)
- c. Section Type: Select either **Featured** or **Navigational**. A featured section appears on the home page of its parent section. A navigational section appears as a link in the browsing map of its parent section.
- d. Status: Select either **Published** or **Unpublished**.
- e. Select to apply status to all descendant sections: Check this checkbox if you want all descendant sections of this section to have the same status.
- f. Start Date Active and End Date Active: Specify the time limit (if any) when this section will be active.
- g. Short Description and Long Description: Fill these in to describe the section in the store. The descriptions will appear on the section pages.
- h. Keywords: Leave this blank.
- i. Template for displaying this section: Several sections can share a template. Leave this blank to revert to the default store-level template.
- j. Display Style for products in this section: Select the display style you want to use to display products in this section. Leave this blank to revert to the default store-level display style for products.

Click **Continue**. The Multimedia Components screen appears.

4. In the Multimedia Components screen, leave the fields blank to use the default multimedia for the multimedia components. Click **Continue**. The Specialty Stores screen appears.
5. Move the specialty store(s) in which this section should appear into the Included Specialty Stores column.

Decide if the section should appear only in the specialty stores that you have selected here or in other specialty stores if those specialty stores' roots point to an ancestor of the current section. Check the box **Include in all future sites if the site's root section is ancestor of this section** as appropriate. Click **Continue**.

The Relationships screen appears with a list of existing relationships.

6. Review relationship rules for this section. See [Section 4.5.1, "Using Seeded Relationship Types"](#) and [Section 5.7, "Creating Relationships"](#) for more information. Click **Continue**.

The Advanced Settings screen appears.

7. Leave the fields blank and click **Finish**.

The Children page appears.

8. Now you have created sections and can add either subsections or products, but not both. Featured sections cannot have subsections as children.

You can add products to a section in the Hierarchy tab after building the product catalog. See [Section 4.5, "Setting Up the Product Catalog"](#) for more information. To add products to a section in the Hierarchy tab:

- a. Navigate to the section's Children page.
- b. Click **Add Product**.
- c. Perform a product search in the pop-up window.
- d. Select one or more of the results and click **Add**.
- e. Choose **Done** to close the pop-up window and return to the main store manager window.

You can also add products to a section when working on the product in the Product tab.

To add a subsection to a section:

- a. Navigate to the section's Children page.
- b. Click **Add Section**. A Basic Information screen for the new subsection appears.

---

---

**Note:** The **Add Section** button appears only if the section has no products.

---

---

- c. Set up the subsection in the Basic Information, Multimedia Components, Specialty Stores, Relationships, and Children screens in the Hierarchy tab.
9. To adjust the order of the section tabs in the specialty store, click its root section in the navigation tree and edit the values in the Order column. The section whose display name should be the first tab should have the lowest number in the Order column.
10. Continue to create sections and subsections for your hierarchy as needed. You can use the **Cut**, **Copy**, and **Paste** buttons to move an entire section from one place in the hierarchy to another.



To cut or copy a section, highlight the section and click **Cut** or **Copy** as appropriate.

To paste a section into the hierarchy after cutting or copying it, highlight the root section that should host the pasted section, and click **Paste**.

---

---

**Note:** If you cut and paste a section from one location in the hierarchy to another, all the information about the section will transfer to the new location. If you copy and paste a section, all the information about the original section will copy into the new section, except for the section-level multimedia component settings.

---

---

### Example

1. In the Hierarchy tab, select the **Home** section in the navigation tree in the left frame.
2. In the right frame, click **Children**.
3. Click **Add Section** in the right frame to create a new section.
4. Enter Featured Products as the name and the code.
5. Select **Featured** as the section type. Everything else is optional. Click **Continue**.
6. On the Multimedia Components screen, everything is optional. Click **Continue**.
7. Accept the defaults on the Specialty Store and Navigation Relationships screens. Click **Continue**.  
The Advanced Settings screen appears. Click **Finish**.
8. In the left frame, click **Refresh**. Expand the Home node, which should have the newly-created section under it.
9. Highlight **Home**, and click **Create**. Repeat the above steps to create another section named Books, with type Navigational.
10. Repeat again for Music, Electronics, and Computers, making them all navigational sections.

## 4.5 Setting Up the Product Catalog

Setting up the product catalog involves the following considerations:

- Designing screen flow and navigation.
- Determining product items to be sold, their display features, and configuration options.
- Determining types of data required. For example, books may require a title, author, and publisher. See *Oracle Inventory User's Guide* for details on how to use the flexfield structures in Inventory to store and sort data accordingly.

Use this procedure to add products to the product catalog and make them available for sale in your store.

While working with the product catalog, you can determine whether or not to publish products. A published product is available in the Customer UI, assuming that it is also listed in at least one published section. An unpublished product is not available in the Customer UI, unless the user has the IBE\_ADMINISTRATOR responsibility. Thus, you can choose to keep a product unpublished until you have tested its appearance in the Customer UI.

### Prerequisites

- Products must be loaded into Oracle Inventory before they can be imported into Oracle iStore 11i.
- Products in Oracle Inventory must have their Web Status flag set to either Published or Unpublished in the Web Option tab of the Master Item form to appear in the Oracle iStore 11i Merchant UI. Only products with a status of Published can be sold in your store.
- Products in Oracle Inventory must be set with the flag Orderable on Web in the Web Option tab of the Master Item form.
- Products in Oracle Inventory must be set with the flag Customer Orders Enabled in the Order Management tab of the Inventory Master Item form.
- JTF and IBE profile options must be set. See [Chapter 7](#) for details.
- Shipping options must be seeded into Oracle Shipping. See *Oracle Order Management User's Guide* and *Oracle Shipping Execution User's Guide* for details.
- Payment options and setup must be seeded into Oracle iPayment. See *Oracle Order Management User's Guide* and *Oracle iPayment Implementation Guide* for details.

- Store layout must be determined, and the hierarchy of products, sections, and specialty stores must be identified. See [Section 4.4, "Creating the Hierarchy"](#) for details.

## Steps

1. Launch the Merchant UI and enter the Product tab.

The Products page appears.

2. Search for products you want to include in your catalog.

The search criteria are:

- Name
- Part number
- Belongs to category
- Created after date  
Use date format DD-MON-RRRR.
- Created before date  
Use date format DD-MON-RRRR.
- Status - values should be PUBLISHED or UNPUBLISHED
- New - products created in last x days where x is the profile value of IBE:  
Number of Days for New Items

---

---

**Note:** % can be used for a wild card character search.

---

---

The Products page lists products in Oracle Inventory that match your search criteria and displays existing product catalog information for those products.

If no products appear on this screen, it is probably because the WEB\_STATUS flag in the MTL\_SYSTEM\_ITEMS table is NULL. For items to appear in the Merchant UI, they should have a Web Status of Published or Unpublished. To make an item show up in the Merchant UI:

- a. Log in to Oracle Forms.
- b. Choose the Inventory responsibility for the Master Inventory Organization.
- c. Choose **Items > Master Items**.

- d. Choose the Master Inventory Organization.
- e. Choose F11 to enter a search query, then choose Ctrl-F11 to execute it.
- f. Use the Web Option tab to publish the desired item(s).

If the Web Option tab in the Master Items Form does not work properly, apply the latest Oracle Inventory patchset.

---

---

**Note:** The value for Inventory Organization ID should be the same as IBE\_ITEM\_VALIDATION\_ORGANIZATION profile.

---

---

If there are unpublished products in your results, the list shows them with a **Publish** button in the Wizard column. To publish an unpublished product, you can either click on the product name and change the Posting Status to **Published** in the Basic Information window that appears, or click on the **Publish** button next to the item name.

---

---

**Note:** Publishing or unpublishing a product in the Oracle iStore 11*i* Merchant UI also changes the product's Web Status setting in Oracle Inventory.

---

---

If you click the **Publish** button:

- a. The Basic Information screen appears.  
Optional: Add or modify the basic and long descriptions.  
Click **Continue**. The Hierarchy Paths window appears.
  - b. Optional: Add or remove parent sections for the item.  
Click **Continue**. The Category and Display Styles window appears.
  - c. Optional: Change template assignments for one or more of the display styles.  
Click **Continue**. The Multimedia Components page appears.
  - d. Optional: Change multimedia assignments for one or more of the multimedia components.
  - e. Click **Publish**.
3. Click on a product's name in the search results list to edit its information.

The Basic Information page displays the Posting Status, the inventory name and part number, and any descriptions you have already added to the product catalog.

4. To make the product available to be sold in your store, set the Posting Status to **Published**.

To remove the product from your store, set the Posting Status to **Unpublished**.

Optional: Enter or modify the short and long descriptions.

Click **Update**. The product is published or unpublished immediately. The descriptions are saved, and are also available for display in your store if the product is published.

---

---

**Note:** Products with Posting Status **Published** are visible on the store. Be careful about changing information, since the changes go to the production system and are published immediately. It is recommended that you unpublish the product before making changes, and then republish the item when finished.

---

---

5. In this product detail page in the Products tab, choose **Hierarchy Paths**.

The Hierarchy Paths page displays the hierarchy of sections that have been set up for the store.

6. Optional: Remove or add parent sections for the product, edit the date range when this product will be available in each section, and number the product's place in the section's product display order.

7. In this product detail page in the Products tab, choose **Category and Display Styles**.

The Category and Display Styles page displays the category to which the product belongs and lists all display styles and any template names already assigned to the product.

Leave fields blank to keep default templates for each display style that you want to use for the product. You can also choose item-level templates here. Click **Update**.

8. In this product detail page in the Products tab, choose **Multimedia Components**.

The Multimedia Components page lists all multimedia components that have been set up for Oracle iStore 11i.

Leave fields blank to assign default multimedia names to multimedia components. Click **Update**.

9. In this product detail page in the Products tab, choose **Relationships**.

The Relationships page displays existing relationships between the product and other products or sections and the rules for those relationships, such as the product to show for an upsell or cross sell.

See [Section 4.5.1, "Using Seeded Relationship Types"](#) to add related items for a relationship.

10. In this product detail page in the Product tab, choose **Specialty Stores**.

The Specialty Stores page lists the specialty stores where the product will be displayed. By default the product appears in those specialty stores to which the product's parent section belongs.

Select the specialty stores where the product should appear and click **Update**.

11. Repeat this procedure for every product you want to include in the product catalog.

## 4.5.1 Using Seeded Relationship Types

Relationships are used for merchandising, for example, to offer a substitute product for a product that is out of stock. Use relationships to associate products, categories, and sections with other products, categories, and sections. See [Section 5.7, "Creating Relationships"](#) for more information.

Oracle iStore 11i ships with several seeded relationships.

### Seeded Values

- RELATED: Entity B is related to Entity A.
- SUBSTITUTE: Entity B can be substituted for Entity A.
- CROSS\_SELL: Entity B can be offered and sold along with Entity A
- UP\_SELL: A newer version Entity B can be sold instead of Entity A.
- SERVICE: Entity B is a service item that can be added to the shopping cart for a serviceable Entity A.
- PREREQUISITE: Customer must have Entity B before purchasing Entity A.

- COLLATERAL: Entity B is collateral (e.g. marketing brochures) that exists for Entity A.
- SUPERSEDED: Entity B supersedes Entity A, which is no longer available.
- COMPLIMENTARY: Entity B is available free of charge with Entity A.
- IMPACT: Entity A is usable together with related Entity B, but only under certain conditions.
- CONFLICT: Entity A is not usable together with a related Entity B.
- MANDATORY\_CHARGE: Mandatory charge
- OPTIONAL\_CHARGE: Optional charge
- PROMOTIONAL\_UPGRADE: Entity A ordered by the customer is upgraded to Entity B of equal or higher value, with no change to the price.

These relationship types are also seeded in Oracle Inventory for Item Relationships. If you use Oracle iStore 11i's Java Application Programming Interface (API) to retrieve related items given an item ID and a seeded relationship type, you will get related items defined in Oracle iStore 11i plus the ones defined in Oracle Inventory.

### **Prerequisites**

Products must exist in Oracle Inventory.

### **Steps**

1. Go to the Relationship tab and review the seeded relationship types.
2. Click the name of a relationship type to create a relationship between items. For example, click SUBSTITUTE to make item B a substitute for item A if item A is out of stock.
3. Click **Add Rules**.
4. In the middle frame, search for the base product, and click the left arrow to add it to the From List.
5. Search for the related product, and click the right arrow to add it to the To List.
6. Click **Done** to save the relationship.

## 4.6 Testing the Store

Test the storefront with the following URL:

```
http://<host>:<apache port>/OA_HTML/ibeCZzpHome.jsp?minisite=<minisite ID>
```

where <minisite ID> is the ID of the specialty store created above. You can determine the minisite ID using the following procedure.

### Steps

1. Launch the Merchant UI and enter the Setup tab.
2. In the Specialty Stores screen, click the name of the specialty store that you want to test.

The Specialty Store Detail Basic Information screen appears.

3. Look at the URL of this screen. It will be in this format:

```
http://<host>:<apache port>/OA_HTML/jtffmbas.jsp?ID=<minisite ID>&ACTION=VIEW
```

The specialty store ID is the number that appears in the place of <minisite ID> in this URL.

See [Section 5.12, "Previewing Products and Sections"](#) for more information on testing the appearance of products and sections in your Web stores.



---

---

# Oracle iStore 11*i* Customization

This chapter describes procedures for customizing Oracle iStore 11*i* specialty stores. Topics include:

- [Overview of Store Customization](#)
- [Customizing Multimedia](#)
- [Cataloging Multimedia Components](#)
- [Customizing Templates](#)
- [Cataloging Display Styles](#)
- [Modifying the Hierarchy](#)
- [Creating Relationships](#)
- [Customizing Product Presentation at the Category Level](#)
- [Customizing Product Presentation at the Item Level](#)
- [Setting Up Product Searches](#)
- [Customizing the Shopping Cart](#)
- [Previewing Products and Sections](#)

---

---

**Note:** When using the Oracle iStore 11*i* Merchant UI, ensure that cookies are enabled.

---

---

## 5.1 Overview of Store Customization

Planning the customization of your store involves the following tasks:

- Identify the ways in which the store will display products.
- Plan page designs and divide them into common components that you can make into templates.
- Invent a name for each possible template to facilitate planning and communication of designs.
- Customize templates now as part of the design or later in the setup cycle.

The following procedure exemplifies the sequence of steps you can use to customize your store.

### Steps

1. Create proprietary media source files for use in the customized store. Some examples of media file types are small .gif, large .gif, descriptive text, audio, and video. The types of media you can use in the store depend on the capabilities of the browsers that will access it.
2. Catalog Oracle iStore 11*i* multimedia in the Multimedia tab to make them available for assignment to multimedia components. Each multimedia name cataloged can have a number of media source files assigned to it.
3. Define and catalog multimedia components under **Multimedia Components** in the Setup tab. Enter a default multimedia name for each multimedia component.
4. Create template source files for pages and for blocks within pages using Oracle JDeveloper or another Web page authoring application.
5. Catalog Oracle iStore 11*i* templates in the Template tab. Each template name cataloged can have a number of template source files assigned to it.
6. Define and catalog the display styles determined during planning, under **Display Styles** in the Setup tab. Enter a default template name for each display style.
7. If necessary, modify the overall hierarchy for your products using the Hierarchy tab. For example, you can add items to a section, remove items from a section, and create or delete sections.
8. Create new relationship types.
9. Customize product presentation at the category level.

10. Customize product presentation at the item level.
11. Set up a product search for the Customer UI.
12. Customize shopping cart presentation and functionality.

## 5.2 Customizing Multimedia

Multimedia consist of files such as graphics, text, audio, and video, that are used to present content on a Web page to your customer.

The Oracle iStore 11i multimedia catalog enables you to make customized multimedia available for use in your stores and organize the multimedia according to specialty stores and languages.

To customize the appearance of your store pages, you must perform these tasks:

- Create proprietary media source files.
- Catalog Oracle iStore 11i multimedia and assign multimedia source files to each multimedia object.

### 5.2.1 Creating Media Source Files

Creating your own media source files for use in your store pages can enhance the appearance of your store to serve better the store's purposes.

Types of media source files can include small graphics (.gif), large .gif, descriptive text, audio, and video. You can create these files through media authoring programs.

All media source files should be placed in the OA\_MEDIA directory.

### 5.2.2 Cataloging Multimedia

Each multimedia object that you list in the Oracle iStore 11i multimedia catalog can have a number of media source files assigned to it. Each of these source files can be assigned to combinations of specialty stores and languages. Each multimedia object is in turn available for assignment to multimedia components, which are called by templates to determine which multimedia object appears on a given store page.

## Naming Multimedia

The multimedia name is the catalog name that is easy to communicate and use when planning your page designs. An example is *CompanyLogo*. This name can be translated for convenience in store administration.

Every multimedia name is given a programmatic access name that is short, unique, and not as descriptive. The programmatic access name is used to display that multimedia file in your Web page, if you want to refer to it directly in the template. An example is *clogo*. This name is not translated.

The multimedia name and programmatic access name represent several source files. You assign each source file to combinations of specialty stores and languages. The following table lists examples of file names for the multimedia name *CompanyLogo*.

**Table 5–1 Sample Media File Names for the Multimedia Name *CompanyLogo***

Multimedia Name	Programmatic Access Name	File	Specialty Store	Language
CompanyLogo	clogo	clog1f.gif	specialty store 1	French
CompanyLogo	clogo	clog1e.gif	specialty store 1	English
CompanyLogo	clogo	clog2f.gif	specialty store 2	French
CompanyLogo	clogo	clog2e.gif	specialty store 2	English

In this example, if a French customer enters specialty store 1, the store displays the logo file *clog1f.gif*. An English customer entering the same specialty store sees *clog1e.gif* instead.

To see the multimedia which have been seeded into Oracle iStore 11*i* and are available for use in your store, enter the Multimedia tab. This page lists the existing multimedia and their programmatic access names, keywords, descriptions, and default source files for all specialty stores and languages. Click individual multimedia names for more detail. Choosing **View All Mappings** from within an individual detail page displays each source file name and its relationship to specialty stores and languages.

Use this procedure to catalog Oracle iStore 11*i* multimedia and assign multimedia source files to the names.

## Prerequisites

- The default language must have been defined.
- At least one speciality store must have been created.

## Steps

1. In the Multimedia tab, search for multimedia that are already cataloged and available to use in your store.

The Multimedia page lists the multimedia that match your search criteria along with their programmatic access names, keywords, descriptions, and the default source files to use for all specialty stores and languages.

2. Click **Create**.

The Multimedia Details page appears.

**Figure 5–1 The Multimedia Details Page**

ORACLE  
Oracle iStore

Sign Out Help

Setup Category Hierarchy Relationship Product Templates Multimedia Cache

### Multimedia Details

#### Information and Source Files

\*Name  Keywords

\*Programmatic Access Name  Displays

Default Source File For All Sites and Languages  Description

3. In the Name field, define the multimedia name. Choose a name that is representative of the multimedia object's characteristics and purpose.

4. In the Programmatic Access Name field, define the programmatic access name, which is the name by which the multimedia object will be accessed from the template. Do not duplicate other programmatic access names.
5. In the Default Source File For All Sites and Languages field, define the default media source file by entering the location of the file relative to the `OA_MEDIA` directory, where all media source files should reside. For example, enter the GIF file `product.gif` from the `OA_MEDIA` directory as `/OA_MEDIA/product.gif`. This default source file will be used by Oracle iStore 11i, unless a specialty store and language has a specific media source file mapping. If only one language or specialty store is defined or if no specialty store has been created, use the defaults.
6. In the Displays pull-down menu, specify if this multimedia object will be available for product- and category-level (**Category**) or section-level (**Section**) presentation, or for page features not specifically associated with catalog presentation (**Others**).
7. Optional: In the Keywords field, enter keywords for the multimedia. Entering keywords enables a keyword-based search for this multimedia object when assigning a multimedia object name to a multimedia component.
8. Optional: In the Description field, enter a multimedia description. Entering a description enables a description-based search for this multimedia object when assigning a multimedia object name to a multimedia component.
9. Click **Update**. The Multimedia Details Information and Source Files window appears.
10. Optional: Click **Add Source File** to provide files for the same multimedia name in different languages and specialty stores. The Source File Details page appears.
  - a. Enter the name of a media source file, such as a graphic file, that you want to display on a Web page for the multimedia name that you are creating, for example, `/OA_MEDIA/video.jpg`. Click **Update**.
  - b. Add each specialty store and language where you want the new source file to appear, then click **Update**.
  - c. The relationship between the multimedia name, source files, specialty stores, and languages is saved.
  - d. Repeat this procedure for each source file that you want to add.
11. Optional: Choose **View All Mappings** in the Multimedia Details Information and Source Files window.

The View All Mappings page displays each source file name and its relationship to specialty stores and languages. This step is highly recommended.

## 5.3 Cataloging Multimedia Components

Multimedia components define the types of multimedia objects available for display on a Web page, such as an image of a certain size, short text description, or a ten-second audio file. They enable assignment of default and specific multimedia objects at the product, category, section, and store levels. Multimedia components are called by the store Web page templates to determine which multimedia appear on a given store page.

When you catalog a multimedia component, you choose a default multimedia object that is active at store level. In the Hierarchy tab, there are multimedia component fields where you can choose a multimedia object name to correspond with each component for each section.

In the Product and Category tabs, there are also multimedia component fields where you can choose a multimedia object name to correspond with each component for the product or category. Pages associated with the product or category will use this multimedia object instead of the store-level multimedia.

If no multimedia name is associated with a multimedia component for either product or category, then the multimedia object chosen for the product or category's parent section in the Hierarchy tab appears on pages associated with the product or category that use the multimedia component. If no specific multimedia name has been chosen for the section's multimedia component, then the store-level default multimedia object appears.

### Seeded Values

- STORE\_PRODUCT\_LARGE\_IMAGE
- STORE\_PRODUCT\_SMALL\_IMAGE
- STORE\_SECTION\_SMALL\_IMAGE

You can view the seeded values in the Multimedia Components tab. This page lists existing multimedia components and their programmatic access names, descriptions, default multimedia, and default source files.

Use this procedure to catalog multimedia components that you want to assign to sections, categories, or products.

## Prerequisites

Define the types of media objects you want to use on your store Web pages.

You can select a default multimedia object name for a multimedia component only after you have cataloged multimedia. If the default information is unavailable, you can continue the setup and select a default multimedia name at a later time.

However, if a multimedia association is requested for any product, category, or section with that multimedia component, and there is no product-specific, category-specific, or section-specific association for the multimedia component, Oracle iStore 11*i* uses the default multimedia object name defined at the store level.

To avoid the error, you can also use the multimedia component's seeded values, as listed in the Multimedia Components tab.

## Steps

1. Launch the Merchant UI.
2. In the Setup tab, choose **Multimedia Components**.

The Multimedia Components page displays a list of existing multimedia components and each component's default multimedia name. It also lists the default media source file for each multimedia name.

3. Click **Create**.

The Multimedia Component Details page appears.



**Figure 5–2 The Multimedia Component Details Page**

ORACLE  
Oracle Applications

Sign Out Help

Setup Category Hierarchy Relationship Product Templates Multimedia Cache

Specialty Stores Multimedia Components Display Styles

### Multimedia Component Details

\*Name

\*Programmatic Access Name

Description

Default Multimedia  Go

Update Restore

4. In the Name field, define the multimedia component name.
5. In the Programmatic Access Name field, define the programmatic access name. This name is called by the templates for the store Web pages that use this multimedia component.
6. Optional: In the description field, enter a description of the multimedia component. Entering a description enables a description-based search for this multimedia component.
7. In the Default Multimedia field, click **Go** to select a default multimedia name for this component. This default multimedia object will appear in pages associated with this multimedia component when the pages' products, categories, or sections have no specific multimedia assignments for this component.
8. Click **Update**.  
The multimedia component information is saved.

## 5.4 Customizing Templates

Oracle iStore 11*i* Web page designs use common components, such as section tabs and browse bins. Each component is based on a template, and the templates are combined to create a store Web page. The templates control the appearance of the store through the use of JavaServer Pages™ (JSP™), which combine Application Programming Interfaces (API) to call dynamic data and HTML to present static data.

Oracle iStore 11*i* comes packaged with a complete set of JSP templates needed to run the store. If you want to expand the functionality of the store Web pages or customize the pre-packaged templates, then you need to identify the flow of the application and the JSP templates needed to implement the flow. See *Oracle iStore Concepts and Procedures* for more information.

To customize templates for your store, perform the following tasks after planning your Web page designs:

- Create template source files for pages and for blocks within pages using a Web authoring application.
- Catalog Oracle iStore 11*i* template names and assign template source files to each template name.

### 5.4.1 Creating Template Source Files

You can create new JSP templates to replace or add to the Oracle iStore 11*i* seeded templates. Different physical JSP templates can be used at run-time based on the language and specialty store.

---

---

**Note:** It is recommended that you use Oracle JDeveloper to create and modify JSP templates. Although you can create JSPs with any HTML or text editor, Oracle JDeveloper also enables you to debug the code.

---

---

The major skills required to create and modify templates are HTML and JSP. JSP embeds Java language methods in the HTML content to generate dynamic content on the Web page. The structure of a JSP page is demonstrated in the following HTML example.

```
<HTML>
<% import="oracle.apps.ibe.util.*" %>
....
```

```
....
<P> Name : <% = customer.getName(12334) %>
    Where customer is a Java class on the server and getName is a public method
    in the class to retrieve the customer Name.

<P> Picture: <IMG SRC = "<% = customer.getPict(12334) %>">
    This step can retrieve the image file name from the customer Java class on the
    server.

....
</HTML>
```

The default UNIX directory for JSP source code is `$COMMON_TOP/html`. All `ibem*.jsp` templates are for the Merchant UI, and all `ibeC*.jsp` templates are for the Customer UI. New templates should also be placed in the `$COMMON_TOP/html` directory. Changes made to the JSPs may not appear immediately on the Web stores, since you must reboot the Apache server before changes take effect.

Deleting the server cache has the same effect as rebooting the Apache server. The server cache is located in the UNIX directory `$COMMON_TOP/html/_pages/oa_html`. This cache directory contains `.java` and `.class` files that are generated after the JSP that has been called is translated. These can be safely deleted and will be regenerated when the JSP is invoked through an HTTP request.

After creating or modifying templates, you can pre-compile them to check for compilation errors and to increase the speed of the initial loading.

---

---

**Note:** Sometimes it is not immediately obvious that templates referred to in the JSP code are in fact JSPs themselves. To find the JSP name of a template, search in the Template tab of the Merchant UI for the template name referenced in the code. The JSP name is included in the template listing. This JSP can then be modified to suit the requirements of the project.

---

---

## JSP Naming Conventions

Modify JSP templates only after renaming them first. All modified JSPs should follow a standard naming convention, e.g., `name of project-name of jsp.jsp`

This will make future Oracle iStore 11i upgrades less problematic.

---

---

**Note:** Never change an original JSP. To modify a JSP, make a copy of the original JSP and modify only the copy. If a bug occurs, compare the JSP copy to the JSP original.

---

---

## Cascading Style Sheets (CSS)

Oracle iStore 11*i* uses the logical template name `JTF_STYLE_SHEET` in the JSP template files to call the Cascading Style Sheet (CSS) that determines fonts, sizes, colors, and other elements of look and feel. As with JSP templates, you can map different CSS source files to the same logical template name for different specialty stores and languages.

Oracle iStore 11*i* comes with a single CSS called `jtfcuss.css`, and the `JTF_STYLE_SHEET` is seeded with this CSS as the default source file for all sites and languages. The CSS resides in the `COMMON_TOP/html` directory and can be modified using an HTML editor. Place any other style sheets that you create in the same directory.

## Modifying the Seeded Template Source Files

The Oracle iStore 11*i* Customer UI page is sectioned into various information containers, also referred to as bins or place holders. These bins hold the content-specific information and display it logically on the page. You can modify the bins' text and layout to change the Customer UI.

**Changing the Text in Bins** The text in the bins (for example, Welcome Message Bin, Shopping Cart Bin, Section Tree Bin) comes from the Messages stored in Oracle Forms. To change the text in the bins:

1. Log in to Oracle Forms with the Application Developer responsibility.
2. Choose the Messages menu option, and search for `IBE% messages`.
3. Modify these messages to change the text in the bins.

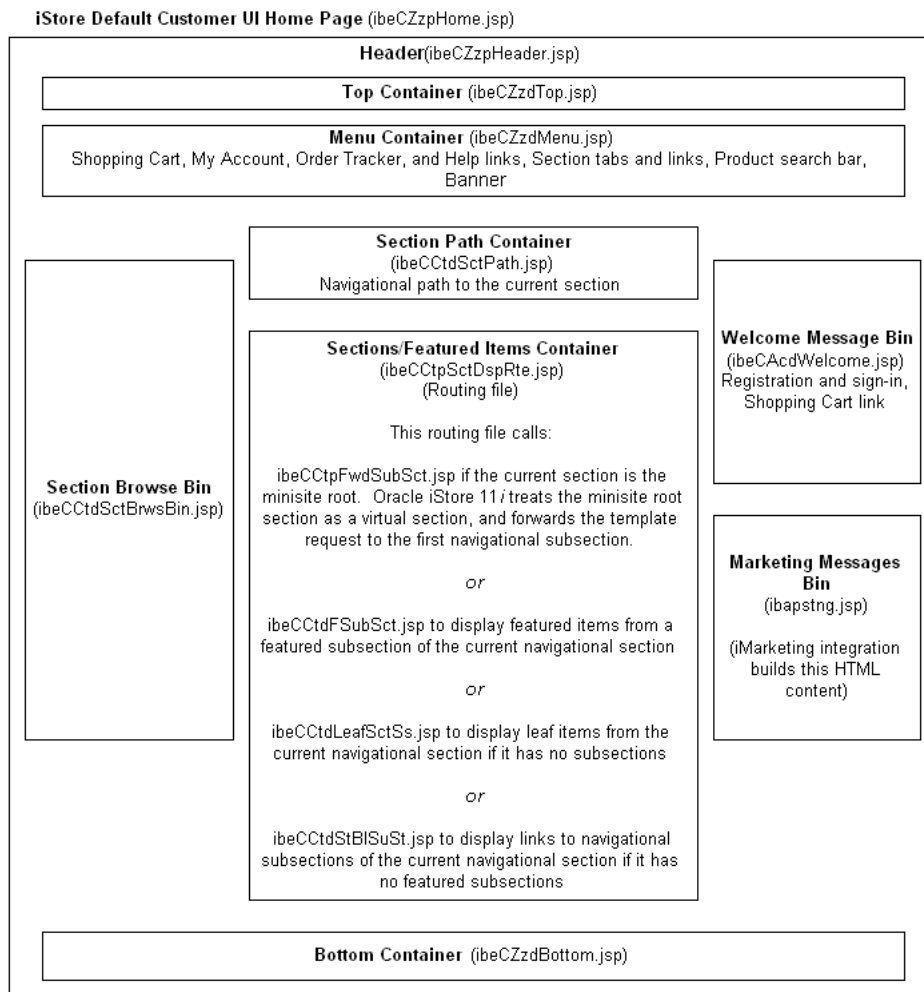
You can find a specific message name by viewing the respective bin JSP file.

**Changing the Layout of Bins** You can change the bin placement or remove a bin from the Customer UI by modifying the home page `ibeCZzpHome.jsp` and other corresponding JSP files. To verify that the home page is `ibeCZzpHome.jsp`, launch the Merchant UI, enter the Templates tab, and search the template catalog for Name = `STORE_HOME`. The default source file listed should be `ibeCZzpHome.jsp`.

**Note:** Be careful when making these changes, which affect the HTML.

The following diagram shows the layout of the bins on the default store page.

**Figure 5–3 Bin Layout on the Default Store Page**



## API Documentation

To make advanced changes to the Customer UI page displays, beyond bin layout and text messages, you must have complete knowledge of the APIs being called from the JSP template source file. The APIs are the key for displaying data on the store pages. These are the application objects and beans. Customers and users cannot modify these class files.

For public class API documentation, see [Chapter 8](#), [Chapter 9](#), and [Chapter 10](#).

## 5.4.2 Cataloging Templates

Cataloging templates involves setting up Oracle iStore 11*i* template objects with names and descriptions and specifying the different physical JSP templates to be used at run-time based on language and specialty store. These template objects can be additions to, or replacements for, the Oracle iStore 11*i* seeded template objects.

### Naming Templates

The template name is the catalog name that is easy to communicate and use when planning your page designs. An example is *ProductHome*. Template names may be translated for convenience in store administration.

Every template name also has a programmatic access name that is short, unique, and not as descriptive. The programmatic access name is inserted into your Web page or template. An example is *phome*. Programmatic access names are not translated.

The template name and programmatic access name represent several physical template source files. Each physical file can be assigned to combinations of specialty stores and languages. When Oracle iStore 11*i* retrieves an assigned template name, the template source file is determined by the mapping of the template name to the current specialty store and language.

The Display Manager is the class that implements Oracle iStore 11*i*'s Template Manager. The Template Manager maintains a mapping from a logical name or access name of a media object to a physical name on the file system. For example, STORE\_HOME (logical) maps to ibeCZzpHome.jsp (physical). When the Web store is active, the Display Manager determines what physical file to call from the logical template or multimedia component name, based on the specialty store and language.

The following table shows examples of file names for ProductHome.

**Table 5–2 Sample JSP File Names for the Template Name ProductHome**

Template Name	Programmatic Access Name	File	Specialty Store	Language
ProductHome	phome	hom1f.jsp	specialty store 1	French
ProductHome	phome	hom1e.jsp	specialty store 1	English
ProductHome	phome	hom2f.jsp	specialty store 2	French
ProductHome	phome	hom2e.jsp	specialty store 2	English

In this example, if a French customer enters specialty store 1, the store displays the home page file `hom1f.jsp`. An English customer in the same specialty store 1 sees `hom1e.jsp` instead.

### Assigning Templates to Presentation Levels

Templates can also be assigned to products, categories, and sections. You can specify these assignments through the Display Style options available in the Product, Category, and Hierarchy tabs, after setting up the Display Styles catalog. See [Section 5.5, "Cataloging Display Styles"](#) for details.

You can indicate that the template associated with a given display style will be used when displaying a product. You can also indicate at the section level the display style to use for displaying products that belong to that section. Oracle iStore 11i uses the following process to determine which template to use when displaying a product according to a given display style.

1. For a given display style, Oracle iStore 11i uses the template that you associated with the product.
2. If no template is associated at the product level, Oracle iStore 11i retrieves the template associated with the product's primary display category.
3. If no template is associated with the product or category, Oracle iStore 11i retrieves the default template for the display style.

### Cataloging Templates in the Template Manager

You can catalog templates using the Template Manager functionality, accessible through the Merchant UI Templates tab. Use this procedure to create template object names and programmatic access names, select default store-level template source files for them, and assign other template source files to them according to specialty store and language settings.

### Prerequisites

- At least one specialty store must have already been created.
- At least one language must have already been defined.

### Steps

1. Launch the Merchant UI.
2. In the Templates tab, search for templates that are already cataloged and available for use in your store.

The Templates page lists the names of the templates that match your search criteria, with their programmatic access names, keywords, descriptions, display level, and the default source files to use for all specialty stores and languages.

3. Click **Create**. To modify a template listing, click the template name in the page instead.

The Template Details Information and Source Files page appears.

4. In the Name field, enter the name by which the template is referred to during the planning stage, i.e., the common name.
5. In the Programmatic Access Name field, enter the name by which the template is referred to in the JSP.
6. In the Default Source File For All Sites and Languages field, enter the JSP to be used as the default if a non-default language or specialty store mapping is not defined.
7. In the Displays field, specify from the pull-down menu whether the template will be used to display a section (**section**), or a product or category (**category**). If the template will be used on Web pages that do not display the product catalog, choose **others** from the pull-down menu.
8. Optional: In the Keywords field, enter keywords for the template. Entering keywords enables a keyword-based search for this template when assigning a template to a display style.
9. Optional: In the Description field, enter a description for the template. Entering a description enables a description-based search for this template when assigning a template to a display style.
10. Click **Update**. An updated Template Details Information and Source Files page appears with a Source File for Other Sites and Languages section.



Figure 5–4 The Template Details Information and Source Files Page

ORACLE  
Oracle iStore

Sign Out Help

Setup Category Hierarchy Relationship Product Templates Multimedia Cache

## Template Details

[Information and Source Files](#) [View All Mappings](#) [Categories](#)

---

\*Name  Keywords

\*Programmatic Access Name  Displays category

Default Source File For All Sites and Languages  Description

[Source File for Other Sites and Languages](#)

Remove	Source File
<input type="checkbox"/>	sntemp.jsp

Add Source File 1-1 of 1

11. Optional: Click **Add Source File** to choose files for the same template in different languages and specialty stores. See the Guidelines below for details.

The Source File Details page appears.

- a. Enter the name of a physical JSP source file that you want to use for the template name you are creating. Click **Update**.  
An updated Source File details page appears with a Specialty Store and Language Mappings section.
- b. Add each store specialty store and language combination where you want the physical file to be used for this template, using the Specialty Store drop-down list and the **Go** buttons next to the Languages fields. Click **Update**.

The relationship between the template name, source file, specialty stores, and languages is saved.

- c. To add another physical file, click on the template name link.

The Template Details Information and Source Files page with the Source File for Other Sites and Languages section appears.

Click **Add Source File** and repeat this step to add another physical file to this template.

- 12. Optional: In the Template Details Information and Source Files page, choose **View All Mappings**.

The View All Mappings page displays each physical file name and its relationship to specialty stores and languages. This step is highly recommended.

- 13. Optional: In the Templates tab, choose **Categories** if the template you created is meant to display product categories.

The Templates - Assigned Categories screen lists the categories to which the template has been assigned. Click a category name to view all templates that have been assigned to the category.

- 14. Optional: If the template you created is meant to display product categories, you can assign it to categories now in the Category tab. In the Category tab, follow this procedure to add a template:

- a. Click the name of the category to which the template is applicable.

The Templates Assigned page lists all template names and default source template files for the chosen category.

- b. In the Templates Assigned page, click **Go**.

A list of available template names appears.

- c. Select the template(s) you wish to assign to the category.

- d. Click **Add**. The pop-up window closes when you select **Done**.

You can edit template information by clicking the template name in the Templates tab.

## 5.5 Cataloging Display Styles

Display styles specify how to present products on a Web page. For example, one display style specifies how to display Product A on a special sale page containing multiple products, and a different display style specifies how to display Product A on a page detailing product information. A display style calls an Oracle iStore 11i template, which then calls the appropriate template source file for the specialty store and language.

When you catalog display styles in the Setup tab, you choose store-level default template names for them. The display styles appear in the Product and Category tabs with fields where you can choose a template name to correspond with each display style for a product or a category. The display style fields also appear in the Hierarchy tab, where you can choose a template name to correspond with each display style for a section.

When a store Web page displays a product using a particular display style, Oracle iStore 11i selects the appropriate template as follows:

- If there is a product-specific template for the given display style, then the product-specific template is used.
- If no mapping is specified at the product level, and there is a category-specific template, then the category-specific template is used.
- If no template name is selected for a product or a category, then the display style's default store-level template is used on the Web page.

Clicking **Display Styles** in the Setup tab lists seeded display styles and their programmatic access names, descriptions, default templates, and default source files.

### Seeded Values

- STORE\_FEATURED\_PRODUCT
- STORE\_PRODUCT\_DESCR
- STORE\_PRODUCT\_DETAIL
- STORE\_PRODUCT\_DETAILS
- STORE\_PRODUCT\_SMALL\_DESCR

Use the following procedure to create more display styles.

### Prerequisites

You can select a default template only after you have cataloged templates. If the information is unavailable, you may continue the setup and select a default template later. However, if a template association is requested for any product or section with that display style and is not specified, Oracle iStore 11*i* will use the default store-level template.

To avoid the error, you can also use the seeded values for display styles, listed under **Display Styles** in the Setup tab.

### Steps

1. Launch the Merchant UI.
2. In the Setup tab, choose **Display Styles**.

The Display Styles page displays a list of existing display styles.

3. Click **Create**.

The Display Style Details page appears.

4. Assign names and descriptions to the display style.
5. Optional: Click **Go** to select a default template for this display style.
6. Click **Update**.

The display style information is saved.

You can edit display style information by clicking the display style name in the Display Styles page in the Setup tab.

## 5.6 Modifying the Hierarchy

All the specialty stores you create will be based in the overall hierarchy you created when setting up your initial specialty store. They can be associated to a portion of the overall hierarchy or to the whole hierarchy itself, depending on the section to which their root section points.

When reorganizing store sections and products or adding new specialty stores, you need to modify the overall hierarchy.

Your template design determines how to manifest the hierarchy for the user. You can create and revise new templates at any time. In the template for a section with a Featured type subsection, you can highlight the products in that section. A section's products are shown using the display style that you specify for that section.

---

While modifying the display parameters of a section, you can choose to keep it unpublished until you have tested its appearance in the Customer UI, since an unpublished section is not available in the Customer UI unless the user has the IBE\_ADMINISTRATOR responsibility.

### Prerequisites

Profile set up for IBE:Item Validation Organization (see definition for IBE\_ITEM\_VALIDATION\_ORGANIZATION profile).

### Steps

1. Launch the Merchant UI and enter the Hierarchy tab.
2. In the left frame of the Hierarchy tab window, the overall hierarchy tree appears. In the tree, select the node that will host the node you want to create, and click **Create**. (To modify existing information, select the node itself.)

The Basic Information Screen appears in the right frame of the Hierarchy tab window.

3. Enter the details.

Optional: Specify a Section Code by which you can refer to the section in customized templates explicitly by name. The Section Code is the name used in the template to access the section information directly.

Optional: Under the Display Parameter heading, specify a section-level default template for displaying the section from the pull-down menu labeled, "Template for displaying this section."

---

---

**Note:** Do not choose a template name that ends with the word "included." These templates display only the center of the section page, not the menu.

---

---

Optional: Under the Display Parameter heading, specify a display style for products in this section from the pull-down menu labeled, "Display Style for products in this section."

---

---

**Note:** If the section is a leaf section, do not choose the display style Product Detail Style.

---

---

Click **Continue**. The Multimedia Components screen appears.

4. Optional: Provide information about multimedia components specific for this section. Use this if you want to show or associate multimedia content with sections. For a given component, click **Go** to search the multimedia catalog. If the desired object is not found, create one and associate it with the section. Click **Continue**.

The Specialty Stores screen appears.

5. Move the specialty store(s) in which this section should appear into the included Specialty Stores column.

Decide if the section should appear only in the specialty stores that you have selected here or in other specialty stores by default if those specialty stores' roots point to an ancestor of the current section. Check the box **Include in all future sites if the site's root section is ancestor of this section** as appropriate. Click **Continue**.

The Relationships screen appears with a list of existing relationships in the section.

6. Review relationship rules for this section. See [Section 5.7, "Creating Relationships"](#) for information on customizing relationship rules. Click **Continue**.

The Advanced Settings screen appears.

7. Optional: Specify if the section is going to be populated automatically with products from Oracle Inventory based on certain SQL clauses, as well as specify the order.

---

---

**Note:** The auto-placement rule is not currently used.

---

---

Optional: Specify Order By clause to specify how the product for a section should be ordered when displayed in the Customer UI. The value for this field can be just one column name from MTL\_SYSTEM\_ITEMS or comma-separated columns of the same table.

Click **Finish**. The Children page appears.

8. Add products or subsections to this section as desired. Note that a section can have either a subsection or products as children, but not both. Featured sections cannot have subsections as children.

---

You can add products to a section in the Hierarchy tab after building the product catalog. See [Section 4.5, "Setting Up the Product Catalog"](#) for more information. To add products to a section in the Hierarchy tab:

- a. Navigate to the section's Children page.
- b. Click **Add Product**.
- c. Perform a product search in the pop-up window.
- d. Select one or more of the results and click **Add**.
- e. Click **Done** to close the pop-up window and return to the main store manager window.

You can also add products to a section when working on the product in the Product tab.

To add a subsection to a section:

- a. Navigate to the section's Children page.
- b. Click **Add Section**. A Basic Information screen for the new subsection appears.

---

---

**Note:** The **Add Section** button appears only if the section has no products.

---

---

- c. Set up the subsection in the Basic Information, Multimedia Components, Specialty Stores, Relationships, and Children screens in the Hierarchy tab.
9. To adjust the ordering of the section tabs in a minisite, click its root section in the navigation tree and edit the values in the Order column. The section whose display name should be the first tab should have the lowest number in the Order column.
  10. Continue to create sections and subsections for your hierarchy as needed. You can use the **Cut**, **Copy**, and **Paste** buttons to move an entire section from one place in the hierarchy to another.

To cut or copy a section, highlight the section and click **Cut** or **Copy** as appropriate.

To paste a section into the hierarchy after cutting or copying it, highlight the root section that should host the pasted section, and click **Paste**.

---

---

**Note:** If you cut and paste a section from one location in the hierarchy to another, all the information about the section will transfer to the new location. If you copy and paste a section, all the information about the original section will copy into the new section, except for the section-level multimedia component settings.

---

---

### Guidelines

- The display style you choose for products in this section is the style used to display products on a section page for this section.
- Choosing **Include in all future specialty stores if the store's root section is ancestor of this section** in the Specialty Stores screen ties the section to its ancestors. When a specialty store is added to or deleted from the ancestor, the same change applies to all its descendant sections.

## 5.7 Creating Relationships

Relationships are used for merchandising, for example, to offer a substitute product for a product that is out of stock. Relationship types are used to create specific relationship rules that associate products, categories, and sections to other products, categories, and sections. One relationship type can contain either rules created using the rule builder or one SQL rule. It cannot contain both.

---

---

**Note:** Using SQL rules to define relationships by querying the database on particular fields is a method primarily used by Oracle Consulting or other highly technical personnel. Most store managers will use the mapping rules.

---

---

The mapping rules define relationships in a From-To form. The types of From and To objects can be categories (defined in Oracle Inventory), sections or hierarchies (defined in Oracle Oracle iStore 11i), or items (defined in Oracle Inventory). The application evaluates each mapping rule and inserts rows in a table maintaining the preevaluated relationships. For example, if you have a category with two products assigned in your From list and a section with four products assigned in your To list, then Oracle Oracle iStore 11i creates a total of eight product relationships.

Your business needs determine the creation of relationships. Oracle iStore 11i ships with several seeded relationship types, listed in [Section 4.5.1, "Using Seeded Relationship Types"](#) and also in the Merchant UI Relationship tab. These seeded



relationship types are also seeded in Oracle Inventory for Item Relationships. If you use Oracle iStore 11i's Java API to retrieve related items given an item ID and a seeded relationship type, you will get related items defined in Oracle iStore 11i plus the ones defined in Inventory.

Use this procedure to define relationship types and add rules to them.

## Prerequisites

Products must exist in Oracle Inventory.

## Steps

1. Launch the Merchant UI.

2. In the Relationship tab, choose **Create**.

The Create Relationship page appears.

3. In the Name field, enter the relationship type name.

Optional: In the Description field, enter a description. This enables a description-based search for the relationship type.

Optional: In the Start Date and End Date fields, enter a start date and an end date for the relationship type to be valid.

Click **Create**.

The relationship type has been created. The Relationship Detail page appears, where you can begin adding rules to the relationship type.

4. Choose to specify the pairs of related items by SQL query or by mapping rules and highlight the appropriate radio button. Click **Create Rule**.

The Add Rules page appears if you choose **Create Mapping Rules**.

The Relationship Detail page appears if you choose **Create a SQL Rule**.

5. If you choose **Create Mapping Rules**, proceed as follows:

- a. Conduct a search to view products, categories, or sections in the center table.

The search results appear in the table.

- b. Select the items in your search results that you want to be in the From side of your rule, and click the left arrow.

The selected items appear in the From List.

- c. Conduct a search to view products, categories, or sections in the center table.

The search results appear in the table.

- d. Select the items in your search results that you want to be in the To side of your rule, and click the right arrow.

The selected items appear in the To List.

- e. Repeat as needed to complete your From and To lists for this rule.

- f. Click **Done** to submit the relationship rule creation, or click **Preview Rules** to validate or exclude the relationship rules to be added.

If you click **Done**, the Relationship Detail page appears. The application generates a rule from every object in the From list to every object in the To list.

If you click **Preview Rules**, the Preview Rules Page appears. At this point the rules have not been added to the system. You may exclude any rules not needed. When finished, click **Done** to see the Relationship Detail page.

- g. From the Relationship Detail page, you may select a link for each rule to view all product to product relationships generated by that rule, or click **View All Results** to view all product to product relationships generated by all rules in this relationship type.

From either option, the Relationship Result screen displays the product-level relationship results.

If you do not want to include one or more of the generated rules, select **Exclude** and click **Update**.

The excluded product-level relationships can be re-included.

- 6. If you choose **Create a SQL Rule**, the following incomplete SQL displays on the Relationship Detail page:

```
Select msi.inventory_item_id
From mtl_system_items msi
Where
```

The SQL should return only the column `inventory_item_id` in the `MTL_SYSTEM_ITEMS` table. You can add as many tables as you want in the From list and add any conditions in the Where clause.

## 5.8 Customizing Product Presentation at the Category Level

Every product is assigned or mapped to a category in Oracle Inventory. With customized multimedia, templates, and display styles, you can set up customized category-level defaults for products belonging to a category.

Use this procedure to modify defaults for categories. You can only specify defaults for categories belonging to the primary display category set (the value of the IBE: Category Set profile). If the product does not belong to any category in the primary display category set, then the store-level defaults are used.

### Prerequisites

- Products must be assigned to categories in Oracle Inventory in order to be returned upon a search of those categories.
- Multimedia, templates, and styles must exist before you can assign them to a category.
- The IBE: Category Set profile must be set. See [Chapter 7](#) for details.

### Steps

1. In the Category tab, search for categories by Category Name or Description.  
The Categories page displays a list of item categories from Oracle Inventory which belong to the category set specified in the IBE: Category Set profile with related templates, display styles, and multimedia components.
2. Click the category name that you want to update.  
The Templates Assigned page lists all template names and default source template files available for Web pages that display items in the chosen category.
3. In the Templates Assigned page that appears, perform the following steps to make a template available for association with a display style in this category:
  - a. Click **Add**.  
A list of available template names appears.
  - b. Select a template.
  - c. Click **Add**. The pop-up box closes when you select **Done**.
4. In the Category tab, choose **Display Styles**.  
The Display Styles page lists all display styles you defined in the Setup tab.

5. For each display style you can choose a template from the templates assigned to the category. Select **Update**.

In store Web pages related to this category that use a given display style, this template overrides the display style default template.

6. In the Category tab, choose **Multimedia Components**.

The Multimedia Components page lists all multimedia components that you defined in the Setup tab.

7. Optional: Assign multimedia names to multimedia components. Click **Update**.

In store Web pages related to this category, when a template accesses a multimedia component, the multimedia name selected here overrides the multimedia component's default multimedia setting. The multimedia name retrieves the media source file mapped to the specialty store and language used by the customer.

## 5.9 Customizing Product Presentation at the Item Level

Item-level customizations override category-level, section-level, and store-level settings. You can also customize product presentation at the item level in the following ways:

- Modify the product catalog.
- Create and associate specific images with certain products.

Defining proprietary multimedia and multimedia components enables association of specific images with certain products.

- Add item descriptive flexfields.

### 5.9.1 Modifying the Product Catalog

Customizing the product catalog involves the following considerations beyond those listed in [Section 4.5, "Setting Up the Product Catalog"](#) of this manual.

- Designing the display appearance for different product types. This process determines the number and type of product templates required. For example, perhaps all music products list the artist first and then provide a link to an audio clip, but all clothing products list the clothing type (e.g., jacket) first, followed by a graphic of the item. Oracle iStore 11i ships with the assumption that all product types appear the same on the Customer UI.

- Creating template text for product types. Text embedded in a template makes that template specific to the given product type. For example, the word “Artist” in front of the flexfield where a performer’s name is to appear can only be used for the compact disc product category. Embedded text must be manually translated and saved in the required multiple languages as additional template types. Oracle iStore 11i does not translate template text. Alternatively, templates using generic terminology can be more easily applied across product types. For example, using the term “Lead Performer(s)” as a flexfield label could apply to both compact disc and videotape product categories. Providing no flexfield labels in a template allows templates to be most broadly applied across product types.

Use this procedure to add products to the product catalog, make them available for sale in your store, and customize their presentation at the item level. For example, defining proprietary multimedia and multimedia components enables association of specific images with certain products. Item-level customizations override category-level, section-level, and store-level settings.

While modifying the display parameters of a product, you can choose to keep it unpublished until you have tested its appearance in the Customer UI, since an unpublished product is not available in the Customer UI unless the user has the IBE\_ADMINISTRATOR responsibility.

### **Prerequisites**

- Products must be loaded into Oracle Inventory before they can be imported into Oracle iStore 11i.
- Products in Oracle Inventory must have their Web Status flag set to either Published or Unpublished in the Web Option tab of the Master Item form to appear in the Oracle iStore 11i Merchant UI. Only products with a status of Published can be sold in your store.
- Products in Oracle Inventory must be set with the flag Orderable on Web in the Web Option tab of the Master Item form.
- Products in Oracle Inventory must be set with the flag Customer Orders Enabled in the Order Management tab of the Inventory Master Item form.
- JTF and IBE profile options must be set.
- Shipping options must be seeded into Oracle Shipping.
- Payment options and setup must be seeded into Oracle iPayment.
- Store layout must have been determined with the following considerations:

- Site appearance must have been decided.
- Hierarchy of products, sections, and specialty stores must have been identified.
- Templates associated with each product, section, and specialty store must have been identified.
- Templates for full Web pages and areas within Web pages must have been identified.
- At the implementation level, the mapping between templates and source files must have been decided.
- Source files (physical templates) must have been created by the UI implementation team with stubs for the dynamic elements, along with the multimedia components to be displayed on the site.
- Templates must have been populated with the dynamic JSP elements calling Oracle iStore 11i, using the templates shipped with Oracle iStore 11i as a model.
- Display styles must exist before you can assign them to a product. See [Section 5.5, "Cataloging Display Styles"](#) for more information.

## Steps

1. Launch the Merchant UI and enter the Product tab.

The Products page appears.

2. Search for products you want to include in your catalog. A search results list appears.

If there are unpublished products in your results, the list shows them with a **Publish** button in the Wizard column. To publish an unpublished product, you can either click on the product name and change the Posting Status to **Published** in the Basic Information window that appears, or click the **Publish** button next to the item name.

---

---

**Note:** Publishing or unpublishing a product in the Oracle iStore 11i Merchant UI also changes the product's Web Status setting in Oracle Inventory.

---

---

If you click the **Publish** button:

- a. The Basic Information screen appears.  
Optional: Add or modify the basic and long descriptions.  
Click **Continue**. The Hierarchy Paths window appears.
  - b. Optional: Add or remove parent sections for the item.  
Click **Continue**. The Category and Display Styles window appears.
  - c. Optional: Change template assignments for one or more of the display styles.  
Click **Continue**. The Multimedia Components page appears.
  - d. Optional: Change multimedia assignments for one or more of the multimedia components.
  - e. Click **Publish**.
3. Click on a product's name in the search results list to edit its information.  
The Basic Information page displays the Posting Status, the inventory name and part number, and any descriptions you have already added to the product catalog.
  4. To make the product available to be sold in your store, set the Posting Status to **Published**.  
To remove the product from your store, set the Posting Status to **Unpublished**.  
Optional: Enter or modify the short and long descriptions.  
Click **Update**. The product is published or unpublished immediately. The descriptions are saved, and are also available for display in your store if the product is published.
- 
- Note:** Products with Posting Status **Published** are visible on the store. Be careful about changing information, since the changes go to the production system and are published immediately. It is recommended that you unpublish the product before making changes, and then republish the item when finished.
- 
5. Optional: Enter or modify the short and long descriptions and click **Update**.  
The descriptions are saved and are available to display in your store.
  6. In the Products tab, choose **Hierarchy Paths**.

The Hierarchy Paths page displays the hierarchy of sections that have been set up for the store.

7. Optional: Remove or add parent sections for the product, edit the date range when this product will be available in each section, and number the product's place in the section's product display order.

8. In the Products tab, choose **Category and Display Styles**.

The Category and Display Styles page displays the category to which the product belongs and lists all display styles and any template names already assigned to the product.

9. Choose a template name for each display style that you want to use for the product.

To use the category-level default template for a display style, highlight the radio button next to the default setting on the display style line.

To set an item-level template, highlight the radio button next to the field on the display style line and click **Go**. Select a template from the pop-up window that appears. Click **Update** when finished.

10. In the Products tab, choose **Multimedia Components**.

The Multimedia Components page lists all multimedia components you defined in the Setup tab.

11. Optional: Assign multimedia names to multimedia components and click **Update**.

See [Section 5.9.2, "Creating Images for Products"](#) for more information on associating specific images with a product.

12. In the Products tab, choose **Relationships**.

The Relationships page displays existing relationships between the product and other products or sections and the rules for those relationships, such as the product to show for an upsell or cross sell.

13. To add relationships, go to the Relationship tab. See [Section 5.7, "Creating Relationships"](#) for more information on customizing relationships.

14. In the Product tab, choose **Specialty Store**.

The Specialty Store page lists the specialty stores where the product will be displayed. By default the product appears in those specialty stores to which the product parent section belongs.



15. Select the specialty stores that should display the product and click **Update**.

## 5.9.2 Creating Images for Products

To associate specific images with certain products, use the following procedure.

### Steps

1. Create images for products. Either create a new image for the product (usually done by a graphic artist) or use an existing image.
2. Locate the image and verify that it exists in the directory `/OA_MEDIA/`, where all images for the store should reside.
3. Set up the multimedia in the Merchant UI.
  - a. Log in to the Merchant UI and enter the Multimedia tab.
  - b. Click **Create**.
  - c. Enter the details: Name, Programmatic Access Name, Keyword, Description, and Default Source File. For the Default Source File, enter the complete path for the image file, starting with `/OA_MEDIA/`.
  - d. Click **Update**.

This creates and saves your multimedia.

4. Set up the Multimedia Component in the Merchant UI to associate the image with the product.
  - a. Enter the Product tab.
  - b. Search for the product that you want to associate with the image.
  - c. Click the product name.
  - d. Click **Multimedia Components**.
  - e. Highlight the radio button for Item Small Image and/or Item Large Image.
  - f. Click **Go**. A popup window with all the multimedia names will open.
  - g. Select the multimedia set up for the image.
5. Reboot the Apache server.
6. Verify that the image is associated with the product in the store's Customer UI.

### 5.9.3 Adding Item Descriptive Flexfields

Oracle iStore 11i allows addition of descriptive flexfields to item detail pages. With this option, the item detail page will display the prompt and value of descriptive flexfield global segments if a value is defined for the item. Only global segments of the descriptive flexfield are supported.

---

---

**Note:** The descriptive flexfields appear only if the IBE: Use CABO UI profile is set to Yes. See [Chapter 7](#) for more information.

---

---

To set up descriptive flexfields on an item detail page, follow this procedure.

#### Steps

1. Log in to Oracle Forms.
2. Select the Application Developer responsibility.
3. Choose **Flexfield > Description > Segments** to open the Descriptive Flexfields window.
4. Choose **View > Find**, and query for the flexfield with Application = Oracle Inventory and Title = Items.
5. Set up flexfield segments for Global Data Elements. See *Oracle Applications Flexfields Guide* for additional details.
6. Switch to the Inventory responsibility.
7. To set up flexfield segments and values for items and their detail pages:
  - a. Navigate to the Inventory Item window and find the item for which flexfield values will be entered. Use the inventory organization that is set in the profile option IBE: Item Validation Organization.
  - b. Click on the rectangle enclosed within [ ] next to the Description field in the Inventory Item window.
  - c. A window appears with the flexfield segments set up in the previous steps.
  - d. Enter values for the flexfield segments you want to display on the item detail page.
8. Test that the descriptive flexfield segments appear in the item detail pages as desired, using the following procedure:
  - a. Reboot the Apache server to clear the cache after entering the data.

- b. In the Customer UI, navigate to the item detail page for an item with flexfield values entered.
- c. The flexfield segment prompts and values should appear on the item detail page.

## 5.10 Setting Up Product Searches

Oracle iStore 11i's product search feature allows you to enable your customers to search a store for products they want to buy.

The product search feature in Oracle iStore 11i is implemented using the interMedia text search utility of the Oracle8i database. The product information (description and long description) is first loaded in an Oracle iStore 11i table (IBE\_CT\_IMEDIA\_SEARCH) via the concurrent program iStore Search Insert. This step is generally performed after the merchant has loaded his inventory with products. Once the data is loaded, any change to product information is updated in the Oracle iStore 11i table IBE\_CT\_IMEDIA\_SEARCH through a database trigger call on the inventory table. This keeps product information current in the search table. Once the data is moved into the search table, the interMedia index is created to facilitate search capability of the keywords.

---

---

**Note:** You must ensure that both Oracle Inventory and Oracle interMedia are installed and configured properly before setting up store search. Refer to the Oracle interMedia documentation for details on how to set up and configure interMedia.

---

---

### Storing Information in the Search Table

The search table IBE\_CT\_IMEDIA\_SEARCH is a denormalized table of MTL\_SYSTEM\_ITEMS\_TL and MTL\_ITEM\_CATEGORIES.

The core text on which you search is stored in a Clob called INDEXED\_SEARCH. Currently it stores a concatenation of name and description of products. The table also stores inventory\_item\_id, organizationId, category\_id, category\_set\_id, and the Web status field from MTL\_SYSTEM\_ITEMS\_B table.

### Searchable Product Attributes

Searches are performed on the name and description of a product. They are stored as description and long description columns in MTL\_SYSTEM\_ITEMS\_TL table.

## Search Dependencies

The product search requires version 8.1.7 of the Oracle database with the interMedia option installed. It also requires the 11*i* version of the Oracle Inventory schema.

---



---

**Note:** For enhanced query performance, enable caching of large object data for the interMedia DR\$R table.

---



---

### 5.10.1 Setting Up Oracle Inventory for Product Search

First, set up your inventory, under a common master org ID. Points to remember while setting up your inventory include:

- Give products unique names.
- Do not leave category names (concatenated segments) blank or non-unique. They can be null or non-unique in the database, but show as blanks or multiple times in the Categories LOV in your customer home page.
- Make sure the products have the WEB\_STATUS flag in the MTL\_SYSTEM\_ITEMS table set to PUBLISHED. See [Section 2.3.5, "Setting Up Product Items in Oracle Inventory"](#) for instructions. You can also query this field by examining the web\_status column of the item.

Next, set the iStore profile options for search.

### 5.10.2 Setting Search Profile Options

Oracle iStore 11*i* search needs four iStore (IBE) profile options to be set. The following table lists these profile options with their descriptions.

**Table 5–3 Search Profile Options**

Profile Option	Description
Enable Fuzzy Search	If set to <b>Yes</b> , allows users to perform fuzzy product searches, so that they do not have to type in the exact spelling of their search criteria to retrieve results that match these criteria. If this profile option is not set, it defaults to <b>No</b> .
No of Results in Search	This option sets the maximum cap on the search results. For example, if the user searches for a very common keyword (not in the stop words list), then the search process will stop after the max cap set as per this profile option. If this profile option is not set, then the code default of 200 results is used.

**Table 5–3 Search Profile Options (Cont.)**

Profile Option	Description
Search Lines Per Page	This option sets the number of lines to be displayed per page. If this profile option is not set, the code default of 20 lines is used.
Use Category Search	This option determines whether the home page pull-down search menu will allow category or section level searching. <b>Yes</b> will cause the pull-down menu to list categories with publishable items, while <b>No</b> will cause it to list the minisite's top level sections. A null value will enable a basic search against all products in the current minisite.  You must set this profile option to <b>Yes</b> if the IBE: Use CABO UI profile option is set to <b>No</b> .

Decide to enable either category or section level searches from the Customer UI home page pull-down menu, then carry out the appropriate procedure to populate the search table with data and create the interMedia text index.

---



---

**Note:** Section-level and basic searches are not supported if the IBE: Use CABO UI profile is set to **No**. If you choose to set this profile to **No**, you must set the IBE: Use Category Search profile to **Yes** to enable category-level search. See [Chapter 7](#) for more details.

---



---

### 5.10.3 Populating the Oracle iStore 11i Search Table for Category Level Search

To enable category-level search on the Customer UI, the iStore Search Insert program only needs to be run once to populate the search table IBE\_CT\_IMEDIA\_SEARCH. After this one-time product data load, the table will get updated product information through a database trigger call on the inventory table.

#### Steps

1. Log in to Oracle Forms as SYSADMIN.
2. Choose the **iStore Concurrent Programs Responsibility**. (If you do not have this responsibility, use the System Administrator responsibility to grant it to yourself.)
3. In the pop-up window, choose **Single Request**, and click **OK**.
4. Click the LOV button in the Name field, and choose **iStore Search Insert**.
5. Click **Submit** to start the concurrent request. Note the request ID.

You can monitor the progress of your request by looking at the request log and output files in `$COMMON_TOP/admin/log/l<request ID>.log` and `$COMMON_TOP/admin/out/o<request ID>.out`, respectively.

You can also view the request status by selecting View Requests and searching by the request ID.

---

---

**Note:** You will only be able to search for products whose WEB\_STATUS is PUBLISHED.

---

---

This process can take a substantial amount of time, depending on the number of items you have. As an estimate, for about 300,000 items in inventory this program can take about 45 minutes to run.

The concurrent manager calls the iStore Search Insert program, which moves the product data from the inventory table to the Oracle iStore 11*i* search table IBE\_CT\_IMEDIA\_SEARCH. When this job is running, the search tables are purged and the product search does not work correctly on the store.

---

---

**Caution:** Since this batch job deletes data from the search table, the rollback segment should be large enough for the process to complete.

---

---

Once the request is complete, you can search for products based on name and description. The pull-down search menu on the store's home page lists categories with publishable items. If additional product attributes are to be added in the search, this SQL script needs to be modified to add the extra search column.

#### 5.10.4 Populating the Oracle iStore 11*i* Search Table for Section Level Search

To enable section level search on the Customer UI for the first time, run the iStore Search Insert program first, then run the iStore Section Search Refresh program to populate the search table IBE\_SECTION\_SEARCH.

Whenever you update the Oracle iStore 11*i* hierarchy, rerun only the iStore Section Search Refresh program to update the search table IBE\_SECTION\_SEARCH.

### Steps

1. Carry out the procedure outlined in [Section 5.10.3, "Populating the Oracle iStore 11i Search Table for Category Level Search"](#) to load data into the main search table IBE\_CT\_IMEDIA\_SEARCH.
2. In the **iStore Concurrent Programs Responsibility**, choose **Single Request** and click **OK**.
3. Click the LOV button in the Name field, and choose **iStore Section Search Refresh**.
4. Click **Submit** to start the concurrent request. Note the request ID.

The concurrent manager calls the iStore Section Search Refresh program, which populates the search table IBE\_SECTION\_SEARCH with product data. The product search is still available to customers while the iStore Section Search Refresh program is running.

Once the request is complete, the pull-down search menu on the store's home page lists the top level sections, not the product categories.

## 5.10.5 Changing Between Category Level Search and Section Level Search

To change the listings in the pull-down search menu from categories to sections or vice versa, rerun the iStore Search Insert concurrent program to ensure that product listings will not be duplicated.

### Prerequisites

Change the IBE: Use Category Search profile option to **Yes** if you are changing to the category-level search. Change the profile option to **No** if you are changing to the section-level search.

### Steps

1. Log in to Oracle Forms as SYSADMIN.
2. Choose the **iStore Concurrent Programs Responsibility**. (If you do not have this responsibility, use the System Administrator responsibility to grant it to yourself.)
3. In the pop-up window, choose **Single Request**, and click **OK**.
4. Click the LOV button in the Name field, and choose **iStore Search Insert**.
5. Click **Submit** to start the concurrent request. Note the request ID.

You can monitor the progress of your request by looking at the request log and output files in `$COMMON_TOP/admin/log/l<request ID>.log` and `$COMMON_TOP/admin/out/o<request ID>.out`, respectively.

You can also view the request status by selecting View Requests and searching by the request ID.

If you are changing the search from section level to category level, the pull-down search menu lists categories once the request is completed.

If you are changing the search from category level to section level, perform the following steps after the request is completed:

6. In the **iStore Concurrent Programs Responsibility**, choose **Single Request** and click **OK**.
7. Click the LOV button in the Name field, and choose **iStore Section Search Refresh**.
8. Click **Submit** to start the concurrent request. Note the request ID.

The pull-down search menu lists the top-level sections once the request is complete.

## 5.10.6 Enabling a Fuzzy Search

The fuzzy search functionality returns search results with product names that do not match the spelling of the users' search criteria exactly. For example, if a user enters "laptops" or "laptp," the search retrieves product names with the word "laptop."

The fuzzy search works only for the base language of your installation of Oracle applications. It does not produce results for other languages in a multilingual instance, due to the current version of Oracle interMedia.

You can enable the fuzzy search functionality by setting the profile option IBE: Enable Fuzzy Search to **Yes** and running the iStore Search Insert concurrent program, as well as the iStore Section Search Refresh concurrent program if you have enabled section-level searches. Every time you change the value of the IBE: Enable Fuzzy Search profile option, you must rerun the iStore search concurrent programs.



## 5.10.7 Creating Search Index Tables

To be able to run external procedures to create a search index table, please ensure that ENV5 is included in your SID\_DESC part of listener.ora as follows.

### Steps

1. Go to 8.1.7 ORACLE\_HOME:

```
cd /u02/visappl
ksh
. ./APPSORA.env
cd $ORACLE_HOME/./8.1.7/network/admin
```

This directory should contain listener.ora.

2. Verify that listener.ora contains the following:

```
(SID_DESC =
  (SID_NAME = PLSExtProc)
  (ORACLE_HOME = /u04/visora/8.1.7)
  (ENV5 = LD_LIBRARY_PATH=/u04/visora/8.1.7/ctx/lib)
  (PROGRAM = extproc)
)
```

3. Before creating the search index table, make sure that the Oracle interMedia server is up. Use the following command to check:

```
$ ps -ef | grep ctxsrv
```

If it is not running, start the Oracle interMedia server as follows:

4. Go to 8.1.7 ORACLE\_HOME:

```
cd /u02/visappl
ksh
. ./APPSORA.env
cd $ORACLE_HOME/./8.1.7
. ./VIS.env
```

This will set up 8.1.7 ORACLE\_HOME env.

5. Run the following command:

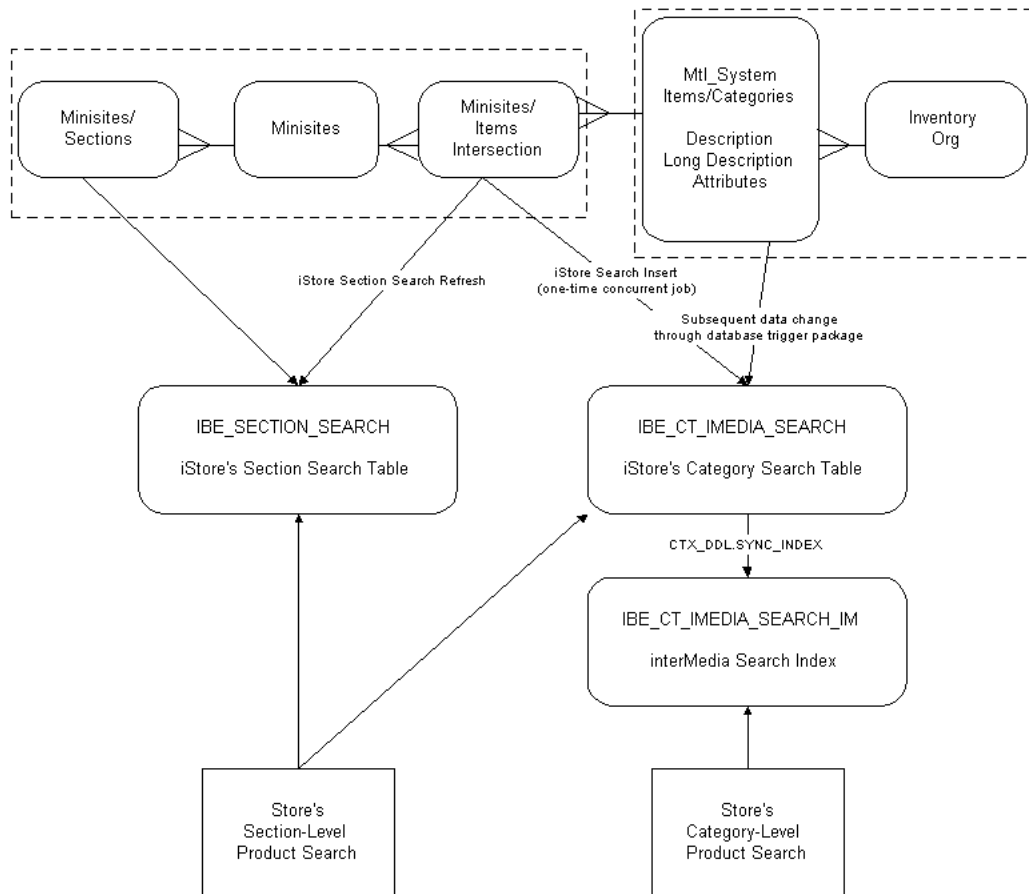
```
ctxsrv -user ctxsys/ctxsys&
```

### 5.10.8 PL/SQL, Java, and JSP's Involved in Search

The following program units are used in the product search process:

- java/catalog/Search.java (main Java program that executes the query)
- ibeckr3.jsp (search result JSP)
- ibeCZzdMenu.jsp (main home page)
- ibeckrf.jsp (search result JSP)
- IBEVCSMV.sql (one time load of product descriptions through concurrent manager)
- IBEVCSKS.pls (package specification)
- IBEVCSKB.pls (package body)
- IBEVIDTS.pls (package specification for the main database trigger)
- IBEVIDTB.pls (package body for the main database trigger)
- java/catalog/PrdRec.java (definition of search result object)
- IBEVCSIS.pls (package specification for section search package)
- IBEVCSIB.pls (package body for section search package)
- ibecsmcr.sql (maintains availability of product search functionality during iStore Section Search Refresh)

Figure 5-5 Oracle iStore 11i Search Tables



### 5.10.9 Customizing Search

If you need to add more attributes of the item to search for, you must modify IBEVCSMV.sql for the initial load and the PL/SQL triggers mentioned above to make sure that updates to these attributes get propagated to the search table.

1. Modify the search package (IBEVCSKS.pls and IBEVCSKB.pls) for adding the additional product search attributes. By default, only the product name (description column) and product description (long description column) are included in the search.

If additional attributes are to be added in the product search, the parameters for the package specification and body will have to be changed accordingly, with the new attributes. This package moves the subsequent changes in the product information, to Oracle iStore 11i's search table IBE\_CT\_IMEDIA\_SEARCH. Any insert, delete, or update on MTL\_ITEM\_CATEGORIES, any delete or update on MTL\_SYSTEM\_ITEMS\_B and any insert, delete, or update on MTL\_SYSTEM\_ITEMS\_TL table will move the change to the search table IBE\_CT\_IMEDIA\_SEARCH through this procedure. This procedure is called from the main database trigger procedures, as explained in the next step.

2. If new parameters are added to the search package, the call to the search package must be modified in the main database trigger package IBEVIDTB.plb to include the new parameters added to the search move procedures. This package body calls all of Oracle iStore 11i's ERP-related database trigger procedures, including the search package procedures.
3. The database trigger on the product tables calls the main database trigger package to move the product data change to the Oracle iStore 11i search table IBE\_CT\_IMEDIA\_SEARCH.

However, modifying the search package call will not recreate the interMedia index to include the changed information in the search table IBE\_CT\_IMEDIA\_SEARCH. Administrators must rebuild the interMedia search index every time a new product is added or an existing product is changed/deleted. This can be achieved by recreating the interMedia index IBE\_CT\_IMEDIA\_SEARCH\_IM through the Oracle Enterprise Manager utility or through executing the command "CTX\_DDL.SYNC\_INDEX" in SQL\*Plus. Note that you should have privileges to alter the interMedia index. After this step, the modified product information is visible in the Oracle iStore 11i product search process.

### 5.10.10 Adding Stopwords to Searches

There are many search words such as "and," "oracle," "if," and "then" which are very common and will return numerous search results. Search results may not be relevant to the user's query if such common search keywords are used. In addition, searches on common keywords use processing resources and slow down performance. These common keywords can be excluded from the search by using the "Stop Words" utility in interMedia.

Log in to Oracle Enterprise Manager as CTXSYS to see the stop words in the Stop List. Additional stop keywords can be added to the stop list.

### iStore Search Query

```

createQueryString(String keywords,
String operator,
String notKeywords,
String categoryId,
String maxRowNum,
String selectList,
String orderByList,
String whereList)

select i.inventory_item_id, i.description, i.category_id,
       score(100) nearness
from ibe_ct_imedia_search i, mtl_system_items_b b
where contains (i.indexed_search, 'laptop' , 100) > 0
and i.language = userenv('LANG')
and i.category_id = i.category_id
and i.organization_id = 204
and exists (
  select 1
  from jtf_dsp_section_items s, jtf_dsp_msite_sct_items b
  where s.section_item_id = b.section_item_id
  and b.mini_site_id = 10120
  and s.inventory_item_id = i.inventory_item_id
  and (s.end_date_active > sysdate or s.end_date_active is null)
  and s.start_date_active < sysdate
)
and rownum < 200
and i.inventory_item_id = b.inventory_item_id
and i.organization_id = b.organization_id
order by SCORE(100)
/

```

## 5.11 Customizing the Shopping Cart

You can customize shopping cart pages in the following ways:

- [Enabling Unit of Measure \(UOM\) Conversions](#)
- [Allowing Decimal Quantities for Items](#)
- [Specifying Flexfields At the Checkout Page](#)
- [Setting Up Credit Card Payments in Oracle iStore 11i](#)

### 5.11.1 Enabling Unit of Measure (UOM) Conversions

Oracle Pricing handles UOM conversions. The Pricing engine will only do UOM conversions if the Primary UOM Code checkbox is checked for the item in the Price List Setup window.

#### UOM Conversion Example

The primary UOM of Item X is Each and a conversion of 12 Each = 1 Dozen has been set up in Inventory. When pricing Item X, Oracle iStore 11i calls the Pricing engine passing in (Item X, Each) and (Item X, Dozen). The price list has a price for (Item X, Each).

- If the price list also contains (Item X, Dozen), the price for (Item X, Dozen) is returned.
- If the price list does not contain (Item X, Dozen) and the Primary UOM Code checkbox is checked for (Item X, Each), the price returned is 12 times the price of (Item X, Each).
- If the price list does not contain (Item X, Dozen) and the Primary UOM Code checkbox is not checked for (Item X, Dozen), an error is returned.

### 5.11.2 Allowing Decimal Quantities for Items

When adding an item or updating its quantity in the Oracle iStore 11i shopping cart, the customer can enter a decimal quantity if it is supported by the item. Oracle iStore 11i calls the same API used by Oracle Order Management for validating quantity. If an item is marked OM Indivisible, decimal quantities are not allowed for its primary UOM.

To prevent the customer from selecting a decimal quantity of an item, follow this procedure:

#### Steps

1. Log in to Oracle Forms as SYSADMIN.
2. Choose the Inventory responsibility for the Master Inventory Organization.
3. Choose Master Items.
4. Choose the appropriate inventory organization (the same as the Oracle iStore 11i Item Validation Organization).
5. Query for the item.

6. Click on the Physical Attributes tab.
7. Check **OM Indivisible**.
8. Repeat for all inventory organizations that contain the item.

To allow the customer to select a decimal quantity of an item, follow the previous steps, but uncheck **OM Indivisible**.

### 5.11.3 Specifying Flexfields At the Checkout Page

Oracle iStore 11i allows addition of flexfields to the checkout page and saves the information they contain to the quote. It passes the content of a flexfield on to Oracle Order Management as a comment in a non-validated field. For example, you can set up a flexfield for the sales representative that will go to Order Management as a comment that Order Administration will use to assign the correct sales representative ID.

Specify flexfields at the checkout page using the following procedure.

#### Steps

1. Log in to Oracle Forms.
2. Choose the Application Developer responsibility.
3. Choose **Flexfield > Description > Segments** to open the Descriptive Flexfields window.
4. Choose **View > Find**, and query for the flexfield with Application = Oracle Order Capture and Title = Header: Additional Information.

Set up flexfield segments. For example, map "Sales Rep Email" to "ATTRIBUTE1."

5. Query for the flexfield with Application = Oracle Order Management and Title = Additional Header Information.

Set up flexfield segments. For example, map "Sales Rep Email" to "ATTRIBUTE1."

Confirm the following:

- The same database columns are in use for both the Order Capture and Order Management flexfield segments (ATTRIBUTE1 in the previous example).

- The usage in Order Capture does not conflict with other flexfield definitions in Order Management.
  - Only global segments of a flexfield are supported.
6. Test your checkout page using the following steps:
    - a. Checkout a shopping cart and proceed to the "Payment And Billing Information" page.
    - b. You should see "Sales Rep Email" field under "Additional Information." You may enter information in this field, then continue.
    - c. After your order is created, go to Forms and check if the Sales Rep Email flexfield information is present in both Order Capture and Order Management Forms.
  7. Customize the prompts for the flexfields.

The default prompt title is, "Additional Information." To change it, log in to Forms and use `fnd_message: IBE_PRMT_ORD_FLEX_TITLE`.

The default additional instruction is, "Please fill in the following fields." To change it, log in to Forms and use `fnd_message: IBE_PRMT_ORD_FLEX_DESCR`.

#### 5.11.4 Setting Up Credit Card Payments in Oracle iStore 11i

Use the following procedure to set up credit card payment functionality in Oracle iStore 11i.

##### Steps

1. Log in to Oracle Forms as SYSADMIN.
2. Under System Administrator Responsibility, choose **(Navigation) Profile/System**.
3. Query Application = `iStore`, User = `ibe_customer`, and set the following profile values at the iStore application level:

- a. iStore Setup Profiles

Set the IBE: Authorize Payment Offline During Normal Checkout profile option to **No** to allow only online authorization.

Set the IBE: Finalize Order On Error in Authorize Payment profile option to **Yes** to submit orders even if the authorize payment error is a system error.



**b.** Order Capture Setup Profiles

Set the ASO: Credit Card Authorization profile option to **No** at the iStore application level only. (Do not set this profile at responsibility and user level).

**4.** Set up Oracle Receivables as follows:

Make sure Merchant ID is assigned to receipt method.

**a.** Log in to **Receivables Manager Responsibility > Setup > Receipt > Receipt classes.**

**b.** Query the receipt method with Name = Credit Card.

Make sure the Merchant ID field has the same value as the Payee ID of Oracle iPayment. (The Merchant ID is the ID that identifies your business through the Oracle iPayment server by Payment System, Credit Card Vendors and Banks. The Merchant ID that you provide in your receipt class is the same as the Payee ID that you have defined in the Oracle iPayment Administration).

**5.** Set up Oracle iPayment as follows:

**a.** Perform the manual post-installation configurations steps described in Chapter 2 of *Oracle iPayment Implementation Guide*. You should also be familiar with typical Oracle iPayment Administration operations, which are documented in *Oracle iPayment Concepts and Procedures*.

**b.** To integrate Oracle iPayment with Oracle iStore 11i, perform the following steps when you create a payee:

- Go to the Oracle iPayment UI Administration screen.
- Click the Payee Tab.
- Click the **Create** button.
- Fill in the form.

**Payee Identifier** should have the same value as the Merchant ID in Accounts Receivable, or else integration will fail.

- Click the **Credit Card** check box.
- Click the **Create** button at the end of the page to save the record.

Oracle iStore 11i uses customers' account ID numbers to store and track their credit card numbers in the AP\_bank\_Account\_uses\_All table, as per the TCA model and inline with the Oracle Receivables Forms UI.

## 5.12 Previewing Products and Sections

You can use the Preview feature of Oracle iStore 11*i* to preview the appearance of products and sections in the Customer UI before publishing them for your customers. Assign multimedia components, display styles, and categories to your products and sections, and set their statuses to Unpublished. Unpublished products and sections appear in the Customer UI only if the user has the IBE\_ADMINISTRATOR responsibility. Next, launch the Customer UI, log in to the appropriate specialty store using a store manager user account, and navigate to the items or sections. After viewing the items or sections, return to the Merchant UI. Here, you can make additional changes or publish the items or sections.

See [Section 4.6, "Testing the Store"](#) for more information about the Customer UI and testing your stores.

---

---

# Oracle iStore 11i Administration

This chapter describes the administration of Oracle iStore 11*i*. Topics include:

- [Overview of Store Administration](#)
- [Roles and Permissions for Oracle iStore 11i Users](#)
- [Setting Up Oracle iStore 11i Customer Types](#)
- [Setting Up B2B Users](#)
- [Managing the Cache](#)

---

---

**Note:** When using the Oracle iStore 11*i* Merchant UI, ensure that cookies are enabled.

---

---

## 6.1 Overview of Store Administration

The administration of your store involves the following tasks:

- Managing roles and permissions
- Setting up customer types
- Setting up B2B users
- Managing the cache

## 6.2 Roles and Permissions for Oracle iStore 11i Users

Oracle iStore 11i is seeded with various roles that you can assign to different types of users. Each of these roles has a different combination of permissions.

You cannot use roles and permissions for B2C customers.

You can view these permissions and roles using the following procedure.

### Steps

1. Log in to the Oracle CRM Applications login page at the following URL as SYSADMIN:

`http://<host>:<apache port>/OA_HTML/jtfllogin.jsp`

2. Enter the Security tab.
3. Choose the Permissions sub-tab.

The Permissions screen appears with a list of all Oracle CRM permissions. The Oracle iStore 11i permissions begin with "IBE."

4. Choose the Roles sub-tab.

The Roles screen appears with a list of all Oracle CRM roles. The Oracle iStore 11i roles begin with "IBE."

The following table summarizes the Oracle iStore 11i permissions.

**Table 6–1 Oracle iStore 11i Permissions**

Name	Description
IBE_ALLOW_PRICE_OVERRIDE	Allows the user to override prices manually
IBE_ASSIGN_SALES_CREDITS	Allows a user to assign sales credits

**Table 6–1 Oracle iStore 11i Permissions (Cont.)**

<b>Name</b>	<b>Description</b>
IBE_BILLTO_ANY_ACCOUNT	Allows a user to search on and retrieve all existing customers rather than only those with an existing billing relationship with the sold-to customer
IBE_CHANGE_BILLTO_CONTACT	Allows a user to change the bill-to contact from the default (if any) bill-to contact
IBE_CHANGE_BILLTO_CUSTOMER	Allows a user to change the bill-to customer from the default bill-to customer
IBE_CHANGE_SHIPTO_CONTACT	Allows a user to change the ship-to contact from the default (if any) ship-to contact
IBE_CHANGE_SHIPTO_CUSTOMER	Allows a user to change the ship-to customer from the default ship-to customer
IBE_CREATE_ADDRESS	Not used in Release 11i
IBE_CREATE_BILLTO_CONTACT	Allows the user to create a new contact for the bill-to customer who will have a bill-to relationship with the bill-to customer
IBE_CREATE_BILLTO_CONTACT_ADDRESS	Allows the user to create a new address associated with the bill-to contact which will have a bill-to relationship with the bill-to contact
IBE_CREATE_BILLTO_CUSTOMER	Allows a user to create a new customer with a billing relationship to the sold-to customer
IBE_CREATE_BILLTO_CUSTOMER_ADDRESS	Allows the user to create a new address associated with the bill-to customer which will have a bill-to relationship with the bill-to customer
IBE_CREATE_ORDER	Allows a user to submit a quote or cart as an order
IBE_CREATE_PAYMENT_INSTRUMENT	Not used in Release 11i
IBE_CREATE_SHIPTO_CONTACT	Allows the user to create a new contact for the ship-to customer who will have a ship-to relationship with the ship-to customer
IBE_CREATE_SHIPTO_CONTACT_ADDRESS	Allows the user to create a new address associated with the ship-to contact which will have a ship-to relationship with the ship-to contact

**Table 6–1 Oracle iStore 11i Permissions (Cont.)**

<b>Name</b>	<b>Description</b>
IBE_CREATE_SHIPTO_CUSTOMER	Allows a user to create a new customer with a shipping relationship to the sold-to customer
IBE_CREATE_SHIPTO_CUSTOMER_ADDRESS	Allows the user to create a new address associated with the ship-to customer which will have a ship-to relationship with the ship-to customer
IBE_CREATE_SOLDTO_CUSTOMER	Allows a user to create a new customer in the context of assigning a sold-to customer during quote creation
IBE_MODIFY_CART	Not used in Release 11i
IBE_MODIFY_ORDER	Not used in Release 11i
IBE_SHIPTO_ANY_ACCOUNT	Allows a user to search on and retrieve all existing customers rather than only those with an existing shipping relationship with the sold-to customer
IBE_USER_ADMIN	Allows a user to create additional users for his or her organization
IBE_USE_ATTACHMENT	Allows the user to use attachments
IBE_USE_PRICING_AGREEMENT	Allows the user to use pricing agreements
IBE_VIEW_ADDRESS	Not used in Release 11i
IBE_VIEW_CUST_WITHOUT_ACCOUNT	Allows a user to search on and retrieve existing customers without an account
IBE_VIEW_INVOICE	Allows a user to view invoices, related to the entire organization, through Order Tracker
IBE_VIEW_ORDER	Allows a user to view orders, placed on behalf of the entire organization, through Order Tracker
IBE_VIEW_PAYMENT	Allows a user to view payments, related to the entire organization, through Order Tracker
IBE_VIEW_PAYMENT_INSTRUMENT	Not used in Release 11i

The following table lists the seeded Oracle iStore 11i user roles and shows the permissions that are assigned by default to each role.

**Table 6–2 Oracle iStore 11i User Roles**

<b>Name</b>	<b>Description</b>	<b>Default Permissions</b>
IBE_BUSINESS_USER_ROLE	Business User Role	IBE_CREATE_ADDRESS IBE_CREATE_BILLTO_CONTACT_ADDRESS IBE_CREATE_ORDER IBE_CREATE_PAYMENT_INSTRUMENT IBE_CREATE_SHIPTO_CONTACT_ADDRESS IBE_MODIFY_CART IBE_MODIFY_ORDER IBE_VIEW_ADDRESS IBE_VIEW_INVOICE IBE_VIEW_ORDER IBE_VIEW_PAYMENT IBE_VIEW_PAYMENT_INSTRUMENT
IBE_PRIMARY_USER_ROLE	Primary User Role	IBE_CREATE_ADDRESS IBE_CREATE_BILLTO_CONTACT_ADDRESS IBE_CREATE_ORDER IBE_CREATE_PAYMENT_INSTRUMENT IBE_CREATE_SHIPTO_CONTACT_ADDRESS IBE_MODIFY_CART IBE_MODIFY_ORDER IBE_USER_ADMIN IBE_VIEW_ADDRESS IBE_VIEW_INVOICE IBE_VIEW_ORDER IBE_VIEW_PAYMENT IBE_VIEW_PAYMENT_INSTRUMENT

**Table 6–2 Oracle iStore 11i User Roles (Cont.)**

Name	Description	Default Permissions
IBE_RESELLER_ROLE	Reseller Role	IBE_CHANGE_SHIPTO_CONTACT IBE_CHANGE_SHIPTO_CUSTOMER IBE_CREATE_ADDRESS IBE_CREATE_BILLTO_CONTACT_ADDRESS IBE_CREATE_ORDER IBE_CREATE_PAYMENT_INSTRUMENT IBE_CREATE_SHIPTO_CONTACT IBE_CREATE_SHIPTO_CONTACT_ADDRESS IBE_CREATE_SHIPTO_CUSTOMER IBE_CREATE_SHIPTO_CUSTOMER_ADDRESS IBE_MODIFY_CART IBE_MODIFY_ORDER IBE_SHIPTO_ANY_ACCOUNT IBE_USE_ATTACHMENT IBE_USE_PRICING_AGREEMENT IBE_VIEW_ADDRESS IBE_VIEW_INVOICE IBE_VIEW_ORDER IBE_VIEW_PAYMENT IBE_VIEW_PAYMENT_INSTRUMENT

IBE\_BUSINESS\_USER\_ROLE and IBE\_PRIMARY\_USER\_ROLE are appropriate for assignment to B2B customer users.

---



---

**Note:** The B2B role for previous releases, IBE\_DEFAULT\_ROLE, is also seeded in Oracle iStore 11i with identical permissions to IBE\_BUSINESS\_USER\_ROLE, for backward compatibility.

---



---



IBE\_RESELLER\_ROLE has quote creation permissions, but does not allow the quote creator to view all customer accounts in your records, bill to anyone other than the sold-to customer, or sell to customers who are not in your records. It is appropriate for assignment to resellers and others who sell your products but are not internal to your organization.

See *Oracle HTML Quoting Implementation Guide* for more information about the Oracle HTML Quoting role IBE\_SALESREP\_ROLE, which also uses Oracle iStore 11i permissions.

## 6.3 Setting Up Oracle iStore 11i Customer Types

There are three basic Oracle iStore 11i customer types:

- Guest users
- Registered B2C users
- Registered B2B users

Guest users (also called walk-in or unregistered users) can browse Web store catalogs and create shopping carts. However, they cannot set up a user profile, save a shopping cart, create a shopping list, submit an order, or access other Web store functionality until they register. See [Chapter 7](#) for details about how Oracle iStore 11i treats a guest user, and instructions for defining the guest user account.

Registered B2C users are individual customers. When they register in a Web store, they are immediately approved and can place orders in the Web store on their own behalf.

Registered B2B users represent customer organizations. When they register in a Web store, they must be approved by you or another merchant representative before they can act as registered users in the Web store. You can give them different levels of permission, including a permission to create B2B users for their organizations who do not need merchant approval. See [Section 6.4, "Setting Up B2B Users"](#) for more information.

### 6.3.1 Setting Default Customer Roles

You can specify the role assigned by default to new B2B customers.

You cannot assign default roles to new B2C customers.

B2B customers who are created in a Web store by B2B administrative users for their organizations are automatically assigned the B2B default customer role set here.

#### Steps

1. Log in as SYSADMIN to the Oracle CRM Applications login page at:

`http://<host>:<apache port>/OA_HTML/jtfllogin.jsp`

2. In the Registration tab, click the Default Roles link.

The Groups screen appears.

3. To specify the default role assigned to new B2B customers, follow these steps:

- a. Select **Business User** from the User Type pull-down menu.
- b. In the Available Roles list, highlight the role that you want to assign by default to new B2B customers and click the ">" button.

In the Assigned Roles list, remove the role that you do not want to assign by default to new B2B customers by highlighting the role and clicking the "<" button.

Click **Update** to save your changes.

### 6.3.2 Setting Default Customer Responsibilities

You must specify the responsibility assigned by default to a new customer upon registration in a Web store. You can assign a default responsibility for both new B2B and B2C customers.

#### Steps

1. Log in as SYSADMIN to the Oracle CRM Applications login page at:

`http://<host>:<apache port>/OA_HTML/jtfllogin.jsp`

2. In the Registration tab, click the Default Responsibility link.

The Register Default Responsibility screen appears.

3. Select **Business User** or **End User** from the Account Type pull-down menu to specify whether you are setting a default responsibility for B2B or B2C customers, respectively.

The Register Default Responsibility screen refreshes with values for the specific Account Type in the pull-down menus.

4. Choose the customer's default responsibility in the Default Responsibility Id pull-down menu. This can be **IBE\_CUSTOMER** or another customer responsibility that you create.

See *Oracle Applications System Administrator's Guide, Release 11i* for information on creating additional responsibilities.

5. Click **Submit** to save the settings.

## 6.4 Setting Up B2B Users

B2B users are representatives of a customer organization. All purchases by B2B users will be assigned to their organizations' accounts.

When customers register as B2B users through the Web stores, you can assign them specific B2B roles that determine the permissions they have in the Web stores. You can use the seeded Oracle iStore 11i roles or create new B2B roles with various combinations of the Oracle iStore 11i B2B permissions.

### Seeded B2B Role Values

- IBE\_BUSINESS\_USER\_ROLE
- IBE\_PRIMARY\_USER\_ROLE

The IBE\_PRIMARY\_USER\_ROLE has permissions identical to those of IBE\_BUSINESS\_USER\_ROLE, with the addition of the IBE\_USER\_ADMIN permission. See [Section 6.2, "Roles and Permissions for Oracle iStore 11i Users"](#) for a list of these permissions.

If you give a B2B user a role with the IBE\_USER\_ADMIN permission, the B2B user can create more users for his or her organization in the Web store without the Oracle iStore 11i system administrator's approval. A B2B user with the IBE\_USER\_ADMIN permission can also assign B2B roles to the users he or she creates, and define new B2B user roles with unique sets of permissions.

When a B2B user logs in, links to the User Management and Role Management pages in the My Account screen are available.

Figure 6–1 The B2B User's My Account Page

The screenshot displays the Oracle iStore user interface. At the top, the Oracle iStore logo is on the left, and navigation icons for Shopping Cart, My Account, Order Tracker, and Help are on the right. Below the logo is a horizontal menu with Home, Books, Music, Electronics, and Computers. A secondary menu contains User Information, Express Checkout Preferences, User Management, and Role Management. A search bar is located below the menus, with a dropdown menu set to 'All Products' and a 'Go' button. On the left side, there is a sidebar with 'User Information' selected, containing links for Personal Information, Change Password, Address Book, Payment Book, and Preferences. The main content area is titled 'Personal Information' and contains a form with the following fields:
 

- \* First Name: B2B
- Middle Name: (empty)
- \* Last Name: User
- \* Email: b2buser@samplecorp.c
- In addition to Order Status Alerts, I would like to receive important information on exclusive Store specials, promotions and events.
- Daytime Phone Number: ( ) ( ) Ext. ( )
- Evening Phone Number: ( ) ( ) Ext. ( )
- Fax Number: ( ) ( ) Ext. ( )

 An 'Update' button is positioned at the bottom of the form.

The User Management page allows B2B administrative users to create B2B users for their organizations who do not need approval from the Oracle iStore 11i system administrator, and to specify roles for these users.

---

**Note:** A B2B administrative user can view all B2B users for his or her organization, even if he or she did not create them. However, the B2B administrative user sees role assignments only for the roles that he or she has. If a B2B administrative user views B2B users in the User Management screen who have roles that he or she does not have, these role assignments will not appear in the B2B users' role details. Instead, IBE\_DEFAULT\_ROLE is checked as the role assignment.

---

Figure 6–2 The B2B Administrative User's User Management Page

ORACLE  
iStore

Shopping Cart My Account Order Tracker Help

Home Books Music Electronics Computers

User Information Express Checkout Preferences **User Management** Role Management

Quick Search All Products  Go [Advanced Search](#)

Users

First Name	Middle Name	Last Name	Email	Username	Password
B2B		User	b2buser@samplecorp.com	A_B2B_USER	<input type="password"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="password"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="password"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="password"/>

[First](#) | [Previous](#) 1 - 1 of 1 [Next](#) | [Last](#)

The Role Management page allows B2B administrative users to view available roles, define new roles, and set permissions for existing roles.

---

**Note:** In the Role Management page, the B2B administrative users can see the Oracle iStore 11i seeded B2B roles that are assigned to them. Although there are Remove checkboxes for these seeded roles, the B2B administrative users should not delete these roles. Deleting the roles in the Role Management page will also delete them from all other Oracle iStore 11i B2B users' accounts.

---

**Figure 6–3 The B2B Administrative User's Role Management Page**

The screenshot shows the Oracle iStore administrative interface. At the top, there's a navigation bar with 'Role Management' highlighted. Below it is a search bar with 'All Products' selected. The main content area is titled 'Roles' and contains a table with the following data:

Remove	Name	Description
<input type="checkbox"/>	IBE_DEFAULT_ROLE	Default Role
<input type="checkbox"/>	IBE_NEW_USER_ROLE	New User Role
<input type="checkbox"/>	IBE_PRIMARY_USER_ROLE	Primary User Role
	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>
	<input type="text"/>	<input type="text"/>

At the bottom right of the table area, there are navigation links: 'First | Previous 1 - 3 of 3 Next | Last'. At the bottom left, there are 'Update' and 'Restore' buttons.

Although the User Management and Role Management links appear in the My Account screen for B2B non-administrative users, they receive a message saying they are not authorized to access these pages if they click on the links.

## 6.4.1 Creating B2B User Roles

Use the following procedure to create new B2B roles with any set of permissions that you wish to specify.

### Steps

1. Log in to the JTF login page at the following URL as SYSADMIN:  

```
http://<host>:<apache port>/OA_HTML/jtflogin.jsp
```
2. Enter the Security tab.
3. Click on the Roles link.

The Roles screen appears with a list of existing JTF roles, and rows of text fields where you can enter B2B user role names and descriptions.

Figure 6–4 The Merchant's Roles Screen

ORACLE  
Oracle Applications

Setup Diagnostics Tuning Profile Sign Out Help

Users Registration Security Payment Processing Advanced

Permissions Roles Data

Quick Find

### Roles

Remove	Name	Description	Data
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_CUSTOMER_READ_ONLY</a>	DMS Defect Customer read only	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_INTERNAL_READ_ONLY</a>	DMS Defect Internal read only	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_KB_SUBMIT</a>	DMS Defect KB Submit	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_MASS_UPDATE_ADMIN</a>	DMS Defect Mass Update Admin	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_MASS_UPDATE_REG</a>	DMS Defect Mass Update Regular	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_RESOLUTION</a>	DMS Defect Resolution Update	
<input type="checkbox"/>	<a href="#">CSS_DEF_DEFECT_USER</a>	DMS Typical Defect User	
<input type="checkbox"/>	<a href="#">CSS_DEF_ENH_CUSTOMER_READ_ONLY</a>	DMS Enhancement Customer read only	
	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>

[<First](#) [<<Previous](#) 1 - 10 of 35 [Next>>](#) [Last>](#)

- In the first text field in the Name column, enter a name for the B2B user role you are creating.

In the adjacent text field in the Description column, enter a description of the user role.

Do not select the checkbox in the Data column.

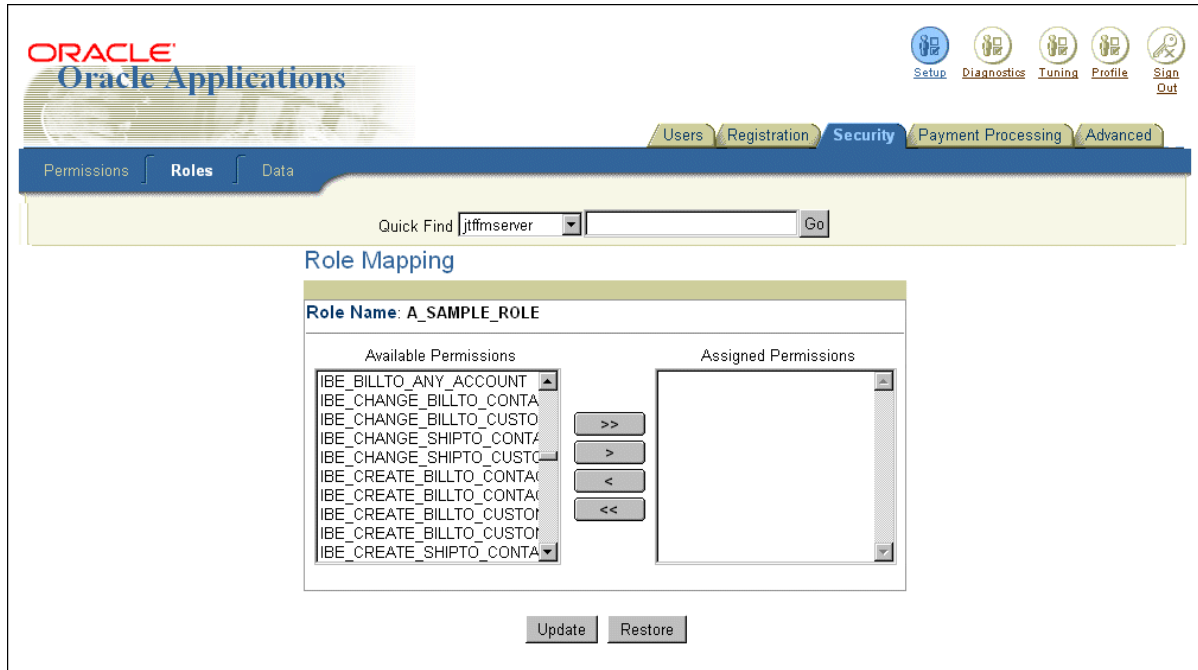
In the remaining text fields, enter names and descriptions for any other B2B user roles that you want to create.

Click **Update**.

The Roles screen appears again, with the role(s) that you have created.

5. Click the name of a role to specify its permissions.
6. The Role Mapping screen appears.

**Figure 6–5 The Role Mapping Screen**



7. To add permissions to the role, highlight the desired permissions in the Available Permissions list and click the ">" button to add them to the Assigned Permissions list.

To add all permissions to the role, click the ">>" button to add them to the Assigned Permissions list.

To remove permissions from the role, highlight the unwanted permissions in the Assigned Permissions list and click the "<" button to add them to the Available Permissions list.

To remove all permissions from the role, click the "<<" button to add them to the Available Permissions list.

Click **Update** to save the permission assignments.



You can now assign this role to any B2B users who register in your Web stores. If necessary, you can also assign this role or other customer roles to B2C users who register in your Web stores.

## 6.4.2 Approving B2B Users

When customers register with your Web stores as B2B users, you need to approve them before they can access B2B features and make purchases.

When you approve a B2B user for an organization that does not yet have an account, Oracle iStore 11i creates an account for the organization in the background.

If a customer directly requests a Web store user name for an organization account that already exists, instead of going through the organization's B2B administrative user(s), the Oracle iStore 11i system administrator needs to approve the user.

Use the following procedure to approve B2B users. You should check for new B2B user requests regularly.

### Steps

1. Log in to the JTF login page at the following URL as SYSADMIN:

```
http://<host>:<apache port>/OA_HTML/jtflogin.jsp
```

2. Enter the Registration tab.

The Pending Requests list shows the B2B users created in your Web stores, with the most recently created users at the end.

3. In the Pending Requests list, click the Username of the Business User that you want to approve.

The Request Details screen appears.

Figure 6–6 The Request Details Screen

**ORACLE**  
Oracle Applications

Setup Diagnostic Tuning Profile Sign Out Help

Users Registration Security Payment Processing Advanced

Approval Default Roles Default Responsibility

Quick Find jiffrserver Go

### Request Details

**User Details**

First Name: B2B Area Code:

Last Name: User Phone Number:

Username: A\_B2B\_USER E-mail: b2buser@samplecorp.c

Registration Date: 20-MAR-2001

Update Restore Assign Accounts

**Company Details**

Name: Sample Corporation Address: 12345 Main St.

City: Anytown Country: US

State: CA Phone:

Registration Date: March 20, 2001 Fax:

**Primary Contacts**

Name	Username	Phone	E-mail
B2B User	A_B2B_USER	-	<a href="mailto:b2buser@samplecorp.com">b2buser@samplecorp.com</a>

Accept User Reject User Return to Summary

- Optional: In the Request Details screen, modify the information in the text fields as necessary.

Click **Update** to save these changes or **Restore** to revert to the last saved version of the data.

5. In the Request Details screen, click **Assign Accounts** to associate an account with the B2B customer.

The Associated Accounts screen appears with a list of the accounts available for this organization.

Select the checkbox in the Attach column next to any accounts that you want to associate with the B2B customer, and click **Update**.

The Request Details screen appears.

6. In the Request Details screen, do one of the following:
  - Click **Accept User** to approve the B2B customer application.
  - Click **Reject User** to reject the B2B customer application.
  - Click **Return to Summary** to view the Pending Requests list again.

Continue with this procedure only if you click **Accept User**.

7. Optional: Enter comments in the text field screen that appears when you click **Accept User** in the Request Details screen.

Click **Submit**.

The B2B customer is approved.

You should now set up the B2B customer's data using the procedure outlined in [Section 6.4.3, "Modifying B2B User Data"](#).

### 6.4.3 Modifying B2B User Data

Use this procedure to modify data for a B2B customer. If necessary, you can also use it to modify data for a B2C customer.

#### Steps

1. Log in to the JTF login page at the following URL as SYSADMIN:  
`http://<host>:<apache port>/OA_HTML/jtflogin.jsp`
2. In the Users tab, search for the customer that you want to modify as follows:
  - a. From the pull-down menu next to Find Users, select **User Name, Last Name, or First Name** as your search criterion.
  - b. Enter your search criterion in the adjacent text field. Use % as a wild-card character if necessary.

**c. Click Go.**

The Users screen reappears with the search results.

- 3. Optional:** In the Users screen, you can perform the following tasks in addition to the user data modifications outlined in the rest of this procedure:
  - To sort by Last Name, First Name, or User Name, click the corresponding heading link at the top of the list of users.
  - To delete a user, select the checkbox in the Select column and click **Delete Users**.
- 4.** In the Users screen, click the User Name for a customer to modify his or her user data.

The User Details screen appears.

Figure 6–7 The User Details Screen

ORACLE  
Oracle Applications

Setup Diagnostics Tuning Profile Sign Out

Users Registration Security Payment Processing Advanced

Assign Roles

Quick Find jtfmserver  Go

### User Details

Party ID: 6464 User ID: A\_B2B\_USER

\*Last Name  \*Email Address

\*First Name

#### Business Details

Company Name: SAMPLE CORPORATION \*Address

\*City  Address Line 2

\*State  Address Line 3

\*Postal Code  Country: AMERICA

\*Phone  \*Fax

\*Indicates required field.

#### Reset Password

New Password

Password should be at least 6 characters long.

5. In the User Details screen, follow these steps to modify the customer name or address:
  - a. Enter the necessary changes in the Last Name, First Name, and Email Address text fields.
  - b. Click **Update** to save the changes.

If you make changes that you do not want to save, click **Restore** instead to revert to the last saved version of the user data.

When you click **Update**, the Acknowledgment screen appears with the newly saved data.

- c. In the Acknowledgment screen, either click **Edit** to continue editing user data in the User Details screen, or click **User Roles** to modify user role assignments for the customer in the User Role Mapping screen.
- 6. In the User Details screen, follow these steps to modify user role assignments for the customer:

- a. Click **Roles**.

The User Role Mapping screen appears. (This screen also appears if you click **User Roles** in the Acknowledgment screen.)

- b. To assign roles to the user, highlight the desired roles in the Available Roles list and click the ">" button to add them to the Assigned Roles list.

To assign all roles to the user, click the ">>" button to add them to the Assigned Roles list.

To remove roles from the user, highlight the unwanted roles in the Assigned Roles list and click the "<" button to add them to the Available Roles list.

To remove all roles from the user, click the "<<" button to add them to the Available Roles list.

---

---

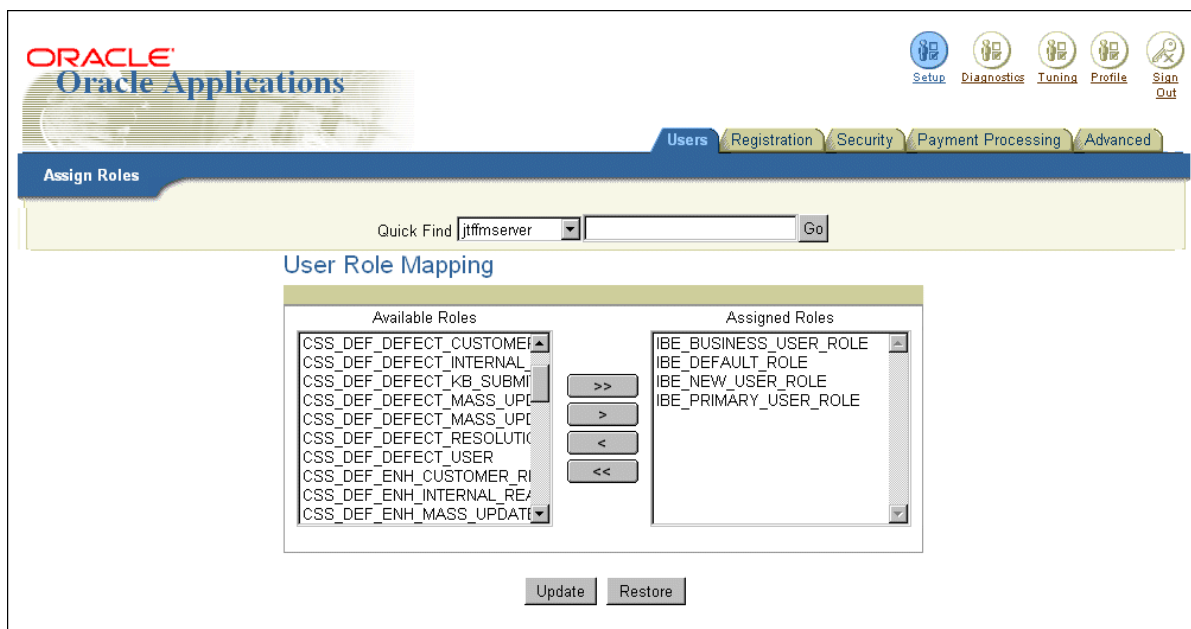
**Note:** If you assign a role with the IBE\_USER\_ADMIN permission to this user, then the set of roles that you assign to the user in the User Role Mapping screen is the same set of roles that he or she can assign to other B2B users for the customer organization.

---

---

- c. Click **Update** to save the role assignments for the user.

Figure 6–8 The User Role Mapping Screen



7. In the User Details screen, follow these steps to reset the customer's password:
  - a. Enter a new password in the New Password field.
  - b. Click **Reset Password**.  
The Acknowledgment screen appears with a confirmation of the password change.
  - c. In the Acknowledgment screen, either click **Edit** to continue editing user data in the User Details screen, or click **User Roles** to modify user role assignments for the customer in the User Role Mapping screen.

## 6.5 Managing the Cache

Oracle iStore 11*i* caches your Web storefront product items and sections to improve performance of your Web site. However, if you make changes to a cached product item or section, the changes are not visible in the Customer UI unless the product item or section is purged from the cache.

Restarting the Apache server purges the entire cache, but you may not always want to do this, since it removes product items or sections you do not want to remove from the cache, and since the Web stores are unavailable while the server is restarting.

Oracle iStore 11*i* offers a cache management feature in the Merchant UI that enables you to purge only the product items or sections that you want to remove from the cache, while keeping the server—and thus the Web stores—up and running. You can purge the cache of the following sets of product items and sections:

- Specific product item(s)
- Specific section(s)
- All product items
- All sections
- The entire cache

Oracle iStore 11*i* uses multicast messages when it purges the cache to ensure that the cache on each Java Virtual Machine (JVM) is purged. The Oracle iStore 11*i* initialization code starts a thread within each JVM that listens for messages on a specific address and port. When you choose to purge the cache from the Merchant UI, Oracle iStore 11*i* sends a multicast message to that address and port. When a thread receives the message, it interprets the message and purges the appropriate cache. Since there is a thread on each JVM listening for messages, the cache will be purged in each JVM. The port number is set in the profile IBE: Port Number to use for multicast messages. See [Chapter 7](#) for more information about this profile.



## 6.5.1 Purging the Entire Cache

Use this procedure to purge either the entire section cache or the entire product cache, or both.

### Steps

1. Launch the Merchant UI.

2. Enter the Cache tab.

If necessary, click on the Purge Entire Cache link.

The Purge Entire Cache screen appears.

3. Highlight the radio button next to the cache purging option you want to exercise:
  - Purge the entire Section Cache and entire Item Cache
  - Purge the entire Section Cache only
  - Purge the entire Item Cache only

4. Click **Update**.

The cache is purged.

## 6.5.2 Purging the Section Cache

Use this procedure to purge individual sections from your section cache.

### Steps

1. Launch the Merchant UI.

2. Enter the Cache tab.

If necessary, click on the Purge Section Cache link.

The Purge Section Cache screen appears.

3. Search for the section(s) you want to purge:
  - a. From the View pull-down menu, specify whether you will search by **Name**, **Section Code**, **Section Type**, or **Status**.
  - b. In the adjacent text field, enter your search criterion. Use % as a wild-card character if necessary.
  - c. Click **Go**.

The search results appear in the Purge Section Cache screen.

4. Select the checkbox in the Select column next to the section(s) you want to purge.
5. Click **Update**.

The cache is purged.

### 6.5.3 Purging the Product Cache

Use this procedure to purge individual product items from your product cache.

#### Steps

1. Launch the Merchant UI.
2. Enter the Cache tab.

If necessary, click on the Purge Product Cache link.

The Purge Product Cache screen appears.

3. Search for the section(s) you want to purge:
  - a. From the View pull-down menu, specify whether you will search by **Name**, **Part Number**, **Category**, or **Status**.
  - b. In the adjacent text field, enter your search criterion. Use % as a wild-card character if necessary.
  - c. Click **Go**.

The search results appear in the Purge Product Cache screen.

4. Select the checkbox in the Select column next to the product(s) you want to purge.
5. Click **Update**.

The cache is purged.

---

---

# Profile Options, Accounts, and Forms Settings

This chapter describes profile option settings, account setups, and Oracle Forms settings that are required for successful implementation. Topics include:

- [Before You Begin](#)
- [Setting Up Store Manager User Accounts](#)
- [Setting Up the Guest User Account](#)
- [Setting Up JTF Properties](#)
- [Setting Foundation \(JTF\) Profile Options](#)
- [Setting Oracle iStore \(IBE\) Profile Options](#)
- [Setting Profile Options for Language and Currency](#)
- [Setting Order Capture \(ASO\) Profile Options](#)
- [Setting Order Management \(OM\) Profile Options](#)
- [Setting Multi Organization \(MO\) Profile Options](#)
- [Setting Oracle Contracts Core \(OKC\) Profile Options](#)
- [Setting Up an Oracle iMarketing User](#)
- [Setting Up Concurrent Program Manager](#)
- [Setting Up Order Management in Forms](#)
- [Setting Up Site-Level Profile Options](#)
- [Setting Up Order Capture in Forms](#)
- [Understanding Cookies](#)

## 7.1 Before You Begin

Before making Oracle Forms settings, ensure that all Oracle Applications server processes are up and running. In particular, if you stopped concurrent managers before applying Oracle Applications patchsets, restart them now by changing to `$COMMON_TOP/admin/scripts`, and executing `adcmctl.sh <APPS username/>APPS password> start`.

## 7.2 Setting Up Store Manager User Accounts

An Oracle iStore 11i store manager performs tasks in the Merchant UI. To do this, the store manager must have a user name with the IBE\_ADMINISTRATOR responsibility.

When the store manager logs in to the Oracle CRM Applications login page at:

`http://<host>:<apache port>/OA_HTML/jtfllogin.jsp`

with this user name, the Oracle iStore 11i Merchant UI opens.

Use the following procedure to set up a user account for an Oracle iStore 11i store manager.

### Steps

1. Launch Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

2. Log in with the System Administrator responsibility.

3. Choose **Security > User > Define**.

The Users window opens.

4. In the User Name field, enter the user name that the store manager will use to log in to the Oracle iStore 11i Merchant UI.
5. In the Password field, enter the store manager's password.
6. In the Responsibilities block, use the LOV button in a Responsibility field to assign the IBE\_ADMINISTRATOR responsibility to the user name.
7. Click the Save icon in the toolbar to save this user record.

8. From the Navigator - System Administrator window, choose **Profile > System**.  
The Find System Profile Values window opens.
9. Check the checkbox next to the User field, and use the User LOV to search for and enter the store manager's user name.
10. In the Profile field, enter JTF\_PROFILE%.
11. Click **Find**.  
The System Profile Values form opens with the results of your search.
12. Verify and/or set the JTF profile options listed in the following table at the user level for this store manager:

**Table 7-1 User-Level JTF Profile Options for Store Managers**

Profile Option Name	Value	Description
JTF_PROFILE_DEFAULT_APPLICATION	671	Default application ID (671=iStore).
JTF_PROFILE_DEFAULT_RESPONSIBILITY	21819	Default responsibility ID (21819=IBE_ADMINISTRATOR).

See *Oracle Applications System Administrator's Guide, Release 11i* for more information on creating additional users and assigning responsibilities.

## 7.3 Setting Up the Guest User Account

If you want guest users to be able to browse your Web stores, you must define a guest user name, assign a responsibility to it, and set up JTF profile options and properties. Anonymous users who visit your Web stores are then automatically logged in with the guest user name.

If a guest user makes any changes, such as modifying the preferred language or currency, or adding items to the shopping cart, the changes are saved in the cookie so that two anonymous users cannot see each other's changes.

You must set up the guest user account before customers can view your Web stores' home page. If you do not set up the guest user account, customers will be unable to view or register in your Web stores, and the Oracle iStore 11i system administrator will have to create a user name for each customer before the customer can access the stores.

Use the following procedure to create the Oracle iStore 11i guest user account.

## Steps

1. Launch Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

2. Log in with the System Administrator responsibility.
3. Choose **Security > User > Define**.  
The Users window opens.
4. In the User Name field, enter a user name, such as `IBEGUEST`, by which a guest user will be automatically logged in to Oracle iStore 11i.
5. In the Password field, enter a password for this user name.
6. In the Responsibilities block, use the LOV button in a Responsibility field to assign an Oracle iStore 11i customer responsibility, such as **IBE\_CUSTOMER**, to the user name.
7. Click the Save icon in the toolbar to save this user record.
8. From the Navigator - System Administrator window, choose **Profile > System**.  
The Find System Profile Values window opens.
9. Check **Application**, and choose **iStore** from the Application LOV.
10. Check **User**, and choose the guest user name from the User LOV.
11. In the Profile field, enter `JTF_PROFILE%`.

12. Click **Find**.

The System Profile Values form opens with the results of your search.

13. Set the JTF profile options listed in the following table at the user level for the guest user name:

**Table 7-2 User-Level JTF Profile Options for the Guest User**

Profile Option Name	Value	Description
<code>JTF_PROFILE_DEFAULT_APPLICATION</code>	<code>APPLICATION_ID</code> value of the guest user's responsibility	Default application ID

**Table 7–2 User-Level JTF Profile Options for the Guest User (Cont.)**

Profile Option Name	Value	Description
JTF_PROFILE_DEFAULT_CURRENCY	Currency code for the guest user's default currency, e.g., USD for U.S. dollars	Default currency
JTF_PROFILE_DEFAULT_RESPONSIBILITY	RESPONSIBILITY_ID	Default responsibility ID value of the guest user's responsibility

14. Next, set up JTF properties to make this user name the Oracle CRM guest user. See [Section 7.4, "Setting Up JTF Properties"](#) for more information.

### Guidelines

For the seeded IBE\_CUSTOMER responsibility, the APPLICATION\_ID value is 671 (for iStore), and the RESPONSIBILITY\_ID value is 22372.

Use the following procedure to find out the APPLICATION\_ID value and RESPONSIBILITY\_ID value of the guest user's responsibility if it is not IBE\_CUSTOMER.

### Steps

1. Launch Oracle Forms by navigating to:  

```
http://<host>:<apache port>/
```

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in with the System Administrator responsibility.
3. Choose **Security > Responsibility > Define**.  

The Responsibilities form opens.
4. Choose **View > Find**. Search for the responsibility that you assigned to the guest user, highlight it, and click **OK** in the search window.  

The Responsibilities form is populated with the record for the responsibility that you chose.
5. With your cursor in any field of the record, choose **Help > Diagnostics > Examine**.

The Examine Field and Variable Values window opens.

6. In the Examine Field and Variable Values window, choose **APPLICATION\_ID** in the Field LOV.

The Value field in the Examine Field and Variable Values window is populated with the value of APPLICATION\_ID.

7. In the Examine Field and Variable Values window, choose **RESPONSIBILITY\_ID** in the Field LOV.

The Value field in the Examine Field and Variable Values window is populated with the value of RESPONSIBILITY\_ID.

### Verifying the Guest User

After setting up the guest user account, responsibilities, JTF profile options, and the JTF properties, you should verify that the guest user works properly.

Use the following procedure to verify that you have set up the guest user correctly.

### Steps

1. Log out of any Oracle applications with which you have been working.
2. Restart the Apache server.
3. Navigate to the URL:

```
http://<host>:<apache port>/OA_HTML/ibeCZzdMinisites.jsp
```

The page that opens should have a list of the specialty stores that are accessible to guest users.

See *Oracle Applications System Administrator's Guide, Release 11i* for more information on creating additional users and assigning responsibilities.



## 7.4 Setting Up JTF Properties

Use the following procedure to set up JTF properties.

### Steps

1. Log in as SYSADMIN to the Oracle CRM Applications login page at:  
`http://<host>:<apache port>/OA_HTML/jtflogin.jsp`
2. Choose **Advanced > Properties**, and select **JTF** from the View pull-down menu to view JTF properties.
3. Click **Next** to go to the next page. Set **guest\_password** and **guest\_username** to the password and user name, respectively, that you specified when creating the guest user according to the instructions in [Section 7.3, "Setting Up the Guest User Account"](#).
4. Optional: To change the password, click on **guest\_password**.

---

---

**Note:** If you change the password here, the change is not reflected in Oracle Forms. To maintain consistency you must also change the password in Oracle Forms, using the System Administrator responsibility.

---

---

5. In the Properties screen, find and click on **framework.Logging.system.level**, set sequence 1 value to **debug**, and click **Update**.

The Properties screen opens.

6. Click **Update**.

The Properties screen refreshes.

7. In the Properties screen, find and click on **service.Logging.common.level**, change it to **debug**, and click **Update**.

The Properties screen opens.

8. Click **Update**.

The Properties screen refreshes.

See *Oracle CRM Foundation Implementation Guide* for more information about JTF properties.

## 7.5 Setting Foundation (JTF) Profile Options

Set JTF profile options for the Oracle iStore 11*i* Merchant UI and Customer UI.

### JTF Profile Options for Merchant UI

These profiles must be set before the Merchant UI can be launched. The values of these profiles determine the Oracle CRM Foundation (JTF) default elements and values. These profiles are seeded in the Profiles form in ERP and the values are defined by the user (System Administrator).

---

---

**Note:** Make sure you set the JTF profile options for each store manager user as well. See [Section 7.2, "Setting Up Store Manager User Accounts"](#) for more information.

---

---

### Steps

1. Launch Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Go to **Profile > System**.

The Find System Profile Values window opens.

5. Check both **Site** and **Application**. Enter **iStore** for the application name.
6. In the Profile field, enter `JTF_PROFILE%`.
7. Click **Find**.

The System Profile Values form opens with the results of your search.

8. Verify and/or set the JTF profile options listed in the following table at both the site level and the application level for iStore:

**Table 7–3 JTF Profile Options**

Profile Option Name	Value	Description
JTF_PROFILE_DEFAULT_APPLICATION	671	Default application ID (671=iStore).
JTF_PROFILE_DEFAULT_BLANK_ROWS	3	Number of blank rows on Merchant UI forms (can be set to any integer > 0).
JTF_PROFILE_DEFAULT_CSS	jtfucss.css	
JTF_PROFILE_DEFAULT_CURRENCY	USD	Default currency
JTF_PROFILE_DEFAULT_NUM_ROWS	10	
JTF_PROFILE_DEFAULT_RESPONSIBILITY (application level only)	21819	Default responsibility ID (21819=IBE_ADMINISTRATOR).

### JTF Profile Options for Customer UI

Set the JTF profile options for the guest user as indicated in [Section 7.3, "Setting Up the Guest User Account"](#). These profiles must be set before the Customer UI can be brought up.

## 7.6 Setting Oracle iStore (IBE) Profile Options

The IBE profile options and values define the way the Customer and Merchant UIs will work. The profiles set is seeded in the product and displayed when the administrator sets the values for these options.

### Steps

1. Launch Oracle Forms by navigating to:  

```
http://<host>:<apache port>/
```

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Go to **Profile > System**.

The Find System Profile Values window opens.

5. Check **Application**, and choose **iStore**.
6. In the Profile field, enter **IBE%**, and click **Find**.
7. Verify and/or set the iStore (IBE) profile options listed below at the application level only, unless another level is specified.

### **IBE Profile Options for the Oracle iStore 11i Merchant UI**

The following table lists the IBE profile options that configure the Oracle iStore 11i Merchant UI.

**Table 7-4 IBE Profile Options for Oracle iStore 11i Merchant UI Setup**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Category Set	Yes	This profile is used when searching for products in the Merchant UI based on category. (The value "Inv. Items" is for the Vision database.)  See <a href="#">Table 7-6</a> for profile description relevant to Customer UI.
IBE: Item Validation Organization	Yes	This profile specifies the inventory organization when adding products to the catalog. It allows only the items belonging to this inventory organization to be added to the catalog hierarchy.  For vision demo, master organization is Vision Operations (204).  Recommended Value: Master Inventory Organization
IBE: Number of Days for New Item Definition	No	An item created within the number of days specified in this profile is considered a new item in the Merchant UI. If this profile is not set, only items created on the current day are considered new items.

See [Section 7.5, "Setting Foundation \(JTF\) Profile Options"](#) for JTF profile options needed to set up the Merchant UI.

## IBE Profile Options for the Oracle iStore 11i Customer UI

The remaining tables in this section list the IBE profile options that configure the Oracle iStore 11i Customer UI. They are categorized according to the aspect of the Customer UI that they impact: Catalog, Template Manager, Shopping Cart, Express Checkout, Postsales, Notifications, Caching, and Functionality.

**Table 7-5 IBE Profile Options for Catalog**

Profile Option Name	Mandatory	Description
<b>Sections</b>		
IBE: Items Per Section for Display	No	This profile specifies the maximum number of items to display per section. If the profile is not set, all items in a section will be displayed.
IBE: Lines Per Section for Multiple Section Display	No	This profile specifies the number of lines per section when the application displays multiple sections on a catalog page. If no value is specified, the value in the profile IBE: Items Per Page for Display is used.
IBE: Number of Menu Subtabs	No	This profile specifies the number of menu subtabs. It defaults to value 5 if not set.
IBE: Number of Menu Tabs	No	This profile specifies the number of menu tabs. It defaults to value 5 if not set.
IBE: Sections Per Page for display	No	This profile specifies the total number of sections to be displayed per catalog page. If no value is specified, the number of sections per page is not limited.
IBE: Use Catalog exclusions	No	This profile specifies whether to use catalog exclusions in the specialty store. If exclusions are never specified in any specialty store, then set to <b>No</b> to improve performance.
IBE: Use Global Bin	No	This profile enables/disables the global bin. If no value is specified, the global bin is enabled. The global bin allows users to move from one specialty store to another without logging in again.
IBE: Use Section Bin	No	This profile enables/disables the section bin. It defaults to value <b>Yes</b> and displays the section bin if not set.

**Table 7-5 IBE Profile Options for Catalog (Cont.)**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Use Section Path	No	This profile enables/disables the section path. It defaults to value <b>Yes</b> and displays the section path if not set.
IBE: Use Welcome Bin	No	This profile enables/disables the welcome bin. If no value is specified, the welcome bin is enabled.
<b>Items</b>		
IBE: Items Per Page for Display	No	This profile specifies the number of items to display on a leaf section page. If the number of items in the section exceeds this value, the page displays a Next link. This profile defaults to value 20 if not set.
IBE: Pricing Event — Before Shopping Cart	Yes	This profile specifies the user-defined pricing event (defined in Pricing) for the catalog stage.  Recommended value: <b>Enter Order Line</b>
IBE: Retrieve All Units of Measure for an Item	No	If this profile is set to <b>Yes</b> , the application retrieves all units of measure with prices for an item and displays them in a UOM pull-down menu in the Web store catalog. If this profile is set to <b>No</b> , the application retrieves only the primary unit of measure and its prices. The application retrieves all units of measure if this profile is not set.
IBE: Retrieve Price When Displaying Items	No	If the profile is set to <b>Yes</b> , the application retrieves prices for an item's primary UOM based on the specialty store's price list when loading the item. Otherwise, prices will be retrieved when the price APIs are called.  Recommended values: <b>Yes</b> if retrieving prices only for the primary UOM. Otherwise, <b>No</b> .

**Search**

**Table 7–5 IBE Profile Options for Catalog (Cont.)**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Enable Fuzzy Search	No	If the profile is set to <b>Yes</b> , the application allows users to perform fuzzy product searches, so that they do not have to type in the exact spelling of their search criteria to retrieve results that match these criteria. If the profile is not set, it defaults to value <b>No</b> .
IBE: Search Lines Per Page	No	This profile specifies the number of search results displayed on a single search result page. It defaults to value 20 if not set.
IBE: Use Category Search	No	If this profile is set to <b>Yes</b> , it activates Category search so that the home page pull-down menu shows categories with publishable items. If this profile is set to <b>No</b> , it activates Section search so that the home page pull-down menu shows top level sections for the minisite where the customer is browsing. Null value removes the home page pull-down menu so that search is only enabled in all products.  You must set this profile option to <b>Yes</b> if the IBE: Use CABO UI profile option is set to <b>No</b> .

**Table 7–6 IBE Profile Options for Template Manager**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Category Set	Yes	This profile specifies the category set for determining the display category for an item. If an item belongs to a category in this category set, the Customer UI will use the display style mapping for the category if an association is not found at the item level. (The value "Inv. Items" is for the Vision database.)

**Table 7-7 IBE Profile Options for Shopping Cart**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Authorize Payment Offline during normal Checkout	No	This profile specifies if payment authorization offline before checkout is allowed. <b>Yes</b> enables offline authorization. <b>No</b> allows only online authorization. If the profile is not set, it defaults to value <b>Yes</b> .
IBE: Create Order in Entered State, if it has errors while booking	No	This profile specifies if the order should be created in the Entered state even if there are errors while booking it. If this profile is not set, it defaults to value <b>No</b> .
IBE: Create Standard Contract	No	This profile specifies whether a contract is created when a customer agrees to standard terms and conditions. If the profile is set to <b>Yes</b> , a contract is created in the signed state when the customer places the order. If the profile is set to <b>No</b> , no contract is created. If the profile is not set, it defaults to value <b>Yes</b> .  You need to set this profile option only if the profile option ASO: Enable Use Contracts is set to <b>Yes</b> .
IBE: Default Payment Term	No	This profile specifies the default payment term. If the profile is not set, the option is ignored.
IBE: Merge Shopping Cart Lines	No	This profile specifies whether to merge item lines in the shopping cart if the same item is added to the cart more than once. If the profile is not set, it defaults to value <b>No</b> .
IBE: Preferred Shipping Method (User level)	No	This profile specifies preferred shipping method.
IBE: Pricing Event for Shopping Cart	Yes	The merchant can choose any user-defined pricing event for processing the price for the shopping cart. These user-defined pricing events should first be created in Pricing.  Recommended value: <b>Enter Order Line</b>



**Table 7-7 IBE Profile Options for Shopping Cart (Cont.)**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Recalculate Price in Order Management	No	If the profile is set to <b>No</b> , the quote-related prices are passed on to Order Management unchanged. If the profile is set to <b>Yes</b> , the quote-related prices can be changed in the order because Order Management is allowed to recalculate the prices for the quote. If the profile is not set, it defaults to value <b>No</b> .
IBE: Request Type to get a Price	Yes	Select the transaction application (e.g., Order Management) that calls the Pricing engine.
IBE: Shopping Cart Expiration Duration	No	This profile specifies the number of days that shopping carts are maintained and available to the merchant and customer. If the profile is not set, it defaults to value 0.
IBE: Shopping Cart Price based on Owner	No	If the profile is set to <b>Yes</b> , the shopping cart price will be based on the shopping cart owner if retrieved by someone sharing the cart. Otherwise, the cart price will be recalculated based on modifications to the cart made by the person sharing the cart. If the profile is not set, it defaults to value <b>No</b> .
IBE: Use Price list associated with Specialty Store	No	If the profile is set to <b>Yes</b> , the price list at the specialty store is used for registered and B2B users. If the profile is set to <b>No</b> , Oracle iStore 11i passes a null price list with the party ID and account ID to the Pricing engine, which then determines a price list for which the user qualifies. If the profile is not set, it defaults to value <b>Yes</b> .

**Table 7-7 IBE Profile Options for Shopping Cart (Cont.)**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Use Sensitive Pages	No	<p>A sensitive page is a page that displays personal user-specific information. In Oracle iStore 11i, the checkout and account pages are sensitive.</p> <p>The checkout pages are the pages that appear from the point when users click <b>Checkout</b>, until they place an order. Some of the pages in the checkout flow include billing information, shipping information, and order review.</p> <p>The account pages are the pages where users view and update their passwords, names, phones, emails, addresses, credit cards, and Express Checkout settings. For B2B users, the User Management and Role Management pages are also sensitive.</p> <p>Catalog (sections and products), shopping cart, and Order Tracker pages are not sensitive.</p> <p>If the profile is set to <b>Yes</b>, the application reauthenticates a logged-in user before displaying a sensitive page. If the profile is set to <b>No</b>, reauthentication does not take place. If the profile is not set, it defaults to value <b>Yes</b>.</p>

**Table 7-8 IBE Profile Options for Express Checkout**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Express Checkout Consolidation Time Interval	No	This profile specifies the time interval, in minutes, in which the Express Checkout shopping cart will be converted in an order by the concurrent batch job. The profile defaults to value 60 if not set.

**Table 7–9 IBE Profile Options for Postsales**

Profile Option Name	Mandatory	Description
IBE: Number of Invoice/Order Lines displayed	No	This profile specifies the number of invoice/order lines displayed in Order Tracker. The profile defaults to value 10 if not set.
IBE: Use Auth Permissions in Order Tracker	No	This profile specifies if permission checking is enforced for users to view only the orders placed by themselves or all of their organization's orders. The profile defaults to value <b>No</b> if not set.

**Table 7–10 IBE Profile Options for Notifications**

Profile Option Name	Mandatory	Description
IBE: Default Order Admin to Send Workflow Notification	No	This profile specifies the default order admin's Oracle Workflow user name. An e-mail is sent to this order admin upon errors in submitting order.  If the profile is set to <b>Yes</b> , the profile option IBE: Use Workflow Features in iStore must also be set to <b>Yes</b> .
IBE: Default Sales Assistant to Send Workflow Notification	No	This profile specifies the default sales assistant's Oracle Workflow user name. An e-mail notification is sent to the sales assistant when a customer requests assistance.  If the profile is set to <b>Yes</b> , the profile option IBE: Use Workflow Features in iStore must also be set to <b>Yes</b> .
IBE: Email Promotions (User level)	No	The merchant can use this profile to send e-mail promotions if the user chooses this option during registration. The e-mail notification for promotions is for customization only.
IBE: Notification User Role	No	This profile specifies the user responsibility for setting notifications.  Recommended value: <b>System Administrator</b>

**Table 7–11 IBE Profile Options for Caching**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Cache	No	This profile specifies whether to enable the store cache for sections and items. If the profile is not set, section and item cache are disabled.  Recommended value: <b>Yes</b>
IBE: Enable Preloading of Cache for Catalog	No	If this profile is set to <b>Yes</b> , the application preloads the catalog when the first user logs in, depending on the size specified in the cache limits profile. At the expense of the first hit, the subsequent catalog search and navigation become faster. If the profile is not set, the application does not preload the section and item cache.
IBE: Item Cache Size	No	This profile specifies the maximum number of items to cache on the middle tier. The profile defaults to value 200 if not set, and if the cache is on.
IBE: No of Results in Search	No	This profile specifies the number of hits returned by a store search. The profile defaults to value 200 if not set.
IBE: Order Tracker Object Cache	No	This profile specifies whether Order Tracker Java objects are cached in the middle tier.  Recommended value: <b>Yes</b>
IBE: Port Number to use for multicast messages	No	This profile specifies the port number to use for multicast messages. If the profile is not set, it defaults to port 50000.
IBE: Preload MiniSite Cache	No	This profile specifies whether the specialty store cache is preloaded to the middle tier. <b>Yes</b> preloads the cache. If the profile is not set, it defaults to value <b>No</b> .
IBE: Section Cache Size	No	This profile specifies the maximum number of sections to cache on the middle tier. The profile defaults to value 100 if not set, and if the cache is on.

**Table 7–12 IBE Profile Options for Functionality**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Use AOL Menu	No	This profile specifies whether to use AOL's menu framework.
IBE: Use B2B Features	No	This profile specifies whether B2B features such as business user registration, My Account user administration, and My Account role management are available to the customer. If the profile is not set, it defaults to <b>Yes</b> .  See <i>Oracle iStore Concepts and Procedures</i> for a detailed review of Oracle iStore 11i B2B features.
IBE: Use CABO UI	No	This profile specifies whether to use CABO or Mona Lisa style for the Customer UI. <b>Yes</b> enables CABO style. <b>No</b> enables Mona Lisa style. If the profile is not set, it defaults to value <b>Yes</b> .  Recommended value: <b>Yes</b> . Functionality is fully enabled only with the CABO style.
IBE: Use Direct Item Entry	No	This profile specifies whether customer-to-merchant part number mapping is available. If the profile is not set, it defaults to value <b>Yes</b> .
IBE: Use Express Checkout	No	This profile specifies whether Express Checkout is enabled. If the profile is not set, it defaults to value <b>Yes</b> .
IBE: Use iMarketing Postings	No	This profile specifies whether Oracle iMarketing postings can display on Oracle iStore 11i Web store pages. <b>Yes</b> enables Oracle iMarketing postings. <b>No</b> disables Oracle iMarketing postings. If the profile is not set, it defaults to value <b>Yes</b> .
IBE: Use Shop List	No	This profile specifies whether shopping list functionality is available to customers. If the profile is not set, it defaults to value <b>Yes</b> .

**Table 7–12 IBE Profile Options for Functionality (Cont.)**

<b>Profile Option Name</b>	<b>Mandatory</b>	<b>Description</b>
IBE: Use Support	No	<p>If the profile is set to <b>Yes</b>, it enables the purchase of service items for serviceable items that are in the shopping cart, through the Select Technical Support pull-down menu in the shopping cart. If the profile is not set, it defaults to value <b>Yes</b>.</p> <p>If the profile is set to <b>Yes</b>, the profile IBE: Use Support Cart Level must also be set to <b>Yes</b>.</p>
IBE: Use Support Cart Level	No	<p>If the profile is set to <b>Yes</b>, it enables the purchase of service items for serviceable items that are in the shopping cart, through the Select Technical Support pull-down menu in the shopping cart. If the profile is not set, it defaults to value <b>Yes</b>.</p> <p>If the profile is set to <b>Yes</b>, the profile IBE: Use Support must also be set to <b>Yes</b>.</p>
IBE: Use Workflow Features in iStore	No	<p>This profile specifies whether Oracle Workflow features, such as e-mail notifications to customers regarding registration, orders, and order status requests, are available in your instance. If the profile is not set, it defaults to value <b>Yes</b>.</p> <p>If the profile is set to <b>Yes</b>, the profile IBE: Use CABO UI must also be set to <b>Yes</b>.</p>

## 7.7 Setting Profile Options for Language and Currency

Set the FND\_LANGUAGES and FND\_CURRENCY profile options in AOL with the appropriate languages and currencies. Defaults for Oracle iStore 11i come from the list of values populated by the values that you entered in AOL for these profile options.

## 7.8 Setting Order Capture (ASO) Profile Options

ASO profile options must be set up to enable Order Capture to work with Oracle iStore 11i. Use the following procedure to set the required profile options.

### Steps

1. Launch Oracle Forms by navigating to:  

```
http://<host>:<apache port>/
```

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Choose **Profile > System**.  

The Find System Profile Values window opens.
5. Check **Application, Site, and Responsibility**. Enter Oracle Order Capture as the application and IBE\_CUSTOMER as the responsibility.
6. In the Profile field, enter ASO%, and click **Find**.
7. Verify and/or set the Order Capture (ASO) profile options listed in the following table at both the site level and the responsibility level for IBE\_CUSTOMER:

**Table 7-13 ASO Profile Options**

Profile Option Name	Value	Description
ASO: Credit Card Authorization	No	If set to <b>Yes</b> , payment is authorized at the time the shopping cart is created.  Recommended value: <b>No</b>
ASO: Default Order Type	Standard	Determines how the order is to be processed in Oracle Order Management. The order types are set up in Oracle Order Management. This profile determines what price list and currency code appears by default in the main Order Capture form.

**Table 7–13 ASO Profile Options (Cont.)**

<b>Profile Option Name</b>	<b>Value</b>	<b>Description</b>
ASO: Default Salesrep	No Sales Credit	The default sales representative who is allocated the sales credits for booked orders when the user is not entered as a sales representative.
ASO: Enable Use Contracts	Yes/No	<b>Yes</b> activates integration with Oracle Contracts, to enable Oracle Contracts-related features such as requirement of agreement with terms and conditions before checkout, and negotiation of terms and conditions. Before setting this profile option to <b>Yes</b> , you must install Oracle Contracts and set the profile option OKC: Contract Template Name for Terms. If ASO: Enable Use Contracts is not set, it defaults to value <b>No</b> .
ASO: Product Organization	Master Inventory Organization	The organization that Order Capture uses to validate inventory items.
ASO: Quote Order Type	Standard	
ASO: Reservation Level (site level only)	Null	

8. In the Navigator - System Administrator window, choose **Profile > System**.  
The Find System Profile Values window opens.
9. Check **Application** and **Responsibility**. Enter `iStore` as the application and `IBE_CUSTOMER` as the responsibility.
10. In the Profile field, enter `ASO%`, and click **Find**.
11. Verify and/or set the ASO: Quote Conversion Type profile option at both the application level and the responsibility level for `IBE_CUSTOMER` to the same value as the Oracle Order Capture application level setting.



## 7.9 Setting Order Management (OM) Profile Options

1. Launch Oracle Forms by navigating to:  
`http://<host>:<apache port>/`  
and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Choose **Profile > System**.  
The Find System Profile Values window opens.
5. Check **Application**. Enter `iStore` as the application.
6. In the Profile field, enter `OM%`, and click **Find**.
7. Verify and/or set the OM: Set of Books profile option at the iStore application level to the same value as the Oracle Order Management application level setting.

## 7.10 Setting Multi Organization (MO) Profile Options

Use the following procedure to set up multi organization profile options.

### Steps

1. Launch Oracle Forms by navigating to:  
`http://<host>:<apache port>/`  
and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).
2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Go to **Profile > System**.  
The Find System Profile Values window opens.
5. Check **Site** and **Responsibility**. Enter `IBE_CUSTOMER` as the responsibility.
6. In the Profile field, enter `MO%`, and click **Find**.

7. Verify and/or set the MO: Operating Unit profile option to the operating unit at the responsibility level for IBE\_CUSTOMER.

### **Setting Up MO Profile Options for Multiple Operating Units**

For a multiple operating unit environment, you must create a separate IBE customer responsibility for each operating unit in Oracle Forms. You can use the seeded responsibility IBE\_CUSTOMER as one of these responsibilities.

For each of the customer responsibilities, set the profile option MO: Operating Unit to its respective operating unit.

Each customer name is assigned one of these responsibilities when the customer name is approved. When doing this, the seeded responsibility IBE\_CUSTOMER, if different from the responsibility the customer receives, should be obsoleted for the customer name so that only one IBE responsibility is effective at a time.

When a customer enters a Web store, Oracle iStore 11i notes the customer's responsibility and the operating unit to which it is assigned, then restricts customers to the items in the Inventory Organization associated with the operating unit. Oracle iStore 11i accomplishes this by retrieving the Inventory Organization ID from OE\_SYSTEM\_PARAMETERS\_ALL, which depends on the current user responsibility's operating unit.

## **7.11 Setting Oracle Contracts Core (OKC) Profile Options**

Use the following procedure to set up Oracle Contracts Core (OKC) profile options at the responsibility level, if you are integrating Oracle iStore 11i with Oracle Contracts Core.

### **Steps**

1. Launch Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX).

2. Log in as SYSADMIN.
3. Choose the System Administrator responsibility.
4. Go to **Profile > System**.

The Find System Profile Values window opens.

5. Check **Responsibility**. Enter `IBE_CUSTOMER` or another customer user responsibility name as the responsibility.
6. In the Profile field, enter `OKC%`, and click **Find**.
7. Verify and/or set the OKC: Contract Template Name For Terms profile option to the default Oracle Contracts template name for Oracle iStore 11i, for the selected responsibility.

This procedure allows you to set up different Oracle Contracts templates for each customer user responsibility that you create for Oracle iStore 11i.

## 7.12 Setting Up an Oracle iMarketing User

Refer to *Oracle Marketing Online Implementation Guide, Release 11i* for instructions on setting up an Oracle iMarketing user.

## 7.13 Setting Up Concurrent Program Manager

In Oracle iStore 11i, there are three concurrent jobs seeded in the Oracle Forms concurrent program manager request setup: iStore Search Insert, iStore Section Search Refresh, and iStore - One Click Consolidation.

### 7.13.1 iStore Search Insert

This is a single request concurrent program that you execute once in the initial setup of Oracle iStore 11i as a post-install step. Run this batch job after you have loaded Oracle Inventory items and set the Inventory Organizations. This program populates Oracle iStore 11i's category search table with product information from the Oracle Inventory tables.

You may need to rerun this program under one of the following three conditions:

- If you add multiple items that do not appear in the search table
- If you want to purge all items from the search table
- If you change the setting of the fuzzy search functionality, by either enabling or disabling it

### 7.13.2 iStore Section Search Refresh

This concurrent program populates Oracle iStore 11i's section search table with product information. Run this program after running iStore Search Insert to switch from category-level to section-level search on the Customer UI. Rerun iStore Section Search Refresh to update the section search table whenever you modify the Oracle iStore 11i hierarchy.

If you have enabled a section-level search, rerun this concurrent program whenever you change the setting of the fuzzy search functionality.

### 7.13.3 iStore - One Click Consolidation

This is not a single request job and should be running as a batch process at pre-determined intervals. Schedule this program as a periodically running batch job. This program converts the one-click shopping carts in orders. The programs pick up all the one-click shopping carts, which are not "touched" for the time that is specified in the profile option "Express Checkout Consolidation Time Interval."

Access this setup through Oracle Forms.

---



---

**Note:** Do not run search insert concurrent program more than once, otherwise it will APPEND records in the search table.

---



---

With the concurrent programs, the iStore Concurrent Programs Responsibility is also seeded. In the initial setup, log in to Oracle Forms with System Administrator responsibility, and associate the seeded iStore Concurrent Programs Responsibility to users who should have it. Another iStore related responsibility is IBE\_ADMINISTRATOR, which is also seeded with Oracle iStore 11i. You should assign this responsibility to store manager users who need to work with the Oracle iStore 11i Merchant UI.

Before running the concurrent program, you need to set profile values. The concurrent program runs as a different responsibility than the end user responsibility. Establish the profile option values listed in the following table for the iStore Concurrent Programs Responsibility, which may or may not match the values of the IBE\_CUSTOMER responsibility.

**Table 7-14 Profile Option Values for the iStore Concurrent Programs Responsibility**

Profile Option Name	Value
ASO: Default Order Type	Standard

**Table 7-14 Profile Option Values for the iStore Concurrent Programs Responsibility**

Profile Option Name	Value
ASO: Default Salesrep	No Sales Credit
ASO: Validate Salesrep	No
IBE: Default Payment Term	IMMEDIATE

Generally, set the same values for the iStore Concurrent Programs responsibility profile options as for the IBE\_CUSTOMER responsibility to submit an order.

### Running the Concurrent Program to Submit Express Checkout Orders

1. Log in to Oracle Forms.
2. Select **iStore Concurrent Programs Responsibility** from the list of responsibilities.
3. The Submit a New Request window appears (with "Single Request" already selected on the radio button). Click **OK**.

The Submit Request window appears. The first field is labeled "Name," and there is a lookup button to the right of that field (three dots). Click it to get a selection of predefined programs.

4. The "Reports" window appears. Select **iStore - One Click Consolidation** and then click **OK**.
5. Set how often and when you want to run this by clicking **Schedule** in the "Submit Request" window.
  - To run right away, select **As Soon as Possible** and click **OK**.
  - To run once at a scheduled time, click **Once**, enter information in the appropriate fields, and click **OK**.
  - To run regularly, click **Periodically** or **On Specific Days**, enter information in the appropriate fields, and click **OK**.
6. Submit the Request by clicking **Submit**. You are prompted for confirmation and offered to submit another request. Click **NO**.

### Checking the Status of the Concurrent Program

1. Switch responsibility to System Administrator.
2. In the Navigator window, double-click **Concurrent**, and then **Requests**.

- The Find Requests window (defaulted to "All My Requests") appears. If the server is not busy, then clicking **Find** may be the fastest way to find your request.
  - If your server is busy, it may be better to enter search criteria and look for "Specific Requests."
3. After clicking **Find**, the "Requests" window displays a list of submitted requests. There should be one (or more) entitled "iStore - One Click Consolidation." Initially, you may find this in the "green" state with Phase = "pending" or "running." Click **Refresh Data** occasionally to check the completion status.
  4. Once in the "red" state or phase = "Completed," the "View Output" and "View Log" buttons should become active (if the log and output files have been setup correctly). Use these buttons to find out how many orders the concurrent program was able to successfully submit and how many failed.

## 7.14 Setting Up Order Management in Forms

Use the following procedure to set up Oracle Order Management in Forms.

### Steps

1. Log in to Oracle Forms.
2. Choose the Order Management Super User responsibility. (If you don't have this responsibility, use the System Administrator responsibility to grant it to yourself.)
3. Choose **Setup > Transaction Types > Define**.
4. Search for the Standard transaction type: press **F11**, click in the Transaction Type field, enter `S%`, press **Ctrl-F11**.
5. In the Order Workflow field, select **Order Flow - Generic** from the pull-down menu.
6. Click **Assign Line Flows**.
7. Click an empty row in the Assign Workflow Processes region.
8. Enter the following information:
  - In the Line Type field, enter `UPG_LINE_TYPE_ORDER_1000`.
  - In the Item Type field, enter `Standard Item`.

- In the Process Name field, enter `Line Flow - Generic`.
  - In the Start Date field, enter any valid date (e.g., today's date).
9. Click **OK**.
  10. Save the form.

### 7.14.1 Setting Up Web-Enabled Shipping Methods

To set up Web-enabled shipping methods within the Order Management Super User responsibility, use the following procedure. This procedure is necessary to make shipping methods available to your customers in your Web stores.

#### Steps

1. Log in to Oracle Forms as SYSADMIN.
2. Choose the Order Management Super User responsibility.
3. Choose **Shipping > Setup > Freight > Define Carrier Ship Methods**.
4. Select **Find All** from the View menu.
5. Check the Web Enabled box next to the shipping methods that you want to be available through the Web stores.
6. Save by clicking the Save icon in the toolbar or choosing **File > Save**.

## 7.15 Setting Up Site-Level Profile Options

You must set profile options at the site level.

#### Steps

1. Switch to the System Administrator responsibility.
2. Go to **Profile > System**.  
The Find System Profile Values window opens.
3. Check **Site** only.
4. Search for the Sequential Numbering profile option.
5. Set the site-level value to **Partially Used** from the LOV.
6. Save the form.

7. Go to **View > Find**, and search for ASO% profiles. Find **ASO: Default Salesrep**, and select **No Sales Credit** at the site level from the LOV.
8. Save the form.
9. Switch to the Application Developer responsibility.
10. Go to Profile.
11. Search for IBE%PAY% profiles; find **IBE\_DEFAULT\_PAYMENT\_TERM\_ID**.
12. Make sure that the System Administrator Access is both Visible and Updateable at both the Application and Site levels (i.e., all four checkboxes are checked).
13. Save the form.
14. Switch back to the System Administrator responsibility.
15. Go to **Profile > System > Find System Profile Values**.
16. Search for IBE% profiles. Find **IBE: Default Payment Term**, and set it to **IMMEDIATE** at the Site level.
17. Save the form.

## 7.16 Setting Up Order Capture in Forms

Use the following procedure to set up Oracle Order Capture.

### Steps

1. Switch to the Order Capture Sales Manager responsibility.
2. Choose **Quote Status Setup**.

Refer to Order Capture documentation for information on how to set up the Allowed Transition Status region. Create records for each value in the Code LOV (i.e., INACTIVE, LOST, ORDERED, PROBLEM, etc.).

3. Save the form.



## 7.17 Understanding Cookies

The user session in Oracle iStore 11*i* is controlled and identified by the help of cookies. The cookies are set on the user's browser and are used to identify return customers and other related data. The Oracle iStore 11*i* process is transparent to cookie administration, setup and control. Cookies are managed by Oracle CRM Foundation (JTF) methods. If the user turns off browser cookies, JTF makes sure that the information is available through the URL.

The cookie domain is set as the Web server domain, for example, "oracle.com" for Oracle's internal store. Once the user registers, the user account is created in the database and is used in the cookies to identify the customer. The Guest user account (Walkin user) is seeded with the product. Personal and Business account types are the basic account types in the store. The business account has the functionality of assigning roles to the business users.



---

---

## Oracle iStore 11i Catalog APIs

This chapter contains the following information about the Oracle iStore 11i Catalog public class APIs:

- [Catalog API Class Summary](#)
- [Class Item](#)
- [Class ItemFlexfield](#)
- [Class PriceObject](#)
- [Class Section](#)
- [Exception Classes for Package oracle.apps.ibe.catalog](#)

## 8.1 Catalog API Class Summary

APIs for the Oracle iStore 11i Catalog are in the package `oracle.apps.ibe.catalog`. The table below describes the classes briefly.

**Table 8–1 Class Summary for the Package `oracle.apps.ibe.catalog`**

<b>Class</b>	<b>Description</b>
<a href="#">Class Item</a>	The Item object catches the selling side information for an item in Oracle Inventory. It is the entity that customers can add into the shopping cart from the Web store. The Item object is used to retrieve basic item attributes stored in <code>MTL_SYSTEM_ITEMS_VL</code> (such as part number, description, long description, etc.). It is also used to retrieve the template and multimedia associated with the item for a specific display context, the list of units of measure for the item, the prices defined for the item, whether the item is configurable, the list of related items, the list of related sections, and the item flexfields.
<a href="#">Class ItemFlexfield</a>	The ItemFlexfield object contains the segment information for an item flexfield segment. It is used to retrieve the name, prompt, value, and database column name for an item flexfield segment. This object is returned by the <code>getFlexfields()</code> APIs in the Item class.
<a href="#">Class PriceObject</a>	The PriceObject contains pricing information retrieved for an item. It is used to retrieve list price and selling price. It also provides the functionality to format a price based on currency. This object is returned by the <code>getListAndBestPrices()</code> APIs in the Item class.

**Table 8–1 Class Summary for the Package *oracle.apps.ibe.catalog* (Cont.)**

Class	Description
<a href="#">Class Section</a>	The Section object is the building block of the display hierarchy. The display hierarchy is a tree structure which defines the possible navigational paths for the store. There are two types of sections: FEATURED and NAVIGATIONAL. Featured sections cannot contain subsections and are not displayed in the browse hierarchy of the store. Navigational sections can contain subsections and are displayed in the browse hierarchy of the store. Each section (except the hierarchy root) has one parent section and one or more subsections. A section without subsections is a leaf section. Leaf sections are the only sections that can contain items. The Section object is used to retrieve basic section attributes stored in <code>jtf_dsp_sections_vl</code> (such as display name, description, long description, etc.). It is also used to retrieve the template associated with the section, the multimedia associated with the section for a specific display context, the list of supersections, the list of subsections of a certain type (FEATURED or NAVIGATIONAL), the list of items, the list of sibling sections, and the list of related sections.

## 8.2 Class Item

```
java.lang.Object > oracle.apps.ibe.catalog.Item
```

```
public class Item
```

```
extends Object
```

The Item object stores the selling side attributes for an Oracle Inventory item in `MTL_SYSTEM_ITEMS_VL`. It is also used to retrieve the template and multimedia associated to the item for a specific display context, the list of units of measure for the item, the prices defined for the item, whether the item is configurable, the list of related items, the list of related sections, and the item flexfields.

An Item object is built by using the `load()` APIs, passing in item ID or part number. The item ID typically comes from the leaf section when browsing through the hierarchy. When retrieving item prices, it is recommended that you use the `getListAndBestPrices()` APIs, which return both list price and best price for all UOMs available to an item. If only the price for a particular UOM is needed, you may prefer to use the `getBestPrices()` and `getListPrices()` APIs. For all price APIs, if price list is null (either a null parameter is passed, or the minisite set up for the price list is null), the current user information (party ID and account ID) are used to determine the price list ID.

## 8.2.1 Variables for Class Item

### **PUBLISHED**

```
public static final String PUBLISHED
```

### **SHALLOW**

```
public static final int SHALLOW
```

SHALLOW is a constant passed into Item load APIs to request an Item shallow load, which will load the following item information from Oracle Inventory:

- BOM\_ENABLED\_FLAG
- ORDERABLE\_ON\_WEB\_FLAG
- BACK\_ORDERABLE\_FLAG
- PRIMARY\_UNIT\_OF\_MEASURE
- UNIT\_OF\_MEASURE\_TL
- PRIMARY\_UOM\_CODE
- ITEM\_TYPE
- DESCRIPTION
- LONG\_DESCRIPTION
- BOM\_ITEM\_TYPE
- CONCATENATED\_SEGMENTS (part number)
- INVENTORY\_ITEM\_ID

If other information other than the above is requested from a shallow loaded item, the item will automatically be DEEP loaded, after which all the information will be available.

### **DEEP**

```
public static final int DEEP
```

DEEP is a constant passed into Item load APIs to request an Item deep load, which will load all item attributes.

**MODEL**

```
public static final int MODEL
```

Constant to identify a BOM\_ITEM\_TYPE of Model

**OPTION\_CLASS**

```
public static final int OPTION_CLASS
```

Constant to identify a BOM\_ITEM\_TYPE of Option Class

**8.2.2 Methods for Class Item**

The following table is an index of Class Item methods:

**Table 8–2 Method Index for Class Item**

Method	Description
<a href="#">checkIfValid</a>	Retrieves whether the item with the ID(s) passed in as parameter is a valid item that should be displayed in the Web store
<a href="#">getATPFlag</a>	Retrieves whether ATP check must be performed on the item
<a href="#">getAttributeCategory</a>	Retrieves attribute_category column of the item
<a href="#">getAttributeColumn</a>	Retrieves a column from attribute1-15 in MTL_SYSTEM_ITEMS_B table
<a href="#">getBestPrice</a>	Retrieves the best price for the item, based on UOM codes
<a href="#">getBOMComponentIDs</a>	Retrieves a BOM item's component items from the BOM structure
<a href="#">getBOMComponents</a>	Retrieves a BOM item's component items from the BOM structure based on the user's organization
<a href="#">getBomItemType</a>	Retrieves item's BOM item type
<a href="#">getColumnValue</a>	Retrieves the value of a column from MTL_SYSTEM_ITEMS_VL for this item
<a href="#">getDescription</a>	Retrieves the item's description column based on the user's language
<a href="#">getFixedOrderQty</a>	Retrieves the item's fixed order quantity
<a href="#">getFlexfields</a>	Retrieves the flexfield segments in the flexfield MTL_SYSTEM_ITEMS

**Table 8–2 Method Index for Class Item (Cont.)**

<b>Method</b>	<b>Description</b>
<code>getGlobalAttributeCategory</code>	Retrieves <code>global_attribute_category</code> column of the item
<code>getGlobalAttributeColumn</code>	Retrieves a column from <code>Global_Attribute1-10</code> in <code>MTL_SYSTEM_ITEMS</code>
<code>getItemID</code>	Retrieves item ID
<code>getItemType</code>	Retrieves item's user defined item type
<code>getListAndBestPrices</code>	Retrieves the list and best prices for each UOM of each item in the array passed in as parameter based on the minisite's price list ID
<code>getListPrice</code>	Retrieves the list price of the item for the primary UOM code
<code>getLongDescription</code>	Retrieves the item's long description column
<code>getMaxOrderQty</code>	Retrieves the item's maximum order quantity
<code>getMediaFileName</code>	Retrieves the file name of the physical media associated with this item for a particular display context
<code>getMinOrderQty</code>	Retrieves the item's minimum order quantity
<code>getPartNumber</code>	Retrieves part number
<code>getPrimaryUOM</code>	Retrieves the item's primary UOM, based on the user's language
<code>getPrimaryUOMCode</code>	Retrieves the item's primary UOM code
<code>getRelatedItemIDs</code>	Retrieves the IDs of items related to this item by the relationship code passed in as parameter
<code>getRelatedItems</code>	Retrieves the items related to this item by the relationship code passed in as parameter
<code>getRelatedPrice</code>	Retrieves the price of an item whose price is based on this item's price
<code>getRelatedPrices</code>	Retrieves the prices of items whose price is based on this item's price
<code>getRelatedSectionIDs</code>	Retrieves IDs of sections related to this item by the relationship code passed in as parameter
<code>getSegmentColumn</code>	Retrieves the value of column from <code>Segment1-20</code> in <code>MTL_SYSTEM_ITEMS_VL</code>
<code>getSrcvDuration</code>	Retrieves the default service duration
<code>getSrcvPeriod</code>	Retrieves the item's period for default service duration



**Table 8–2 Method Index for Class Item (Cont.)**

Method	Description
<a href="#">getSrcvStartDelay</a>	Retrieves the number of days after shipment that service begins
<a href="#">getTemplateFileName</a>	Retrieves the file name of the physical template associated with this item for a particular display context
<a href="#">getUOM</a>	Retrieves the translated UOM based on the user's language for the UOM code passed in as parameter
<a href="#">getUOMCodes</a>	If profile IBE: Retrieve All Units of Measure for an Item is set to 'Yes' or does not have a value, retrieves all the UOM codes defined for the item. If profile IBE: Retrieve All Units of Measure for an Item is set to 'No', retrieves the primary UOM code.
<a href="#">getUOMs</a>	Retrieves the UOMs, based on the user's language
<a href="#">isBackOrderable</a>	Retrieves whether the item can be back ordered
<a href="#">isBomEnabled</a>	Retrieves whether the item is BOM enabled
<a href="#">isConfigurable</a>	Retrieves whether the item can be configured
<a href="#">isCouponExempt</a>	Retrieves whether the item is coupon exempt
<a href="#">isDownloadable</a>	Retrieves whether the item is downloadable
<a href="#">isElectronic</a>	Retrieves whether the item is electronic
<a href="#">isOrderable</a>	Retrieves whether the item is orderable via Web
<a href="#">isReturnable</a>	Retrieves whether the item is returnable
<a href="#">isService</a>	Retrieves whether the item is a service item
<a href="#">isServiceable</a>	Retrieves whether the item is serviceable
<a href="#">isShippable</a>	Retrieves whether the item is shippable
<a href="#">isTaxable</a>	Retrieves whether the item is taxable
<a href="#">isVolDiscountExempt</a>	Retrieves whether the item is volume discount exempt
<a href="#">load</a>	Loads the item with the parameters passed in
<a href="#">validateQuantity</a>	Determines whether the input quantity is valid for the item

**checkIfValid**

```
public static boolean checkIfValid(BigDecimal itmid)
throws SQLException, FrameworkException
```

Check if the item with the ID passed in as parameter is a valid item that should be displayed in the Web store. A valid item must be effective, published, and exist in the current inventory validation organization. This API always queries the database.

**Parameters:** itmid (ID of the item to be validated)

**Returns:** boolean - true if the item is valid and should be displayed, false otherwise

### **checkIfValid**

```
public static boolean[] checkIfValid(BigDecimal itmids[])  
throws SQLException, FrameworkException
```

Check if the items with the IDs passed in as parameter are valid items that should be displayed in the Web store. This API always queries the database.

**Parameters:** itmids (IDs of the items to be validated)

**Returns:** boolean[] - array containing whether an item is valid and should be displayed. If itmids[i] is a valid item, boolean[i] is true. Otherwise, boolean[i] is false. Returns boolean[] of size 0 if itmids is null or itmids.length is 0.

### **getATPFlag**

```
public boolean getATPFlag()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether ATP check must be performed on the item

**Returns:** true if ATP check must be performed, false otherwise

### **getAttributeCategory**

```
public String getAttributeCategory()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves the attribute\_category column of the item

**Returns:** attribute category of the item

### **getAttributeColumn**

```
public String getAttributeColumn(int k)  
throws SQLException, FrameworkException, InvalidColumnNumberException,  
ItemNotFoundException
```

Retrieves a column from attribute1-15 in MTL\_SYSTEM\_ITEMS\_B table

**Parameters:** k (int representing which attribute column to return)

**Returns:** attribute value in column MTL\_SYSTEM\_ITEMS\_VL.ATTRIBUTEk

### **getBestPrice**

```
public BigDecimal getBestPrice()  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price for the item, based on primary UOM code. Uses minisite to determine price list ID. If price list ID is null, uses party ID and account ID to determine which price list to use.

**Returns:** item's best price for the primary UOM code and minisite price list ID.

### **getBestPrice**

```
public BigDecimal getBestPrice(String uomCode)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price for the item, based on UOM code passed in as parameter. Uses minisite to determine price list ID. If price list ID is null, uses party ID and account ID to determine which price list to use.

**Parameters:** uomCode (UOM code used to retrieve the price)

**Returns:** item's best price for uomCode passed in as parameter

### **getBestPrice**

```
public BigDecimal getBestPrice(BigDecimal priceListId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price of the item, based on primary UOM code and price list ID passed in as parameter.

**Parameters:** priceListId (price list ID used to retrieve the price)

**Returns:** item's best price for primary UOM code and price list ID passed in as parameter.

### **getBestPrice**

```
public BigDecimal getBestPrice(String uomCode, BigDecimal priceListId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price for the item, based on UOM code and price list ID passed in as parameter.

**Parameters:** uomCode (UOM code used to retrieve the price)

priceListId (price list ID used to retrieve the price)

**Returns:** item's best price for uomCode and price list passed in as parameter

### **getBestPrice**

```
public BigDecimal getBestPrice(BigDecimal partyId, BigDecimal accountId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price of the item based on the customer for the primary UOM code. Minisite's price list ID will be used. If minisite's price list ID is null, uses customer information to determine price list.

**Parameters:** partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's best price based on customer for the primary UOM code

### **getBestPrice**

```
public BigDecimal getBestPrice(BigDecimal priceListId, BigDecimal partyId,
BigDecimal accountId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price of the item based on price list and customer for the primary UOM code. If price list ID is null, uses customer information to determine which price list to use. Otherwise, uses the price list passed in as parameter.

**Parameters:** priceListId (price list ID to use for pricing)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's best price based on price list and customer for the primary UOM code

### **getBestPrice**

```
public BigDecimal getBestPrice(String uomCode, BigDecimal partyId, BigDecimal
accountId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price of the item based on the customer for the UOM code passed in as parameter. Minisite's price list ID will be used. If price list ID is null, Customer information will be used to determine the price list to use.

**Parameters:** uomCode (UOM code used to retrieve the price)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's best price based on customer

### **getBestPrice**

```
public BigDecimal getBestPrice(String uomCode, BigDecimal priceListId,  
BigDecimal partyId, BigDecimal accountId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the best price of the item based on the price list and customer for the UOM code passed in as parameter. If price list ID is null, customer information will be used to determine which price list to use. Otherwise, uses the price list passed in as parameter.

**Parameters:** uomCode (UOM code used to retrieve the price)

priceListId (price list to use for pricing)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's best price based on customer

### **getBOMComponentIDs**

```
public int[] getBOMComponentIDs()  
throws FrameworkException, SQLException
```

If the item is a BOM item, this method retrieves its component items from the BOM structure.

**Returns:** array of item IDs which are the first level components of the current item. Array of size 0 if the current item is not a BOM item or has no active BOM components

### **getBOMComponents**

```
public Item[] getBOMComponents()  
throws SQLException, FrameworkException
```

If the item is a BOM item, this method retrieves its component items from the BOM structure based on the user's organization.

**Returns:** array of items which are the first level components of the current item, array of size 0 if the current item is not a BOM item or has no active BOM components

### **getBomItemType**

```
public int getBomItemType()
```

Retrieves the item's BOM item type

**Returns:** item's BOM item type

### **getColumnValue**

```
public String getColumnValue(String colName)  
throws SQLException, FrameworkException
```

Retrieves the value of a column from MTL\_SYSTEM\_ITEMS\_VL for this item

**Parameters:** colName (name of the column in MTL\_SYSTEM\_ITEMS\_VL whose value will be retrieved)

**Returns:** value of a column from MTL\_SYSTEM\_ITEMS\_VL for this item

### **getDescription**

```
public String getDescription()  
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves the item's description column based on the user's language

**Returns:** display name of the item based on the user's language

### **getFixedOrderQty**

```
public int getFixedOrderQty()  
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves the item's fixed order quantity

**Returns:** item's fixed order quantity; -1 if NULL in MTL\_SYSTEM\_ITEMS

### getFlexfields

```
public ItemFlexfield[] getFlexfields()
throws FrameworkException, SQLException
```

Retrieves the flexfield segments in the flexfield "MTL\_SYSTEM\_ITEMS." Flexfield segment information (name, prompt, value, database column name) can be retrieved from the ItemFlexfield object using the methods getName(), getPrompt(), getValue(), and getDbColumnName().

**Returns:** array of ItemFlexfield containing flexfield segment information.

### getFlexfields

```
public ItemFlexfield[] getFlexfields(String application, String flexfieldName)
throws FrameworkException, SQLException
```

Retrieves the flexfield segments in the flexfield with the application short name and flexfield name passed in as parameter. Flexfield segment information (name, prompt, value, database column name) can be retrieved from the ItemFlexfield object using the methods getName(), getPrompt(), getValue(), and getDbColumnName().

**Parameters:** application (short name of the application module to which the flexfield belongs; for example, "INV")

flexfieldName (name of the flexfield; for example, "MTL\_SYSTEM\_ITEMS")

**Returns:** array of ItemFlexfield containing flexfield segment information.

### getGlobalAttributeCategory

```
public String getGlobalAttributeCategory()
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves global\_attribute\_category column of the the item

**Returns:** global attribute category of the item

### getGlobalAttributeColumn

```
public String getGlobalAttributeColumn(int k)
throws FrameworkException, SQLException, InvalidColumnNumberException,
ItemNotFoundException
```

Retrieves a column from Global\_Attribute1-10 in MTL\_SYSTEM\_ITEMS

**Parameters:** k (int representing which attribute to return)

**Returns:** attribute value in column MTL\_SYSTEM\_ITEMS.GLOBAL\_ATTRIBUTEK

### **getItemID**

```
public int getItemID()
```

Retrieves item ID

**Returns:** the ID of the item

### **getItemType**

```
public String getItemType()
```

Retrieves the item's user defined item type

**Returns:** item's item type

### **getListAndBestPrices**

```
public static Vector[] getListAndBestPrices(Item itms[])  
throws FrameworkException, SQLException, CatalogException
```

Retrieves the list and best prices for each UOM of each item in the array passed in as parameter based on the minisite's price list ID. If there is no price list ID for the minisite, passes party ID and account ID to the pricing engine to determine the price list. Prices will be returned in a Vector[]. The size of the Vector[] will be the same as the size of the Item[] passed in as parameter. Vector[i] will contain a vector of PriceObject for Item[i]. Each PriceObject corresponds to the price based on a UOM code. The PriceObjects in Vector[i] will be ordered in the same order as the UOM codes of Item[i]. To ensure this ordering, iterate through each Vector using elementAt(n) since Enumeration does not guarantee the order in which the elements are returned. The Vector will be trimmed to size to ensure that there will be a PriceObject for each n from 0 to size()-1. To obtain the list and best price call, getListPrice, getBestPrice() on each PriceObject.

**Parameters:** itms (items whose list and best prices will be retrieved)

**Returns:** Vector[] containing list and best prices for the UOMs of the corresponding item. If Item[] is null or the size of Item[] is 0, returns Vector[] of size 0. If Item[] has no UOM codes, Vector[] will be an empty Vector.

### **getListAndBestPrices**

```
public static Vector[] getListAndBestPrices(Item itms[], BigDecimal priceListId)  
throws FrameworkException, SQLException, CatalogException
```



Retrieves the list and best prices for each UOM of each item in the array passed in as parameter, based on the price list ID. If price list ID is not null, retrieves price based only on price list ID and caches the prices. Otherwise, uses party ID and account ID to determine price list. Prices will be returned in a Vector[]. The size of the Vector[] will be the same as the size of the Item[] passed in as parameter. Vector[i] will contain a vector of PriceObject for Item[i]. Each PriceObject corresponds to the price based on a UOM code. The PriceObjects in Vector[i] will be ordered in the same order as the UOM codes of Item[i]. To ensure this ordering, iterate through each Vector using elementAt(n) since Enumeration does not guarantee the order in which the elements are returned. The Vector will be trimmed to size to ensure that there will be PriceObjects for each n from 0 to size()-1. To obtain the list and best price call getListPrice(), getBestPrice() on each PriceObject.

**Parameters:** itms (items whose list and best prices will be retrieved)

priceListId (price list ID used to retrieve price)

**Returns:** Vector[] containing list and best prices for the UOMs of the corresponding item. If Item[] is null or the size of Item[] is 0, returns Vector[] of size 0. If Item[i] has no UOM codes, Vector[i] will be an empty Vector.

### **getListAndBestPrices**

```
public static Vector[] getListAndBestPrices(Item itms[], BigDecimal partyId,
BigDecimal accountId)
throws FrameworkException, SQLException, CatalogException
```

Retrieves the best and list prices for each UOM of each item in the array passed in as parameter, based on the minisite's price list ID and customer. List and best prices will be returned in a Vector[]. The size of the Vector[] will be the same as the size of the Item[] passed in as parameter. Vector[i] will contain a vector of PriceObject for Item[i]. Each PriceObject corresponds to the price based on a UOM code. The PriceObjects in Vector[i] will be ordered in the same order as the UOM codes of Item[i]. To ensure this ordering, iterate through each Vector using elementAt(n) since Enumeration does not guarantee the order in which the elements are returned. The Vector will be trimmed to size to ensure that there will be PriceObjects for each n from 0 to size()-1. To obtain the list price and best price call getListPrice(), getBestPrice() on each PriceObject.

**Parameters:** itms (items whose list and best prices will be retrieved)

priceListId (price list ID used to retrieve price)

**Returns:** Vector[] containing list and best prices for the UOMs of the corresponding item. If Item[] is null or the size of Item[] is 0, returns Vector[] of size 0. If Item[i] has no UOM codes, Vector[i] will be an empty Vector.

### **getListAndBestPrices**

```
public static Vector[] getListAndBestPrices(Item itms[], BigDecimal priceListId,
BigDecimal partyId, BigDecimal accountId)
throws FrameworkException, SQLException, CatalogException
```

Retrieves the best and list prices for each UOM of each item in the array passed in as parameter, based on the price list and customer. If price list ID is null, uses the customer information to determine which price list to use. Otherwise, uses the price list passed in as parameter. List and best prices will be returned in a Vector[]. The size of the Vector[] will be the same as the size of the Item[] passed in as parameter. Vector[i] will contain a vector of PriceObject for Item[i]. Each PriceObject corresponds to the price based on a UOM code. The PriceObjects in Vector[i] will be ordered in the same order as the UOM codes of Item[i]. To ensure this ordering, iterate through each Vector using elementAt(n) since Enumeration does not guarantee the order in which the elements are returned. The Vector will be trimmed to size to ensure that there will be PriceObjects for each n from 0 to size()-1. To obtain the list price and best price call getListPrice(), getBestPrice() on each PriceObject.

**Parameters:** itms (items whose best prices will be retrieved)

priceListId (price list used for pricing)

partyId (customer's party ID)

accountId (customer's account ID)

**Returns:** Vector[] containing list and best prices for the UOMs of the corresponding item. If Item[] is null or the size of Item[] is 0, returns Vector[] of size 0. If Item[i] has no UOM codes, Vector[i] will be an empty Vector.

### **getListPrice**

```
public BigDecimal getListPrice()
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item for the primary UOM code

**Returns:** item's list price for the primary UOM code

**getListPrice**

```
public BigDecimal getListPrice(String uomCode)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item for the UOM code passed in as parameter. Uses minisite to determine price list ID.

**Parameters:** uomCode (UOM code used to retrieve price)

**Returns:** item's list price for the UOM code passed in as parameter

**getListPrice**

```
public BigDecimal getListPrice(BigDecimal priceListId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item for the primary UOM code based on price list ID passed in as parameter

**Parameters:** priceListId (price list ID used to retrieve price)

**Returns:** item's list price for the primary UOM code

**getListPrice**

```
public BigDecimal getListPrice(String uomCode, BigDecimal priceListId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item for the UOM code passed in as parameter. Uses price list ID passed in as parameter.

**Parameters:** uomCode (UOM code used to retrieve price)

priceListId (price list ID used to retrieve price)

**Returns:** item's list price for the UOM code passed in as parameter

**getListPrice**

```
public BigDecimal getListPrice(BigDecimal partyId, BigDecimal accountId)
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item based on the customer for the item's primary uomCode

**Parameters:** partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's list price based on customer

### **getListPrice**

```
public BigDecimal getListPrice(String uomCode, BigDecimal partyId, BigDecimal  
accountId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item based on minisite's price list ID and the customer for the uomCode passed in as parameter

**Parameters:** uomCode (UOM code used to get the price)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's list price based on customer

### **getListPrice**

```
public BigDecimal getListPrice(BigDecimal priceListId, BigDecimal partyId,  
BigDecimal accountId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item based on price list and customer for the item's primary uomCode. If price list ID is null, uses the customer information to determine which price list to use. Otherwise, uses the price list passed in as parameter.

**Parameters:** priceListId (price list ID used for pricing)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's list price based on customer

### **getListPrice**

```
public BigDecimal getListPrice(String uomCode, BigDecimal priceListId,  
BigDecimal partyId, BigDecimal accountId)  
throws SQLException, FrameworkException, PriceNotFoundException
```

Retrieves the list price of the item based on price list and customer for the uomCode passed in as parameter. If price list ID is null, uses the customer information to determine which price list to use. Otherwise, uses the price list passed in as parameter.

**Parameters:** uomCode (UOM code used to get the price)

priceListId (price list used for pricing)

partyId (customer's partyId)

accountId (customer's accountId)

**Returns:** item's list price based on customer

### **getLongDescription**

```
public String getLongDescription()  
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves the item's long description column

**Returns:** longDescription of the item based on the user's language

### **getMaxOrderQty**

```
public int getMaxOrderQty()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves item's maximum order quantity

**Returns:** item's maximum order quantity; -1 if NULL in MTL\_SYSTEM\_ITEMS

### **getMediaFileName**

```
public String getMediaFileName(String dispCtx)  
throws FrameworkException, SQLException
```

Retrieves the file name of the physical media associated with this item for a particular display context

**Parameters:** dispCtx (display context for the media; for example, STORE\_PRODUCT\_SMALL\_IMAGE or STORE\_PRODUCT\_LARGE\_IMAGE)

**Returns:** file name of the physical media. If the required media is not found, returns null.

### **getMinOrderQty**

```
public int getMinOrderQty()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves item's minimum order quantity

**Returns:** item's minimum order quantity; -1 if NULL in MTL\_SYSTEM\_ITEMS

### **getPartNumber**

```
public String getPartNumber()
```

Retrieves part number

**Returns:** the part number of the item

### **getPrimaryUOM**

```
public String getPrimaryUOM()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves the item's primary UOM, based on the user's language

**Returns:** item's UOM based on the user's language

### **getPrimaryUOMCode**

```
public String getPrimaryUOMCode()
```

Retrieves primary UOM code for the item

**Returns:** item's primary UOM code

### **getRelatedItemIDs**

```
public int[] getRelatedItemIDs(String relationshipCode)  
throws SQLException, FrameworkException, ItemNotFoundException, CatalogException
```

Retrieves IDs of items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that are not defined by a SQL rule and that do not require bind arguments.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

**Returns:** IDs of items related to this item by the relationship code passed in as parameter

### **getRelatedItemIDs**

```
public int[] getRelatedItemIDs(String relationshipCode, boolean isSQLRule)  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves IDs of items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that do not require bind arguments.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

**Returns:** IDs of items related to this item by the relationship code passed in as parameter

### **getRelatedItemIDs**

```
public int[] getRelatedItemIDs(String relationshipCode, boolean isSQLRule,  
String bindArgs[])  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves IDs of items related to this item by the relationship code passed in as parameter.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

bindArgs (array of Strings containing bind arguments, used in relationships defined using SQL rules with bind arguments)

**Returns:** IDs of items related to this item by the relationship code passed in as parameter

### **getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode)  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that are not defined by a SQL rule and that do not require bind arguments.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

**Returns:** array of items related to this item by the relationship code passed in as parameter

**getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode, int front, int end)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves subarray of items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that are not defined by a SQL rule and that do not require bind arguments. The subarray will contain items whose position in the related item ID array is between the front and end indexes passed in as parameter. If the end index is larger than the number of related items, returns the items from the front index to the end of the list of items.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

front (beginning index of items to load. Indexing starts at 0)

end (ending index of items to load)

**Returns:** subarray of items related to this item by the relationship code passed in as parameter

**getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode, boolean isSQLRule)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that do not require bind arguments.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

**Returns:** items related to this item by the relationship code passed in as parameter

**getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode, boolean isSQLRule, int
front, int end)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves subarray of items related to this item by the relationship code passed in as parameter. This method should only be used for relationships that do not require bind arguments. The subarray will contain items whose position in the related item



ID array is between the front and end indexes passed in as parameter. If the end index is larger than the number of related items, returns the items from the front index to the end of the list of items.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

front (beginning index of items to load. Indexing starts at 0)

end (ending index of items to load)

**Returns:** subarray of items related to this item by the relationship code passed in as parameter

### **getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode, boolean isSQLRule, String bindArgs[])  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves items related to this item by the relationship code passed in as parameter.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

bindArgs (array of Strings containing bind arguments, used in relationships defined using SQL rules with bind arguments)

**Returns:** items related to this item by the relationship code passed in as parameter

### **getRelatedItems**

```
public Item[] getRelatedItems(String relationshipCode, boolean isSQLRule, String bindArgs[], int front, int end)  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Retrieves subarray of items related to this item by the relationship code passed in as parameter. The subarray will contain items whose position in the related item ID array is between the front and end indexes passed in as parameter. If the end index is larger than the number of related items, returns the items from the front index to the end of the list of items.

**Parameters:** relationshipCode (specifies the type of relationship; for example, "SUBSTITUTE")

isSQLRule (whether the relationship is defined by a SQL rule)

bindArgs (array of Strings containing bind arguments, used in relationships defined using SQL rules with bind arguments)

front (beginning index of items to load. Indexing starts at 0)

end (ending index of items to load)

**Returns:** subarray of items related to this item by the relationship code passed in as parameter

### **getRelatedPrice**

```
public PriceObject getRelatedPrice(Item itm)
throws FrameworkException, SQLException, CatalogException
```

Retrieves price of an item whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the minisite's price list ID and the primary UOM codes of the items. If minisite price list ID is null, passes party ID and account ID to pricing engine to determine the price list. Returns PriceObject containing the price of the item passed in as parameter. Call getListPrice(), getBestPrice() on the PriceObject returned to get the price of the item whose price depends on this item.

**Parameters:** itm (item whose price depends on the price of this base item)

**Returns:** PriceObject (PriceObject containing the price of the item passed in as parameter)

### **getRelatedPrice**

```
public PriceObject getRelatedPrice(Item itm, BigDecimal priceListId)
throws FrameworkException, SQLException, CatalogException
```

Retrieves price of an item whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the price list ID passed in and the primary UOM codes of the items. If price list ID is null, passes party ID and account ID to pricing engine to determine the price list. Returns PriceObject containing the price of the item passed in as parameter. Call getListPrice(), getBestPrice() on the PriceObject returned to get the price of the item whose price depends on this item.

**Parameters:** itm (item whose price depends on the price of this base item)

priceListId (price list ID)

**Returns:** PriceObject (PriceObject containing the price of the item passed in as parameter)

### **getRelatedPrice**

```
public PriceObject getRelatedPrice(Item itm, BigDecimal partyId, BigDecimal  
accountId)  
throws FrameworkException, SQLException, CatalogException
```

Retrieves price of an item whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the minisite price list, partyId, accountId, and the primary UOM codes. Returns PriceObject containing the price of the item passed in as parameter. Call getListPrice(), getBestPrice() on the PriceObject returned to get the price of the item whose price depends on this item.

**Parameters:** itm (item whose price depends on the price of this base item)

partyId (customer's party ID)

accountId (customer's account ID)

**Returns:** PriceObject (PriceObject containing the price of the item passed in as parameter)

### **getRelatedPrice**

```
public PriceObject getRelatedPrice(Item itm, BigDecimal priceListId, BigDecimal  
partyId, BigDecimal accountId)  
throws FrameworkException, SQLException, CatalogException
```

Retrieves price of an item whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the priceListId, partyId, accountId, and the primary UOM codes. If priceListId is null, partyId and accountId will be used to determine which price list to use for pricing. Otherwise, uses the price list passed in as parameter. Returns PriceObject containing the price of the item passed in as parameter. Call getListPrice(), getBestPrice() on the PriceObject returned to get the price of the item whose price depends on this item.

**Parameters:** itm (item whose price depends on the price of this base item)

priceListId (price list to use for pricing)

partyId (customer's party ID)

accountId (customer's account ID)

**Returns:** PriceObject (PriceObject containing the price of the item passed in as parameter)

### **getRelatedPrices**

```
public PriceObject[] getRelatedPrices(Item itms[])  
throws FrameworkException, SQLException, CatalogException
```

Retrieves prices of items whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the minisite's price list ID and the primary UOM codes of the items. If minisite's price list is null, uses party ID and account ID to determine which price list to use. Returns PriceObject[] the size of the Item[] passed in as parameter. The price contained in PriceObject[i] corresponds to the price of Item[i]. To retrieve the price of Item[i], call PriceObject[i].getListPrice() or PriceObject[i].getBestPrice().

**Parameters:** itms (array of items whose prices depend on the price of this base item)

**Returns:** PriceObject[] (array of PriceObject containing the price of each item in the item array)

### **getRelatedPrices**

```
public PriceObject[] getRelatedPrices(Item itms[], BigDecimal priceListId)  
throws FrameworkException, SQLException, CatalogException
```

Retrieves prices of items whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the price list ID passed in and the primary UOM codes of the items. If price list ID is null, uses party ID and account ID to determine which price list to use. Returns PriceObject[] the size of the Item[] passed in as parameter. The price contained in PriceObject[i] corresponds to the price of Item[i]. To retrieve the price of Item[i], call PriceObject[i].getListPrice() or PriceObject[i].getBestPrice().

**Parameters:** itms (array of items whose prices depend on the price of this base item)

priceListId (price list ID)

**Returns:** PriceObject[] (array of PriceObject containing the price of each item in the item array)

### **getRelatedPrices**

```
public PriceObject[] getRelatedPrices(Item itms[], BigDecimal partyId,  
BigDecimal accountId)
```

---

throws `FrameworkException`, `SQLException`, `CatalogException`

Retrieves prices of items whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the `partyId`, `accountId`, and the primary UOM codes of the items. Returns `PriceObject[]` the size of the `Item[]` passed in as parameter. The price contained in `PriceObject[i]` corresponds to the price of `Item[i]`. To retrieve the price of `Item[i]`, call `PriceObject[i].getListPrice()` or `PriceObject[i].getBestPrice()`.

**Parameters:** `itms` (array of items whose prices depend on the price of this base item)

`partyId` (customer's party ID)

`accountId` (customer's account ID)

**Returns:** `PriceObject[]` (array of `PriceObject` containing the price of each item in the item array)

### **getRelatedPrices**

```
public PriceObject[] getRelatedPrices(Item itms[], BigDecimal priceListId,  
BigDecimal partyId, BigDecimal accountId)  
throws FrameworkException, SQLException, CatalogException
```

Retrieves prices of items whose price is based on this item's price. This API can be used to retrieve the price of items whose price depends on the price of another item. Prices retrieved are based on the `priceListId`, `partyId`, `accountId`, and the primary UOM codes of the items. If `priceListId` is null, uses the `partyId` and `accountId` to determine which price list to use for pricing. Otherwise, uses the price list passed in as parameter. Returns `PriceObject[]` the size of the `Item[]` passed in as parameter. The price contained in `PriceObject[i]` corresponds to the price of `Item[i]`. To retrieve the price of `Item[i]`, call `PriceObject[i].getListPrice()` or `PriceObject[i].getBestPrice()`.

**Parameters:** `itms` (array of items whose prices depend on the price of this base item)

`priceListId` (price list used for pricing)

`partyId` (customer's party ID)

`accountId` (customer's account ID)

**Returns:** `PriceObject[]` (array of `PriceObject` containing the price of each item in the item array)

### **getRelatedSectionIDs**

```
public int[] getRelatedSectionIDs(String relationshipCode)
```

throws `SQLException`, `FrameworkException`, `SectionNotFoundException`

Retrieves IDs of sections related to this item by the relationship code passed in as parameter

**Parameters:** `relationshipCode` (specifies the type of relationship; for example "SUBSTITUTE")

**Returns:** IDs of sections related to this item by the relationship code passed in as parameter

### **getSegmentColumn**

```
public String getSegmentColumn(int k)
throws InvalidColumnNumberException, FrameworkException, SQLException,
ItemNotFoundException
```

Retrieves the value of column from Segment1-20 in `MTL_SYSTEM_ITEMS_VL`

**Parameters:** `k` (int representing which segment to return)

**Returns:** segment value in column `MTL_SYSTEM_ITEMS.SEGMENTk`

### **getSrvcDuration**

```
public int getSrvcDuration()
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves default service duration

**Returns:** default service duration

### **getSrvcPeriod**

```
public String getSrvcPeriod()
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves item's period for default service duration

**Returns:** period for default service duration

### **getSrvcStartDelay**

```
public int getSrvcStartDelay()
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves number of days after shipment that service begins

**Returns:** days after shipment that service begins

### **getTemplateFileName**

```
public String getTemplateFileName(int dispCtxID)
throws FrameworkException, SQLException
```

Retrieves the file name of the physical template associated with this item for a particular display context.

**Parameters:** dispCtx (display context ID for the template)

**Returns:** file name of physical template. If the required template is not found, returns null.

### **getTemplateFileName**

```
public String getTemplateFileName(String displayContext)
throws FrameworkException, SQLException
```

Retrieves the file name of the physical template associated with this item for a particular display context.

**Parameters:** displayContext (display context for the template; for example, STORE\_PRODUCT\_DETAILS, STORE\_PRODUCT\_DESCR)

**Returns:** file name of the physical template. If the required template is not found, returns null.

### **getUOM**

```
public String getUOM(String uomCode)
throws SQLException, FrameworkException, ItemNotFoundException
```

Retrieves translated UOM based on the user's language for the UOM code passed in as parameter. Returns null if the UOM code is not in the item's list of UOM codes.

**Parameters:** uomCode (UOM codes used to get the translated UOM)

**Returns:** translated UOM based on the user's language

### **getUOMCodes**

```
public String[] getUOMCodes()
```

If profile IBE: Retrieve All Units of Measure for an Item is set to 'Yes' or does not have a value, retrieves all the UOM codes defined for the item. If profile IBE:

Retrieve All Units of Measure for an Item is set to 'No', retrieves the primary UOM code.

**Returns:** item's UOM codes

### **getUOMs**

```
public String[] getUOMs()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves units of measure, based on the user's language

**Returns:** String[ ] containing item's UOMs based on the user's language. String[ ] of size 0 if no units of measures were found. For UOMs where a translated version cannot be found for the user's language, an empty string will be returned.

### **isBackOrderable**

```
public boolean isBackOrderable()
```

Retrieves whether item can be back ordered

**Returns:** true if item can be back ordered, false otherwise

### **isBomEnabled**

```
public boolean isBomEnabled()
```

Retrieves whether item is BOM enabled

**Returns:** true if item is BOM enabled, false otherwise

### **isConfigurable**

```
public boolean isConfigurable()
```

Retrieves whether item can be configured

**Returns:** true if item can be configured, false otherwise

### **isCouponExempt**

```
public boolean isCouponExempt()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is coupon exempt

**Returns:** true if item is coupon exempt, false otherwise



**isDownloadable**

```
public boolean isDownloadable()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is downloadable

**Returns:** true if item is downloadable, false otherwise

**isElectronic**

```
public boolean isElectronic()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is electronic

**Returns:** true if item is electronic, false otherwise

**isOrderable**

```
public boolean isOrderable()
```

Retrieves whether item is orderable via Web

**Returns:** true if item is orderable on the Web, false otherwise

**isReturnable**

```
public boolean isReturnable()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is returnable

**Returns:** true if item is returnable, false otherwise

**isService**

```
public boolean isService()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is a service item

**Returns:** true if item is a service item, false otherwise

**isServiceable**

```
public boolean isServiceable()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is serviceable

**Returns:** true if item is serviceable, false otherwise

### **isShippable**

```
public boolean isShippable()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is shippable

**Returns:** true if item is shippable, false otherwise

### **isTaxable**

```
public boolean isTaxable()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is taxable

**Returns:** true if item is taxable, false otherwise

### **isVolDiscountExempt**

```
public boolean isVolDiscountExempt()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves whether item is volume discount exempt

**Returns:** true if item is volume discount exempt, false otherwise

### **load**

```
public static Item load(int itemID)  
throws SQLException, FrameworkException, ItemNotFoundException
```

Loads the item with the inventory\_item\_id passed in as parameter. Load level of the item will be SHALLOW. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for the item's primary UOM. Otherwise, does not load the price.

**Parameters:** itemID (item ID corresponding to MTL\_SYSTEM\_ITEMS\_VL.INVENTORY\_ITEM\_ID)

**Returns:** item with values loaded for the proper members

**load**

```
public static Item load(int itemID, int mode)
throws SQLException, FrameworkException, ItemNotFoundException
```

Loads the item with the `inventory_item_id` passed in as parameter. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for the item's primary UOM. Otherwise, does not load the price.

**Parameters:** `itemID` (item ID corresponding to `MTL_SYSTEM_ITEMS_VL.INVENTORY_ITEM_ID`)

`mode` (load level for the item; `Item.SHALLOW` or `Item.DEEP`)

**Returns:** item with values loaded for the proper members

**load**

```
public static Item load(int itemID, int mode, boolean retrievePrice)
throws SQLException, FrameworkException, ItemNotFoundException
```

Loads the item with the `inventory_item_id` and `mode` passed in as parameter. Uses the `retrievePrice` parameter to determine whether to retrieve the price for the primary UOM of the item.

**Parameters:** `itemID` (item ID corresponding to `MTL_SYSTEM_ITEMS_VL.INVENTORY_ITEM_ID`)

`mode` (load level for the item; `Item.SHALLOW` or `Item.DEEP`)

`retrievePrice` (whether to retrieve the price for the item's primary UOM)

**Returns:** item with values loaded for the proper members

**load**

```
public static Item load(String partNum)
throws SQLException, FrameworkException, ItemNotFoundException,
CatalogException
```

Loads the item with the part number passed in as parameter. Load level of the item will be `SHALLOW`. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for the item's primary UOM. Otherwise, does not load the price.

**Parameters:** `partNum` (item part number corresponding to `MTL_SYSTEM_ITEMS_VL.CONCATENATED_SEGMENTS`)

**Returns:** item with values loaded for the proper members

### load

```
public static Item load(String partNum, int mode)
throws SQLException, FrameworkException, ItemNotFoundException, CatalogException
```

Loads the item with the part number and mode passed in as parameter. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for the item's primary UOM. Otherwise, does not load the price.

**Parameters:** partNum (item part number corresponding to MTL\_SYSTEM\_ITEMS\_VL.CONCATENATED\_SEGMENTS)

mode (load level for the item; Item.SHALLOW or Item.DEEP)

**Returns:** item with values loaded for the proper members

### load

```
public static Item load(String partNum, int mode, boolean retrievePrice)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads the item with the part number and mode passed in as parameter. Uses the retrievePrice parameter to determine whether to retrieve the price for the primary UOM of the item.

**Parameters:** partNum (item part number corresponding to MTL\_SYSTEM\_ITEMS\_VL.CONCATENATED\_SEGMENTS)

mode (load level for the item; Item.SHALLOW or Item.DEEP)

retrievePrice (whether to retrieve the price for the item's primary UOM)

**Returns:** item with values loaded for the proper members

### load

```
public static Item[] load(int itemIDs[])
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads the items with the item IDs passed in as parameter. The load level of the items will be SHALLOW. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for each item's primary UOM. Otherwise, does not load the price. The order of the items will be in the order of the IDs passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** itemIDs (item IDs corresponding to MTL\_SYSTEM\_ITEMS\_VL.INVENTORY\_ITEM\_ID)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(int itemIDs[], int mode)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads items with the array of item IDs and mode passed in as parameter. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for each item's primary UOM. Otherwise, does not load the price. The order of the items will be in the order of the IDs passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** itemIDs (array of item IDs corresponding to mtl\_sysetm\_items\_vl.inventory\_item\_id)

mode (load level for the item; Item.SHALLOW or Item.DEEP)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(int itemIDs[], int mode, boolean retrievePrice)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Load the items with the item IDs and mode passed in as parameter. Uses the retrievePrice parameter to determine whether to retrieve the price for each item's primary UOM. The order of the items will be in the order of the IDs passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** itemIDs (array of item IDs corresponding to MTL\_SYSTEM\_ITEMS\_VL.INVENTORY\_ITEM\_ID)

mode (load level for the item; Item.SHALLOW or Item.DEEP)

retrievePrice (whether to retrieve the prices for the item's primary UOM)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(int itemIDs[], int mode, boolean retrievePrice,
boolean compact)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads the items with the item IDs and mode passed in as parameter. Uses the retrievePrice parameter to determine whether to retrieve the price for each item's primary UOM. The order of the items will be in the order of the IDs passed in. If compact is true, removes items that were not found in the database. If compact is false, returns a null for items not found in the database.

**Parameters:** itemIDs (array of item IDs corresponding to MTL\_SYSTEM\_ITEMS\_VL,INVENTORY\_ITEM\_ID)

mode (load level for the item; Item.SHALLOW or Item.DEEP)

retrievePrice (whether to retrieve the price for the item's primary UOM)

compact (whether to remove items not found in the database)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(String partNums[])  
throws SQLException, FrameworkException, ItemNotFoundException, CatalogException
```

Loads the items with the part numbers passed in as parameter. The load level of the items will be SHALLOW. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for each item's primary UOM. Otherwise, does not load the price. The order of the items will be in the order of the part numbers passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** partNums (array of item part numbers corresponding to MTL\_SYSTEM\_ITEMS\_VL.CONCATENATED\_SEGMENTS)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(String partNums[], int mode)  
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads the items with the part numbers and mode passed in as parameter. If the profile IBE: Retrieve Price When Displaying Items is set to 'Yes', loads the price for each item's primary UOM. Otherwise, does not load the price. The order of the items will be in the order of the part numbers passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** partNums (array of item part numbers corresponding to mtl\_system\_item\_vl.concatenated\_segments)

mode (load level for the items; Item.SHALLOW or Item.DEEP)

**Returns:** array of items with values loaded for the proper members

### load

```
public static Item[] load(String partNums[], int mode, boolean retrievePrice)
throws SQLException, FrameworkException, CatalogException, ItemNotFoundException
```

Loads the items with the part numbers and mode passed in as parameter. Uses the parameter retrievePrice to determine whether to retrieve the price for each item's primary UOM. The order of the items will be in the order of the part numbers passed in. Duplicates and items not found in the database will be removed from the Item array.

**Parameters:** partNums (array of item part numbers corresponding to MTL\_SYSTEM\_ITEMS\_VL.CONCATENATED\_SEGMENTS)

mode (load level for the items; Item.SHALLOW or Item.DEEP)

retrievePrice (whether to retrieve the price for each item's primary UOM)

**Returns:** array of items with values loaded for the proper members

### validateQuantity

```
public static boolean validateQuantity(int itemId, BigDecimal organizationId,
String inputQty, String uomCode)
throws FrameworkException, SQLException, ItemNotFoundException
```

Determines whether the input quantity is valid for the item. If the item is OM: Indivisible or serial code controlled, the quantity must be an integer when converted to the primary UOM. Otherwise, the quantity can be a decimal value.

**Parameters:** itemId (item ID corresponding to MTL\_SYSTEM\_ITEMS\_VL.INVENTORY\_ITEM\_ID)

organizationId (inventory organization ID corresponding to MTL\_SYSTEM\_ITEMS\_VL.ORGANIZATION\_ID)

inputQty (input quantity to validate)

uomCode (UOM code)

**Returns:** true if the input quantity is valid for the item and UOM, false otherwise

**validateQuantity**

```
public static boolean[] validateQuantity(int itemIds[], BigDecimal
organizationId[], String inputQty[], String uomCodes[])
throws FrameworkException, SQLException, ItemNotFoundException
```

For each item, determines whether the input quantity is valid. If an item is OM: Indivisible or serial code controlled, the quantity must be an integer when converted to the primary UOM. Otherwise, the quantity can be a decimal value.

**Parameters:** itemIds (item IDs corresponding to MTL\_SYSTEM\_ITEMS\_VL.INVENTORY\_ITEM\_ID)

organizationId (inventory organization IDs corresponding to MTL\_SYSTEM\_ITEMS\_VL.ORGANIZATION\_ID)

inputQty (input quantity to validate)

uomCodes (UOM code)

**Returns:** boolean array indicating whether each input quantity is valid for the given item and UOM. If inputQty[i] is valid for itemIds[i], organizationId[i], and uomCodes[i], the value of boolean[i] is true. Otherwise, the value of boolean[i] is false.

## 8.3 Class ItemFlexfield

```
java.lang.Object > oracle.apps.ibe.catalog.ItemFlexfield
public class ItemFlexfield
```

extends Object

The ItemFlexfield object contains the segment information for an item flexfield segment. It is used to retrieve the name, prompt, value, and database column name for an item flexfield segment. This object is returned by the getFlexfields() methods in the Item class.

### 8.3.1 Methods for Class ItemFlexfield

The following table is an index of Class ItemFlexfield methods:

**Table 8–3 Method Index for Class ItemFlexfield**

Method	Description
<a href="#">getDbColumnName</a>	Retrieves the database column name of the flexfield segment



**Table 8–3 Method Index for Class ItemFlexfield (Cont.)**

Method	Description
<a href="#">getName</a>	Retrieves the name of the flexfield segment
<a href="#">getPrompt</a>	Retrieves the prompt of the flexfield segment
<a href="#">getValue</a>	Retrieves the value of the flexfield segment

**getDbColumnName**

```
public String getDbColumnName()
```

Retrieves the database column name of the flexfield segment

**Returns:** database column name of the flexfield segment

**getName**

```
public String getName()
```

Retrieves the name of the flexfield segment

**Returns:** name of the flexfield segment

**getPrompt**

```
public String getPrompt()
```

Retrieves the prompt of the flexfield segment

**Returns:** prompt of the flexfield segment

**getValue**

```
public String getValue()
```

Retrieves the value of the flexfield segment

**Returns:** value of the flexfield segment

## 8.4 Class PriceObject

```
java.lang.Object > oracle.apps.ibe.catalog.PriceObject
public class PriceObject
extends Object
```

The PriceObject contains pricing information retrieved for an item. It is used to retrieve list price and selling price. It also provides the functionality to format a price based on currency. This object is returned by the getListAndBestPrices() APIs in the Item class.

## 8.4.1 Methods for Class PriceObject

The following table is an index of Class PriceObject methods:

**Table 8–4 Method Index for Class PriceObject**

Method	Description
<a href="#">formatNumber</a>	Formats the price based on currency code
<a href="#">getBestPrice</a>	Retrieves the best price stored in the PriceObject
<a href="#">getListPrice</a>	Retrieves the list price stored in the PriceObject

### **formatNumber**

```
public static String formatNumber(String currencyCode, BigDecimal number)
throws FrameworkException, SQLException
```

Formats the price based on currency code. Prepends currency symbol to the front of the number, adds the appropriate decimal and grouping separators. If currencyCode is null, uses Java's default formatting. If number is null, returns an empty string.

**Parameters:** currencyCode (currency code used to format the price)  
number (price to be formatted)

**Returns:** string containing price formatted based on currency code

### **formatNumber**

```
public static String formatNumber(String currencyCode, double number)
throws FrameworkException, SQLException
```

Format the price based on currency code. Prepends currency symbol to the front of the number, adds the appropriate decimal and grouping separators. If currencyCode is null, uses Java's default formatting.

**Parameters:** currencyCode (currency code used to format the price)  
number (price to be formatted)

**Returns:** string containing price formatted based on currency code

### **getBestPrice**

```
public BigDecimal getBestPrice()  
throws PriceNotFoundException
```

Retrieves the best price stored in the PriceObject

**Returns:** best price store in the PriceObject

### **getListPrice**

```
public BigDecimal getListPrice()  
throws PriceNotFoundException
```

Retrieves the list price stored in the PriceObject

**Returns:** list price stored in the PriceObject

## 8.5 Class Section

```
java.lang.Object > oracle.apps.ibe.catalog.Section  
public class Section
```

extends Object

The Section object is the building block of the display hierarchy. There are two types of sections: FEATURED and NAVIGATIONAL. Featured sections cannot contain subsections and are not displayed in the browse hierarchy of the store. Navigational sections can contain subsections and are displayed in the browse hierarchy of the store. Each section (except the hierarchy root) has one parent section and one or more subsections. A section with subsections is a non-leaf section. A section without subsections is a leaf section. Leaf sections are the only sections that can contain items.

The Section object is used to retrieve basic section attributes stored in jtf\_dsp\_sections\_v1 (such as display name, description, long description, etc.). It is also used to retrieve the template associated to the section, the media associated to the section for a specific display context, the list of supersections, the list of subsections of a certain type (FEATURED or NAVIGATIONAL), the list of items, the list of sibling sections, and the list of related sections.

## 8.5.1 Variables for Class Section

### SHALLOW

```
public static final int SHALLOW
```

SHALLOW is a constant passed into Section load APIs to request a Section shallow load, which will load the following Section attributes SECTION\_ID, ACCESS\_NAME, DISPLAY\_NAME, DESCRIPTION, OBJECT\_VERSION\_NUMBER, SECTION\_TYPE\_CODE. If information other than the above is requested from a shallow loaded section, the section will automatically be DEEP loaded, after which all the information will be available.

### DEEP

```
public static final int DEEP
```

DEEP is a constant passed into Section load APIs to request a Section deep load, which will load all section attributes, subsections IDs, item IDs, supersection IDs.

### NAVIGATIONAL

```
public static final String NAVIGATIONAL
```

Constant for Navigational section type: those sections that can be browsed through navigation. Out of the box, iStore has NAVIGATIONAL and FEATURED section types. The merchant can define new section types.

### FEATURED

```
public static final String FEATURED
```

Constant for Featured section type: those sections that are not browsed through navigation.

## 8.5.2 Methods for Class Section

The following table is an index of Class Section methods:

**Table 8–5 Method Index for Class Section**

Method	Description
<a href="#">getAccessName</a>	Retrieves section access name
<a href="#">getAttributeCategory</a>	Retrieves section's attribute_category column

**Table 8–5 Method Index for Class Section (Cont.)**

<b>Method</b>	<b>Description</b>
<a href="#">getAttributeColumn</a>	Retrieves an attribute column value of the section
<a href="#">getDescription</a>	Retrieves description of the section in the current language
<a href="#">getDisplayContextID</a>	Retrieves the display context for items in the section. This method is only applicable to leaf sections.
<a href="#">getDisplayName</a>	Retrieves display name of the section in the current language
<a href="#">getFeaturedItemIDs</a>	Retrieves an array containing the IDs of the items in the section's featured subsections
<a href="#">getFeaturedItems</a>	Retrieves an array containing the items in the section's featured subsections
<a href="#">getFeaturedSubSectionIDs</a>	Retrieves an array of featured subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.
<a href="#">getFeaturedSubSections</a>	Retrieves an array of featured subsections
<a href="#">getItemIDs</a>	Retrieves array of item IDs in current section for the user's validation organization ID. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.
<a href="#">getItems</a>	Retrieves an array of items in current section. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.
<a href="#">getLeafSubSectionIDs</a>	Retrieves an array of leaf level descendent subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.
<a href="#">getLeafSubSections</a>	Retrieves an array of leaf level descendent subsections
<a href="#">getLongDescription</a>	Retrieves long description of the section in the current language
<a href="#">getMediaFileName</a>	Retrieves the file name of the physical media associated with the section for a particular display context
<a href="#">getNonNavSubSectionIDs</a>	Retrieves an array of all non-navigational subsection IDs
<a href="#">getNonNavSubSections</a>	Retrieves an array of non-navigational subsections
<a href="#">getNumberOfItems</a>	Retrieves the number of item IDs within the section available in the user's validation organization. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Table 8–5 Method Index for Class Section (Cont.)**

<b>Method</b>	<b>Description</b>
<a href="#">getRelatedSectionIDs</a>	Retrieves an array of section IDs for all sections related to this section by the relationship code passed in as parameter
<a href="#">getRelatedSections</a>	Retrieves an array of sections related to this section by the relationship code passed in as parameter
<a href="#">getSectionID</a>	Retrieves section ID
<a href="#">getSectionType</a>	Retrieves section type
<a href="#">getSiblingSectionIDs</a>	Retrieves an array of section IDs containing the IDs of the same level siblings
<a href="#">getSiblingSections</a>	Retrieves an array of sections containing the same level siblings
<a href="#">getSubSectionIDs</a>	Retrieves an array of navigational subsection IDs
<a href="#">getSubSectionItemIDs</a>	Retrieves an array containing the IDs of items in subsections with the specified section type
<a href="#">getSubSectionItems</a>	Retrieves an array containing items in a subsection with the specified section type
<a href="#">getSubSections</a>	Retrieves an array of navigational subsections
<a href="#">getSuperSection</a>	Retrieves the immediate super section
<a href="#">getSuperSectionID</a>	Retrieves ID of the immediate super section
<a href="#">getSuperSectionIDs</a>	Retrieves an array of supersection IDs, starting with the minisite root section
<a href="#">getSuperSections</a>	Retrieves an array of supersections, starting with the minisite root section
<a href="#">getTemplateFileName</a>	Retrieves the file name of the physical template associated with the section
<a href="#">isFeatured</a>	Determines whether the section is a featured section
<a href="#">isLeafSection</a>	Determines whether this section is a leaf section
<a href="#">load</a>	Loads a section with the parameters passed in

**getAccessName**

```
public String getAccessName()
```

Retrieves section access name

**Returns:** access name of the section

### **getAttributeCategory**

```
public String getAttributeCategory()  
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves section's attribute\_category column

**Returns:** section's attribute category

### **getAttributeColumn**

```
public String getAttributeColumn(int k)  
throws FrameworkException, SQLException, SectionNotFoundException,  
InvalidColumnNumberException
```

Retrieves an attribute column value of the section

**Parameters:** k (the attribute column number, 1 to 15)

**Returns:** attribute value in column JTF\_DSP\_SECTIONS\_VL.ATTRIBUTEK

### **getDescription**

```
public String getDescription()  
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves description of the section in the current language

**Returns:** section description

### **getDisplayContextID**

```
public int getDisplayContextID()  
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves the display context for items in the section. This method is only applicable to leaf sections.

**Returns:** display context ID defined for this section. If jtf\_dsp\_sections\_vl.display\_context\_id is null, returns -1.

### **getDisplayName**

```
public String getDisplayName()  
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves display name of the section in the current language

**Returns:** section display name

### **getFeaturedItemIDs**

```
public int[] getFeaturedItemIDs()  
throws FrameworkException, SQLException, SectionNotFoundException,  
ItemNotFoundException
```

Retrieves an array containing the IDs of the items in the section's featured subsections.

**Returns:** array of item IDs

### **getFeaturedItems**

```
public Item[] getFeaturedItems()  
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves an array containing the items in the section's featured subsections.

**Returns:** array of items

### **getFeaturedSubSectionIDs**

```
public int[] getFeaturedSubSectionIDs()  
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of featured subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of featured subsection IDs

### **getFeaturedSubSections**

```
public Section[] getFeaturedSubSections()  
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of featured subsections. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of featured subsections

### **getItemIDs**

```
public int[] getItemIDs()
```



throws `FrameworkException`, `SQLException`, `ItemNotFoundException`

Retrieves array of item IDs in current section for the user's validation organization ID. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of item IDs

### **getItemIDs**

```
public int[] getItemIDs(String orderByClause)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves an array of item IDs ordered using the `orderByClause` parameter.

**Parameters:** `orderByClause` (comma-separated list of columns in `MTL_SYSTEM_ITEMS_VL` that will be used to order the items; for example "description asc, inventory\_item\_id, concatenated\_segments desc")

**Returns:** array of item IDs ordered by the `orderByClause` passed in as parameter

### **getItems**

```
public Item[] getItems()
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves an array of items in current section. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of items

### **getItems**

```
public Item[] getItems(int front)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves a subarray of items in current section. The subarray will start with the item whose position in the item ID array is the `front` index passed in as parameter. It will end after retrieving number of items shown on a page (profile IBE: Items Per Page for Display).

**Parameters:** `front` (index indicating the position in the item ID array at which to start retrieving items; indexing starts at 0)

**Returns:** subarray of items

**getItems**

```
public Item[] getItems(int front, int end)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves a subarray of items in current section. The subarray will contain items whose position in the item ID array is between the front and end indexes passed in as parameter.

**Parameters:** front (index indicating the position in the item ID array at which to start retrieving items; indexing starts at 0)

end (index indicating the position in the item ID array at which to stop retrieving items)

**Returns:** subarray of items

**getItems**

```
public Item[] getItems(String orderByClause)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves an array of items in current section, ordered by the orderByClause parameter. This method will always go to the database to retrieve the item IDs.

**Parameters:** orderByClause (comma-separated list of columns in MTL\_SYSTEM\_ITEMS\_VL used to order the items; for example, "description asc, inventory\_item\_id, concatenated\_segments desc")

**Returns:** array of items, ordered by the orderByClause parameter

**getItems**

```
public Item[] getItems(String orderByClause, int front)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves a subarray of items, ordered by the orderByClause parameter. The subarray will start with the item whose position in the item ID array is the front index passed in as parameter. It will end after retrieving number of items shown on a page (profile IBE: Items Per Page for Display). This method will always go to the database to retrieve the item IDs.

**Parameters:** orderByClause (comma-separated list of columns in MTL\_SYSTEM\_ITEMS\_VL used to order the items; for example, "description asc, inventory\_item\_id, concatenated\_segments desc")

front (index indicating the position at which to start retrieving items; indexing starts at 0)

**Returns:** subarray of items, ordered by the orderByClause parameter

### getItems

```
public Item[] getItems(String orderByClause, int front, int end)
throws FrameworkException, SQLException, ItemNotFoundException
```

Retrieves a subarray of items, ordered by the orderByClause parameter. The subarray will contain items whose position in the array is between the front and end indexes passed in as parameter. This method will always go to the database to retrieve the item IDs.

**Parameters:** orderByClause (comma-separated list of columns in MTL\_SYSTEM\_ITEMS\_VL used to order the items; for example, "description asc, inventory\_item\_id, concatenated\_segments desc")

front (index indicating the position at which to start retrieving items; indexing starts at 0)

end (index indicating the position in the item ID array at which to stop retrieving items)

**Returns:** subarray of items, ordered by the orderByClause parameter

### getLeafSubSectionIDs

```
public int[] getLeafSubSectionIDs()
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of leaf level descendent subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of leaf level descendent subsection IDs

### getLeafSubSections

```
public Section[] getLeafSubSections()
throws FrameworkException, SQLException, SectionNotFoundException,
CatalogException
```

Retrieves an array of leaf level descendent subsections. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of leaf level descendent subsections

### **getLongDescription**

```
public String getLongDescription()  
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves long description of the section in the current language

**Returns:** section long description

### **getMediaFileName**

```
public String getMediaFileName(String mediaCtx)  
throws FrameworkException, SQLException
```

Retrieves the file name of the physical media associated with the section for a particular display context

**Parameters:** mediaCtx (display context for the media file; for example, STORE\_SECTION\_SMALL\_IMAGE)

**Returns:** File name of the media. If the required media is not found, returns null.

### **getNonNavSubSectionIDs**

```
public int[] getNonNavSubSectionIDs()  
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of all non-navigational subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of non-navigational subsection IDs

### **getNonNavSubSections**

```
public Section[] getNonNavSubSections()  
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves an array of non-navigational subsections. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of non-navigational subsections

### **getNumberOfItems**

```
public int getNumberOfItems()
```

---

throws `FrameworkException`, `SQLException`, `SectionNotFoundException`

Retrieves the number of item IDs within the section available in the user's validation organization. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** number of item IDs in the section available in the user's organization ID.

### **getRelatedSectionIDs**

```
public int[] getRelatedSectionIDs(String relationCode)
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of section IDs for all sections related to this section by the relationship code passed in as parameter. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** String - `relationCode` (specifies the type of relationship; for example, "SUBSTITUTE")

**Returns:** array of related section IDs

### **getRelatedSections**

```
public Section[] getRelatedSections(String relationCode)
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of sections related to this section by the relationship code passed in as parameter. The load level of the related sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of related sections

### **getSectionID**

```
public int getSectionID()
```

Retrieves section ID

**Returns:** the ID of the section

### **getSectionType**

```
public String getSectionType()
```

Retrieves section type

**Returns:** section type of the section

### **getSiblingSectionIDs**

```
public int[] getSiblingSectionIDs(boolean includeSelf)
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves an array of section IDs containing the IDs of the same level siblings. Uses the `includeSelf` parameter to determine whether to return this section's ID in the array. This section's ID should be returned when it is necessary to preserve the ordering of this section within its list of siblings. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** `includeSelf` (whether to return this section's ID in the array)

**Returns:** array of section IDs

### **getSiblingSections**

```
public Section[] getSiblingSections(boolean includeSelf)
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of sections containing the same level siblings. The load level of the sections will be SHALLOW. Uses the `includeSelf` parameter to determine whether to return this section in the array. This section should be returned when it is necessary to preserve the ordering of this section within its list of siblings. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** `includeSelf` (whether to return this section in the array)

**Returns:** array of sections

### **getSubSectionIDs**

```
public int[] getSubSectionIDs()
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of navigational subsection IDs. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of navigational subsection IDs

### **getSubSectionIDs**

```
public int[] getSubSectionIDs(String sectType)
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of subsection IDs with the section type passed in as parameter. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** sectType (section type; Section.NAVIGATIONAL, Section.FEATURED)

**Returns:** array of subsection IDs for those subsections with the specified section type

### **getSubSectionItemIDs**

```
public int[] getSubSectionItemIDs(String sectionType)
throws FrameworkException, SQLException, ItemNotFoundException,
SectionNotFoundException
```

Retrieves an array containing the IDs of items in subsections with the specified section type. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** sectionType (section type; Section.NAVIGATIONAL or Section.FEATURED)

**Returns:** array of item IDs

### **getSubSectionItems**

```
public Item[] getSubSectionItems(String sectionType)
throws FrameworkException, SQLException, ItemNotFoundException,
SectionNotFoundException
```

Retrieves an array containing items in a subsection with the specified section type. The load level of the items will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Parameters:** sectionType (section type; Section.NAVIGATIONAL or Section.FEATURED)

**Returns:** array of items

### **getSubSections**

```
public Section[] getSubSections()
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves an array of navigational subsections. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of navigational subsections

### **getSubSections**

```
public Section[] getSubSections(int num)
throws SQLException, FrameworkException, SectionNotFoundException
```

Retrieves an array of navigational subsections up to the maximum number passed in as parameter. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to Yes, minisite exclusions will be removed.

**Parameters:** num (maximum number of navigational subsections to return)

**Returns:** array of navigational subsections

### **getSubSections**

```
public Section[] getSubSections(String sectType)
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of subsections with the section type passed in as parameter. The load level of the sections will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to Yes, minisite exclusions will be removed.

**Parameters:** sectType (section type; Section.NAVIGATIONAL or Section.FEATURED)

**Returns:** array of subsections for those subsections with the specified section type.

### **getSuperSection**

```
public Section getSuperSection()
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves the immediate super section. The load level of the supersection will be SHALLOW. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** parent section

### **getSuperSectionID**

```
public int getSuperSectionID()
```



---

throws `FrameworkException`, `SQLException`, `SectionNotFoundException`

Retrieves ID of the immediate super section. If the profile IBE: Use Catalog exclusions is set to Yes, minisite exclusions will be removed.

**Returns:** parent section ID

### **getSuperSectionIDs**

```
public int[] getSuperSectionIDs()
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of supersection IDs, starting with the minisite root section. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of supersection IDs, starting with the minisite root section

### **getSuperSections**

```
public Section[] getSuperSections()
throws FrameworkException, SQLException, SectionNotFoundException
```

Retrieves an array of supersections, starting with the minisite root section. If the profile IBE: Use Catalog exclusions is set to 'Yes', minisite exclusions will be removed.

**Returns:** array of supersections, starting with the minisite root section

### **getTemplateFileName**

```
public String getTemplateFileName()
throws TemplateNotFoundException, FrameworkException, SQLException
```

Retrieves the file name of the physical template associated with the section

**Returns:** file name of the template associated with this section

### **isFeatured**

```
public boolean isFeatured()
```

Determines whether the section is a featured section

**Returns:** true if the section is featured, false otherwise

**isLeafSection**

```
public boolean isLeafSection()  
throws FrameworkException, SQLException, SectionNotFoundException
```

Determines whether this section is a leaf section. A leaf section is a section with no subsections.

**Returns:** true if the section is a leaf section, otherwise false

**load**

```
public static Section load(int sectID)  
throws FrameworkException, SQLException, SectionNotFoundException
```

Loads a section with the section ID passed in as parameter. The load level of the section will be SHALLOW.

**Parameters:** sectID (Section ID corresponding to `jtf_dsp_sections_vl.section_id`)

**Returns:** section with values loaded for the proper members.

**load**

```
public static Section load(int sectID, int mode)  
throws FrameworkException, SQLException, SectionNotFoundException
```

Loads a section with the section ID and mode passed in as parameter.

**Parameters:** sectID (Section ID corresponding to `jtf_dsp_sections_vl.section_id`)

mode (load level for the section; `Section.SHALLOW` or `Section.DEEP`)

**Returns:** section with values loaded for the proper members.

**load**

```
public static Section load(String accessName)  
throws FrameworkException, SQLException, SectionNotFoundException
```

Loads a section with the access name passed in as parameter. The load level of the section will be SHALLOW.

**Parameters:** accessName (Section access name corresponding to `jtf_dsp_sections_vl.access_name`)

**Returns:** section with values loaded for the proper members.

**load**

```
public static Section load(String accessName, int mode)
throws FrameworkException, SQLException, SectionNotFoundException
```

Loads a section with the access name and mode passed in as parameter.

**Parameters:** accessName (Section access name corresponding to jtf\_dsp\_sections\_vl.access\_name)

mode (load level for the section; Section.SHALLOW or Section.DEEP)

**Returns:** section with values loaded for the proper members

**load**

```
public static Section[] load(int sectIDs[])
throws FrameworkException, SQLException, SectionNotFoundException
```

Load the sections with the section IDs passed in as parameter.

**Parameters:** sectIDs (array of section IDs corresponding to jtf\_dsp\_sections\_vl.section\_id)

**Returns:** array of sections with values loaded for the proper members

**load**

```
public static Section[] load(int sectIDs[], int mode)
throws FrameworkException, SQLException, SectionNotFoundException,
CatalogException
```

Loads the sections with the sectionIDs and mode passed in as parameter.

**Parameters:** sectIDs (array of section IDs corresponding to jtf\_dsp\_sections\_vl.section\_id)

mode (load level for the sections; Section.SHALLOW or Section.DEEP)

**Returns:** array of sections with values loaded for the proper members

**load**

```
public static Section[] load(String accessNames[])
throws SQLException, FrameworkException, SectionNotFoundException,
CatalogException
```

Loads the sections with the access names passed in as parameter. The load level of the sections will be SHALLOW.

**Parameters:** accessNames (array of section access names corresponding to jtf\_dsp\_sections\_vl.access\_name)

**Returns:** array of sections with values loaded for the proper members

### **load**

```
public static Section[] load(String accessNames[], int mode)
throws SQLException, FrameworkException, SectionNotFoundException
```

Loads the sections with the access names and mode passed in as parameter.

**Parameters:** accessNames (array of section access names corresponding to jtf\_dsp\_sections\_vl.access\_name)

mode (load level for the section; Section.SHALLOW or Section.DEEP)

**Returns:** array of sections with values loaded for the proper members

## **8.6 Exception Classes for Package oracle.apps.ibe.catalog**

### **SQLException**

Exception class to throw if database error occurs.

### **FrameworkException**

Exception class to throw if error occurs while trying to get connection.

### **ItemNotFoundException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >
oracle.apps.jtf.base.resources.FrameworkException >
oracle.apps.ibe.catalog.ItemNotFoundException
```

```
public class ItemNotFoundException
```

```
extends oracle.apps.jtf.base.resources.FrameworkException
```

Exception class to throw for Item class methods when item is not found in the database.

Exception class to throw for Section class methods when a section/subsection does not contain any items.

Used by getItem(), getItemIDs(), getSubSectionItemIDs(), getSubSectionItems() in the Section class.

### **InvalidColumnNumberException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.catalog.InvalidColumnNumberException  
public class InvalidColumnNumberException
```

extends oracle.apps.jtf.base.resources.FrameworkException

Exception class to throw when a table attribute column number passed in is not between 1 and 15, or a table segment column number passed in is not between 1 and 20.

Used by `getAttributeColumn(int columnNumber)` and `getSegmentColumn(int columnNumber)` in Item and Section classes.

### **PriceNotFoundException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.catalog.PriceNotFoundException  
public class PriceNotFoundException
```

extends oracle.apps.jtf.base.resources.FrameworkException

Exception class to throw when a price for an item is not found by the Pricing engine.

Used by `getListPrice()` and `getBestPrice()` in the Item and PriceObject class.

### **CatalogException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.catalog.CatalogException  
public class CatalogException
```

extends oracle.apps.jtf.base.resources.FrameworkException

General Exception for the Catalog. Exception class to throw if error occurs while retrieving requested data.

### **SectionNotFoundException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.catalog.SectionNotFoundException  
public class SectionNotFoundException
```

extends oracle.apps.jtf.base.resources.FrameworkException

Exception class to throw when a section is not found in the database or a section does not contain a subsection of a particular type.

Used by `getLeafSubSectionIDs()`, `getLeafSubSections()`, `getRelatedSections()`, `getRelatedSectionIDs()`, `getSiblingSectionIDs()`, `getSiblingSections()`, `getSubSection()`, `getSubSectionID()`, `getSubSections()`, `getSubSectionIDs()`, `getSuperSection`, `getSuperSectionID`, `getSuperSections()`, `getSuperSectionIDs()`, `getFeaturedSubsection()`, `getFeaturedSubSectionID()` in the `Section` class.

---

---

# Oracle iStore 11i Shopping Cart Quote APIs

This chapter contains the following information about the Oracle iStore 11i Shopping Cart Quote public class APIs:

- [Shopping Cart Quote API Class Summary](#)
- [Class CCTrxnOutRecord](#)
- [Class Contract](#)
- [Class ControlRecord](#)
- [Class FreightChargeRecord](#)
- [Class HeaderRecord](#)
- [Class LineAttributeExtRecord](#)
- [Class LineDetailRecord](#)
- [Class LineRecord](#)
- [Class LineRelationshipRecord](#)
- [Class OrderHeaderRecord](#)
- [Class PaymentRecord](#)
- [Class PriceAdjustmentAttributeRecord](#)
- [Class PriceAdjustmentRecord](#)
- [Class PriceAdjustmentRelationshipRecord](#)
- [Class PriceAttributeRecord](#)
- [Class Quote](#)
- [Class QuoteAccessRecord](#)

- 
- [Class ShipmentRecord](#)
  - [Class SubmitControlRecord](#)
  - [Class TaxDetailRecord](#)
  - [Exceptions for Package oracle.apps.ibe.shoppingcart.quote](#)



## 9.1 Shopping Cart Quote API Class Summary

APIs for the Oracle iStore 11i Shopping Cart Quote are in the package `oracle.apps.ibe.shoppingcart.quote`. The table below describes the classes briefly.

**Table 9–1 Shopping Cart Quote Class Summary**

Class	Description
<a href="#">Class CCTrxnOutRecord</a>	
<a href="#">Class Contract</a>	A class for Contracts.
<a href="#">Class ContractException</a>	Exception class to throw if error occurs during Contract class method.
<a href="#">Class ControlRecord</a>	Java wrapper class of the PL/SQL record type <code>Control_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class FreightChargeRecord</a>	Java wrapper class of the PL/SQL record type <code>Freight_Charge_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class HeaderRecord</a>	Java wrapper class of the PL/SQL record type <code>Qte_Header_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class LineAttributeExtRecord</a>	Java wrapper class of the PL/SQL record type <code>Line_Attr_Ext_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class LineDetailRecord</a>	Java wrapper class of the PL/SQL record type <code>Qte_Line_Dtl_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class LineRecord</a>	Java wrapper class of the PL/SQL record type <code>Qte_Line_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class LineRelationshipRecord</a>	Java wrapper class of the PL/SQL record type <code>Line_Relationship_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class OrderHeaderRecord</a>	
<a href="#">Class PaymentRecord</a>	Java wrapper class of the PL/SQL record type <code>Payment_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class PriceAdjustmentAttributeRecord</a>	Java wrapper class of the PL/SQL record type <code>Price_Adj_Attr_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class PriceAdjustmentRecord</a>	Java wrapper class of the PL/SQL record type <code>Price_Adj_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .

**Table 9–1 Shopping Cart Quote Class Summary (Cont.)**

<b>Class</b>	<b>Description</b>
<a href="#">Class PriceAdjustmentRelationshipRecord</a>	Java wrapper class of the PL/SQL record type <code>Price_Adjustment_Relationship_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class PriceAttributeRecord</a>	Java wrapper class of the PL/SQL record type <code>Price_Attributes_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class Quote</a>	A class that represents quotes which are used as shopping carts, Express Checkout carts, and checked out carts.
<a href="#">Class QuoteAccessRecord</a>	The sharees' access control information for quotes. The fields are based on the table <code>IBE_SH_QUOTE_ACCESS</code> .
<a href="#">Class QuoteException</a>	Exception class to throw if Quote class method action has already been performed by others or if there is an application error.
<a href="#">Class ShipmentRecord</a>	Java wrapper class of the PL/SQL record type <code>Shipment_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .
<a href="#">Class SubmitControlRecord</a>	Java wrapper class of the PL/SQL record type <code>Submit_Control_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> . It is used to control the submit process.
<a href="#">Class TaxDetailRecord</a>	Java wrapper class of the PL/SQL record type <code>Tax_Detail_Rec_Type</code> in the PL/SQL package <code>ASO_QUOTE_PUB</code> .

## 9.2 Class CCTrxnOutRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.CCTrxnOutRecord
public class CCTrxnOutRecord
    extends Object
```

### 9.2.1 Variables for Class CCTrxnOutRecord

#### **auth\_code**

```
public String auth_code
```

#### **bep\_err\_code**

```
public String bep_err_code
```

**bep\_err\_message**

```
public String bep_err_message
```

**err\_code**

```
public String err_code
```

**err\_location**

```
public BigDecimal err_location
```

**err\_message**

```
public String err_message
```

**nls\_lang**

```
public String nls_lang
```

**RCS\_ID**

```
public static final String RCS_ID
```

**status**

```
public BigDecimal status
```

**txn\_date**

```
public Timestamp txn_date
```

**txn\_id**

```
public BigDecimal txn_id
```

## 9.2.2 Constructors for Class CCTrxnOutRecord

**CCTrxnOutRecord**

```
public CCTrxnOutRecord()
```

**CCTrxnOutRecord**

```
public CCTrxnOutRecord(boolean __RosettaUseGMISSValues)
```

## 9.3 Class Contract

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.Contract
public class Contract
    extends Object
    A class for contracts
```

### 9.3.1 Variables for Class Contract

#### **ACTIVE**

```
public static final int ACTIVE
```

#### **APPROVED**

```
public static final int APPROVED
```

#### **CANCELLED**

```
public static final int CANCELLED
```

#### **ENTERED**

```
public static final int ENTERED
```

The possible values for the state of a contract

#### **EXPIRED**

```
public static final int EXPIRED
```

#### **HOLD**

```
public static final int HOLD
```

#### **RCS\_ID**

```
public static final String RCS_ID
```

Standard public final static String which is initialized with the usual RCS header used by ARCS

#### **RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

Standard public final static boolean which is initialized by a call to `oracle.apps.fnd.common.VersionInfo.recordClassVersion`

### SIGNED

```
public static final int SIGNED
```

### TERMINATED

```
public static final int TERMINATED
```

## 9.3.2 Constructors for Class Contract

### Contract

```
public Contract()
```

## 9.3.3 Methods for Class Contract

The following table is an index of Class Contract methods:

**Table 9–2 Method Index for Class Contract**

Method	Description
<a href="#">createContract</a>	Creates a contract using the specified quote and template
<a href="#">getContract</a>	Returns the contracts associated with this quote
<a href="#">getContractID</a>	Retrieves the contractID
<a href="#">getContractNumber</a>	Retrieves the contractNumber
<a href="#">getContractText</a>	Returns the text of the articles contained in this contract
<a href="#">getQuoteID</a>	Retrieves the quoteID
<a href="#">getState</a>	Retrieves the state of the contract possible values
<a href="#">notifyContractChange</a>	Sends a notification to the specified sales rep with the comments
<a href="#">setEntered</a>	Sets the state of the contract to entered
<a href="#">setSigned</a>	Sets the state of the contract to signed

### createContract

```
public static Contract createContract(BigDecimal quoteID, BigDecimal templateID)
```

throws `FrameworkException`, `SQLException`, `ContractException`

Creates a contract using the specified quote and template

### **getContract**

```
public static Contract[] getContract(BigDecimal quoteID)
throws FrameworkException, SQLException
```

Returns the contracts associated with this quote. If no contracts are associated, then it returns null.

### **getContractID**

```
public BigDecimal getContractID()
```

Retrieves the contractID

**Returns:** the contract ID

### **getContractNumber**

```
public String getContractNumber()
```

Retrieves the contractNumber

**Returns:** the contract number

### **getContractText**

```
public Reader[] getContractText()
throws FrameworkException, SQLException
```

Returns the text of the articles contained in this contract. For performance reasons, a reader stream (one for each article) is returned. After use, each reader stream should be closed by the calling application.

### **getContractText**

```
public static Reader[] getContractText(BigDecimal contractID)
throws FrameworkException, SQLException
```

Returns the text of the articles contained in the specified contract. For performance reasons, a reader stream (one for each article) is returned. After use, each reader stream should be closed by the calling application.

**getQuoteID**

```
public BigDecimal getQuoteID()
```

Retrieves the quoteID

**Returns:** the ID of the quote associated to this contract

**getState**

```
public int getState()
```

Retrieves the state of the contract possible values: Contract.ENTERED, Contract.APPROVED, Contract.SIGNED, Contract.ACTIVE, Contract.CANCELLED, Contract.TERMINATED.

**Returns:** the state of the contract

**notifyContractChange**

```
public void notifyContractChange(String salesRepEmail, String comments)
throws FrameworkException, SQLException, ContractException
```

Sends a notification to the specified sales rep with the comments

**setEntered**

```
public void setEntered()
throws FrameworkException, SQLException, ContractException
```

Sets the state of the contract to entered

**setSigned**

```
public void setSigned()
throws FrameworkException, SQLException, ContractException
```

Sets the state of the contract to signed

## 9.4 Class ControlRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.ControlRecord
public class ControlRecord
extends Object
```

Java wrapper class of the PL/SQL record type `Control_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields `calculate_tax_flag` and `calculate_freight_charge_flag` can have the value "Y" for yes and "N" for no.

This class is used in methods of the class `Quote` like `appendToAndShare`, `merge`, `replaceAndShare`, and `save` for pricing.

You should not set the field `line_pricing_event` if you want to get prices for the whole quote.

### 9.4.1 Variables for Class ControlRecord

#### **auto\_version\_flag**

```
public String auto_version_flag
```

#### **calculate\_freight\_charge\_flag**

```
public String calculate_freight_charge_flag
```

#### **calculate\_tax\_flag**

```
public String calculate_tax_flag
```

#### **header\_pricing\_event**

```
public String header_pricing_event
```

#### **last\_update\_date**

```
public Timestamp last_update_date
```

#### **line\_pricing\_event**

```
public String line_pricing_event
```

#### **pricing\_request\_type**

```
public String pricing_request_type
```

#### **RCS\_ID**

```
public static final String RCS_ID
```



## 9.4.2 Constructors for Class ControlRecord

### **ControlRecord**

```
public ControlRecord()
```

## 9.5 Class FreightChargeRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.FreightChargeRecord
```

```
public class FreightChargeRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Freight_Charge_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_FREIGHT_CHARGES`.

### 9.5.1 Variables for Class FreightChargeRecord

#### **attribute\_category**

```
public String attribute_category
```

#### **attribute1**

```
public String attribute1
```

#### **attribute2**

```
public String attribute2
```

#### **attribute3**

```
public String attribute3
```

#### **attribute4**

```
public String attribute4
```

#### **attribute5**

```
public String attribute5
```

#### **attribute6**

```
public String attribute6
```

**attribute7**

public String attribute7

**attribute8**

public String attribute8

**attribute9**

public String attribute9

**attribute10**

public String attribute10

**attribute11**

public String attribute11

**attribute12**

public String attribute12

**attribute13**

public String attribute13

**attribute14**

public String attribute14

**attribute15**

public String attribute15

**charge\_amount**

public BigDecimal charge\_amount

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**freight\_charge\_id**

```
public BigDecimal freight_charge_id
```

**freight\_charge\_type\_id**

```
public BigDecimal freight_charge_type_id
```

**last\_update\_date**

```
public Timestamp last_update_date
```

**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**operation\_code**

```
public String operation_code
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

```
public Timestamp program_update_date
```

**quote\_line\_id**

```
public BigDecimal quote_line_id
```

**quote\_shipment\_id**

```
public BigDecimal quote_shipment_id
```

**qte\_line\_index**

```
public BigDecimal qte_line_index
```

### **RCS\_ID**

```
public static final String RCS_ID
```

### **request\_id**

```
public BigDecimal request_id
```

### **shipment\_index**

```
public BigDecimal shipment_index
```

## **9.5.2 Constructors for Class FreightChargeRecord**

### **FreightChargeRecord**

```
public FreightChargeRecord()
```

## **9.6 Class HeaderRecord**

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.HeaderRecord  
public class HeaderRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Qte_Header_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the view `ASO_QUOTE_HEADERS_V`.

### **9.6.1 Variables for Class HeaderRecord**

The following table is an index of Class HeaderRecord variables:

#### **accounting\_rule\_id**

```
public BigDecimal accounting_rule_id
```

#### **attribute\_category**

```
public String attribute_category
```

#### **attribute1**

```
public String attribute1
```

**attribute2**

```
public String attribute2
```

**attribute3**

```
public String attribute3
```

**attribute4**

```
public String attribute4
```

**attribute5**

```
public String attribute5
```

**attribute6**

```
public String attribute6
```

**attribute7**

```
public String attribute7
```

**attribute8**

```
public String attribute8
```

**attribute9**

```
public String attribute9
```

**attribute10**

```
public String attribute10
```

**attribute11**

```
public String attribute11
```

**attribute12**

```
public String attribute12
```

**attribute13**

```
public String attribute13
```

**attribute14**

public String attribute14

**attribute15**

public String attribute15

**contract\_id**

public BigDecimal contract\_id

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**currency\_code**

public String currency\_code

**cust\_account\_id**

public BigDecimal cust\_account\_id

**employee\_person\_id**

public BigDecimal employee\_person\_id

**exchange\_rate**

public BigDecimal exchange\_rate

**exchange\_rate\_date**

public Timestamp exchange\_rate\_date

**exchange\_type\_code**

public String exchange\_type\_code

**ffm\_request\_id**

public BigDecimal ffm\_request\_id

**invoice\_to\_address1**

```
public String invoice_to_address1
```

**invoice\_to\_address2**

```
public String invoice_to_address2
```

**invoice\_to\_address3**

```
public String invoice_to_address3
```

**invoice\_to\_address4**

```
public String invoice_to_address4
```

**invoice\_to\_city**

```
public String invoice_to_city
```

**invoice\_to\_contact\_first\_name**

```
public String invoice_to_contact_first_name
```

**invoice\_to\_contact\_last\_name**

```
public String invoice_to_contact_last_name
```

**invoice\_to\_contact\_middle\_name**

```
public String invoice_to_contact_middle_name
```

**invoice\_to\_country**

```
public String invoice_to_country
```

**invoice\_to\_country\_code**

```
public String invoice_to_country_code
```

**invoice\_to\_county**

```
public String invoice_to_county
```

**invoice\_to\_party\_id**

```
public BigDecimal invoice_to_party_id
```

**invoice\_to\_party\_name**

public String invoice\_to\_party\_name

**invoice\_to\_party\_site\_id**

public BigDecimal invoice\_to\_party\_site\_id

**invoice\_to\_postal\_code**

public String invoice\_to\_postal\_code

**invoice\_to\_province**

public String invoice\_to\_province

**invoice\_to\_state**

public String invoice\_to\_state

**invoicing\_rule\_id**

public BigDecimal invoicing\_rule\_id

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**marketing\_source\_code**

public String marketing\_source\_code

**marketing\_source\_code\_id**

public BigDecimal marketing\_source\_code\_id

**marketing\_source\_name**

public String marketing\_source\_name



**order\_id**

```
public BigDecimal order_id
```

**order\_number**

```
public BigDecimal order_number
```

**order\_type\_id**

```
public BigDecimal order_type_id
```

**order\_type\_name**

```
public String order_type_name
```

**ordered\_date**

```
public Timestamp ordered_date
```

**org\_contact\_id**

```
public BigDecimal org_contact_id
```

**org\_id**

```
public BigDecimal org_id
```

**orig\_mktg\_source\_code\_id**

```
public BigDecimal orig_mktg_source_code_id
```

**original\_system\_reference**

```
public String original_system_reference
```

**party\_id**

```
public BigDecimal party_id
```

**party\_name**

```
public String party_name
```

**party\_type**

```
public String party_type
```

**payment\_amount**

public BigDecimal payment\_amount

**person\_first\_name**

public String person\_first\_name

**person\_last\_name**

public String person\_last\_name

**person\_middle\_name**

public String person\_middle\_name

**phone\_id**

public BigDecimal phone\_id

**price\_frozen\_date**

public Timestamp price\_frozen\_date

**price\_list\_id**

public BigDecimal price\_list\_id

**price\_list\_name**

public String price\_list\_name

**program\_application\_id**

public BigDecimal program\_application\_id

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**qte\_contract\_id**

public BigDecimal qte\_contract\_id

**quote\_category\_code**

```
public String quote_category_code
```

**quote\_expiration\_date**

```
public Timestamp quote_expiration_date
```

**quote\_header\_id**

```
public BigDecimal quote_header_id
```

**quote\_name**

```
public String quote_name
```

**quote\_number**

```
public BigDecimal quote_number
```

**quote\_password**

```
public String quote_password
```

**quote\_source\_code**

```
public String quote_source_code
```

**quote\_status**

```
public String quote_status
```

**quote\_status\_code**

```
public String quote_status_code
```

**quote\_status\_id**

```
public BigDecimal quote_status_id
```

**quote\_version**

```
public BigDecimal quote_version
```

**RCS\_ID**

```
public static final String RCS_ID
```

**request\_id**

public BigDecimal request\_id

**sales\_channel\_code**

public String sales\_channel\_code

**salesrep\_first\_name**

public String salesrep\_first\_name

**salesrep\_last\_name**

public String salesrep\_last\_name

**surcharge**

public BigDecimal surcharge

**total\_adjusted\_amount**

public BigDecimal total\_adjusted\_amount

**total\_adjusted\_percent**

public BigDecimal total\_adjusted\_percent

**total\_list\_price**

public BigDecimal total\_list\_price

**total\_quote\_price**

public BigDecimal total\_quote\_price

**total\_shipping\_charge**

public BigDecimal total\_shipping\_charge

**total\_tax**

public BigDecimal total\_tax

## 9.6.2 Constructors for Class HeaderRecord

**HeaderRecord**

public HeaderRecord()

## 9.7 Class LineAttributeExtRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.LineAttributeExtRecord  
public class LineAttributeExtRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Line_Attribs_Ext_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_QUOTE_LINE_ATTRIBS_EXT`.

This class stores attribute and value for quote line attributes not captured in `LineRecord` and `LineDetailRecord`.

### 9.7.1 Variables for Class LineAttributeExtRecord

#### **application\_id**

```
public BigDecimal application_id
```

#### **attribute\_type\_code**

```
public String attribute_type_code
```

#### **created\_by**

```
public BigDecimal created_by
```

#### **creation\_date**

```
public Timestamp creation_date
```

#### **end\_date\_active**

```
public Timestamp end_date_active
```

#### **last\_update\_date**

```
public Timestamp last_update_date
```

#### **last\_update\_login**

```
public BigDecimal last_update_login
```

#### **last\_updated\_by**

```
public BigDecimal last_updated_by
```

**line\_attribute\_id**

public BigDecimal line\_attribute\_id

**name**

public String name

**operation\_code**

public String operation\_code

**program\_application\_id**

public BigDecimal program\_application\_id

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**qte\_line\_index**

public BigDecimal qte\_line\_index

**quote\_line\_id**

public BigDecimal quote\_line\_id

**RCS\_ID**

public static final String RCS\_ID

**request\_id**

public BigDecimal request\_id

**start\_date\_active**

public Timestamp start\_date\_active

**status**

public String status

**value**

```
public String value
```

**value\_type**

```
public String value_type
```

## 9.7.2 Constructors for Class LineAttributeExtRecord

**LineAttributeExtRecord**

```
public LineAttributeExtRecord()
```

## 9.8 Class LineDetailRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.LineDetailRecord
```

```
public class LineDetailRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Qte_Line_Dtl_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_QUOTE_LINE_DETAILS`.

Stores service related attributes, model/option related attributes, and return related attributes of quote lines.

### 9.8.1 Variables for Class LineDetailRecord

**attribute\_category**

```
public String attribute_category
```

**attribute1**

```
public String attribute1
```

**attribute2**

```
public String attribute2
```

**attribute3**

```
public String attribute3
```

**attribute4**

public String attribute4

**attribute5**

public String attribute5

**attribute6**

public String attribute6

**attribute7**

public String attribute7

**attribute8**

public String attribute8

**attribute9**

public String attribute9

**attribute10**

public String attribute10

**attribute11**

public String attribute11

**attribute12**

public String attribute12

**attribute13**

public String attribute13

**attribute14**

public String attribute14

**attribute15**

public String attribute15



**change\_reason\_code**

```
public String change_reason_code
```

**complete\_configuration\_flag**

```
public String complete_configuration_flag
```

**component\_code**

```
public String component_code
```

**config\_header\_id**

```
public BigDecimal config_header_id
```

**config\_item\_id**

```
public BigDecimal config_item_id
```

**config\_revision\_num**

```
public BigDecimal config_revision_num
```

**creation\_date**

```
public Timestamp creation_date
```

**created\_by**

```
public BigDecimal created_by
```

**last\_update\_date**

```
public Timestamp last_update_date
```

**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**operation\_code**

```
public String operation_code
```

**program\_application\_id**

public BigDecimal program\_application\_id

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**qte\_line\_index**

public BigDecimal qte\_line\_index

**quote\_line\_detail\_id**

public BigDecimal quote\_line\_detail\_id

**quote\_line\_id**

public BigDecimal quote\_line\_id

**RCS\_ID**

public static final String RCS\_ID

**request\_id**

public BigDecimal request\_id

**return\_attribute\_category**

public String return\_attribute\_category

**return\_attribute1**

public String return\_attribute1

**return\_attribute2**

public String return\_attribute2

**return\_attribute3**

public String return\_attribute3

**return\_attribute4**

```
public String return_attribute4
```

**return\_attribute5**

```
public String return_attribute5
```

**return\_attribute6**

```
public String return_attribute6
```

**return\_attribute7**

```
public String return_attribute7
```

**return\_attribute8**

```
public String return_attribute8
```

**return\_attribute9**

```
public String return_attribute9
```

**return\_attribute10**

```
public String return_attribute10
```

**return\_attribute11**

```
public String return_attribute11
```

**return\_attribute12**

```
public String return_attribute12
```

**return\_attribute13**

```
public String return_attribute13
```

**return\_attribute14**

```
public String return_attribute14
```

**return\_attribute15**

```
public String return_attribute15
```

**return\_reason\_code**

public String return\_reason\_code

**return\_ref\_header\_id**

public BigDecimal return\_ref\_header\_id

**return\_ref\_line\_id**

public BigDecimal return\_ref\_line\_id

**return\_ref\_type**

public String return\_ref\_type

**service\_coterminate\_flag**

public String service\_coterminate\_flag

**service\_duration**

public BigDecimal service\_duration

**service\_number**

public BigDecimal service\_number

**service\_period**

public String service\_period

**service\_ref\_line\_id**

public BigDecimal service\_ref\_line\_id

**service\_ref\_line\_number**

public BigDecimal service\_ref\_line\_number

**service\_ref\_option\_num**

public BigDecimal service\_ref\_option\_num

**service\_ref\_order\_number**

public BigDecimal service\_ref\_order\_number

```
service_ref_qte_line_index  
public BigDecimal service_ref_qte_line_index
```

```
service_ref_shipment_num  
public BigDecimal service_ref_shipment_num
```

```
service_ref_system_id  
public BigDecimal service_ref_system_id
```

```
service_ref_type_code  
public String service_ref_type_code
```

```
service_unit_list_percent  
public BigDecimal service_unit_list_percent
```

```
service_unit_selling_percent  
public BigDecimal service_unit_selling_percent
```

```
unit_percent_base_price  
public BigDecimal unit_percent_base_price
```

```
valid_configuration_flag  
public String valid_configuration_flag
```

## 9.8.2 Constructors for Class LineDetailRecord

```
LineDetailRecord  
public LineDetailRecord()
```

## 9.9 Class LineRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.LineRecord  
public class LineRecord  
extends Object
```

Java wrapper class of the PL/SQL record type `Qte_Line_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the view ASO\_QUOTE\_LINES.

## 9.9.1 Variables for Class LineRecord

### **accounting\_rule\_id**

```
public BigDecimal accounting_rule_id
```

### **attribute\_category**

```
public String attribute_category
```

### **attribute1**

```
public String attribute1
```

### **attribute2**

```
public String attribute2
```

### **attribute3**

```
public String attribute3
```

### **attribute4**

```
public String attribute4
```

### **attribute5**

```
public String attribute5
```

### **attribute6**

```
public String attribute6
```

### **attribute7**

```
public String attribute7
```

### **attribute8**

```
public String attribute8
```

### **attribute9**

```
public String attribute9
```

**attribute10**

```
public String attribute10
```

**attribute11**

```
public String attribute11
```

**attribute12**

```
public String attribute12
```

**attribute13**

```
public String attribute13
```

**attribute14**

```
public String attribute14
```

**attribute15**

```
public String attribute15
```

**backorder\_flag**

```
public String backorder_flag
```

**created\_by**

```
public BigDecimal created_by
```

**creation\_date**

```
public Timestamp creation_date
```

**currency\_code**

```
public String currency_code
```

**end\_date\_active**

```
public Timestamp end_date_active
```

**ffm\_content\_name**

```
public String ffm_content_name
```

**ffm\_content\_type**

public String ffm\_content\_type

**ffm\_document\_type**

public String ffm\_document\_type

**ffm\_media\_id**

public String ffm\_media\_id

**ffm\_media\_type**

public String ffm\_media\_type

**ffm\_user\_note**

public String ffm\_user\_note

**inventory\_item\_id**

public BigDecimal inventory\_item\_id

**invoice\_to\_party\_id**

public BigDecimal invoice\_to\_party\_id

**invoice\_to\_party\_site\_id**

public BigDecimal invoice\_to\_party\_site\_id

**invoicing\_rule\_id**

public BigDecimal invoicing\_rule\_id

**item\_relationship\_type**

public String item\_relationship\_type

**item\_type\_code**

public String item\_type\_code

**last\_update\_date**

public Timestamp last\_update\_date



**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**line\_adjusted\_amount**

```
public BigDecimal line_adjusted_amount
```

**line\_adjusted\_percent**

```
public BigDecimal line_adjusted_percent
```

**line\_category\_code**

```
public String line_category_code
```

**line\_list\_price**

```
public BigDecimal line_list_price
```

**line\_number**

```
public BigDecimal line_number
```

**line\_quote\_price**

```
public BigDecimal line_quote_price
```

**marketing\_source\_code\_id**

```
public BigDecimal marketing_source_code_id
```

**operation\_code**

```
public String operation_code
```

**order\_line\_type\_id**

```
public BigDecimal order_line_type_id
```

**org\_id**

```
public BigDecimal org_id
```

**organization\_id**

public BigDecimal organization\_id

**price\_list\_id**

public BigDecimal price\_list\_id

**price\_list\_line\_id**

public BigDecimal price\_list\_line\_id

**pricing\_quantity\_uom**

public String pricing\_quantity\_uom

**program\_application\_id**

public BigDecimal program\_application\_id

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**quantity**

public BigDecimal quantity

**quote\_header\_id**

public BigDecimal quote\_header\_id

**quote\_line\_id**

public BigDecimal quote\_line\_id

**RCS\_ID**

public static final String RCS\_ID

**request\_id**

public BigDecimal request\_id

**related\_item\_id**

```
public BigDecimal related_item_id
```

**split\_shipment\_flag**

```
public String split_shipment_flag
```

**start\_date\_active**

```
public Timestamp start_date_active
```

**uom\_code**

```
public String uom_code
```

## 9.9.2 Constructors for Class LineRecord

**LineRecord**

```
public LineRecord()
```

## 9.10 Class LineRelationshipRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.LineRelationshipRecord  
public class LineRelationshipRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Line_Rltship_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_LINE_RELATIONSHIP`.

This class stores the relationship between quote lines.

### 9.10.1 Variables for Class LineRelationshipRecord

**created\_by**

```
public BigDecimal created_by
```

**creation\_date**

```
public Timestamp creation_date
```

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**line\_relationship\_id**

public BigDecimal line\_relationship\_id

**operation\_code**

public String operation\_code

**program\_application\_id**

public BigDecimal program\_application\_id

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**qte\_line\_index**

public BigDecimal qte\_line\_index

**quote\_line\_id**

public BigDecimal quote\_line\_id

**RCS\_ID**

public static final String RCS\_ID

**reciprocal\_flag**

public String reciprocal\_flag

**request\_id**

```
public BigDecimal request_id
```

**related\_qte\_line\_index**

```
public BigDecimal related_qte_line_index
```

**related\_quote\_line\_id**

```
public BigDecimal related_quote_line_id
```

**relationship\_type\_code**

```
public String relationship_type_code
```

## 9.10.2 Constructors for Class LineRelationshipRecord

**LineRelationshipRecord**

```
public LineRelationshipRecord()
```

## 9.11 Class OrderHeaderRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.OrderHeaderRecord
```

```
public class OrderHeaderRecord
```

```
extends Object
```

### 9.11.1 Variables for Class OrderHeaderRecord

**contract\_id**

```
public BigDecimal contract_id
```

**order\_header\_id**

```
public BigDecimal order_header_id
```

**order\_number**

```
public BigDecimal order_number
```

**order\_request\_id**

```
public BigDecimal order_request_id
```

### **RCS\_ID**

```
public static final String RCS_ID
```

### **status**

```
public String status
```

## **9.11.2 Constructors for Class OrderHeaderRecord**

### **OrderHeaderRecord**

```
public OrderHeaderRecord()
```

## **9.12 Class PaymentRecord**

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.PaymentRecord
```

```
public class PaymentRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Payment_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_PAYMENTS`.

Store payment related information for the quote.

### **9.12.1 Variables for Class PaymentRecord**

#### **attribute\_category**

```
public String attribute_category
```

#### **attribute1**

```
public String attribute1
```

#### **attribute2**

```
public String attribute2
```

#### **attribute3**

```
public String attribute3
```

**attribute4**

```
public String attribute4
```

**attribute5**

```
public String attribute5
```

**attribute6**

```
public String attribute6
```

**attribute7**

```
public String attribute7
```

**attribute8**

```
public String attribute8
```

**attribute9**

```
public String attribute9
```

**attribute10**

```
public String attribute10
```

**attribute11**

```
public String attribute11
```

**attribute12**

```
public String attribute12
```

**attribute13**

```
public String attribute13
```

**attribute14**

```
public String attribute14
```

**attribute15**

```
public String attribute15
```

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**credit\_card\_approval\_code**

public String credit\_card\_approval\_code

**credit\_card\_approval\_date**

public Timestamp credit\_card\_approval\_date

**credit\_card\_code**

public String credit\_card\_code

**credit\_card\_expiration\_date**

public Timestamp credit\_card\_expiration\_date

**credit\_card\_holder\_name**

public String credit\_card\_holder\_name

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**operation\_code**

public String operation\_code

**payment\_amount**

public BigDecimal payment\_amount



**payment\_id**

```
public BigDecimal payment_id
```

**payment\_option**

```
public String payment_option
```

**payment\_ref\_number**

```
public String payment_ref_number
```

**payment\_term\_id**

```
public BigDecimal payment_term_id
```

**payment\_type\_code**

```
public String payment_type_code
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

```
public Timestamp program_update_date
```

**qte\_line\_index**

```
public BigDecimal qte_line_index
```

**quote\_header\_id**

```
public BigDecimal quote_header_id
```

**quote\_line\_id**

```
public BigDecimal quote_line_id
```

**quote\_shipment\_id**

```
public BigDecimal quote_shipment_id
```

**RCS\_ID**

```
public static final String RCS_ID
```

**request\_id**

```
public BigDecimal request_id
```

**shipment\_index**

```
public BigDecimal shipment_index
```

## 9.12.2 Constructors for Class PaymentRecord

**PaymentRecord**

```
public PaymentRecord()
```

## 9.13 Class PriceAdjustmentAttributeRecord

```
java.lang.Object >  
oracle.apps.ibe.shoppingcart.quote.PriceAdjustmentAttributeRecord  
public class PriceAdjustmentAttributeRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Price_Adj_Attr_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the view `ASO_PRICE_ADJ_ATTRIBS_V`.

### 9.13.1 Variables for Class PriceAdjustmentAttributeRecord

**comparison\_operator**

```
public String comparison_operator
```

**created\_by**

```
public BigDecimal created_by
```

**creation\_date**

```
public Timestamp creation_date
```

**flex\_title**

```
public String flex_title
```

**last\_update\_date**

```
public Timestamp last_update_date
```

**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**operation\_code**

```
public String operation_code
```

**price\_adj\_attrib\_id**

```
public BigDecimal price_adj_attrib_id
```

**price\_adj\_index**

```
public BigDecimal price_adj_index
```

**price\_adjustment\_id**

```
public BigDecimal price_adjustment_id
```

**pricing\_attr\_value\_from**

```
public String pricing_attr_value_from
```

**pricing\_attr\_value\_to**

```
public String pricing_attr_value_to
```

**pricing\_attribute**

```
public String pricing_attribute
```

**pricing\_context**

```
public String pricing_context
```

```
program_application_id  
public BigDecimal program_application_id  
  
program_id  
public BigDecimal program_id  
  
program_update_date  
public Timestamp program_update_date  
  
qte_line_index  
public BigDecimal qte_line_index  
  
RCS_ID  
public static final String RCS_ID  
  
request_id  
public BigDecimal request_id  
  
shipment_index  
public BigDecimal shipment_index
```

### 9.13.2 Constructors for Class PriceAdjustmentAttributeRecord

```
PriceAdjustmentAttributeRecord  
public PriceAdjustmentAttributeRecord()
```

## 9.14 Class PriceAdjustmentRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.PriceAdjustmentRecord  
public class PriceAdjustmentRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Price_Adj_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the view `ASO_PRICE_ADJUSTMENTS_V`.

### 9.14.1 Variables for Class PriceAdjustmentRecord

**accrual\_conversion\_rate**

```
public BigDecimal accrual_conversion_rate
```

**accrual\_flag**

```
public String accrual_flag
```

**adjusted\_amount**

```
public BigDecimal adjusted_amount
```

**applied\_flag**

```
public String applied_flag
```

**arithmetic\_operator**

```
public String arithmetic_operator
```

**attribute\_category**

```
public String attribute_category
```

**attribute1**

```
public String attribute1
```

**attribute2**

```
public String attribute2
```

**attribute3**

```
public String attribute3
```

**attribute4**

```
public String attribute4
```

**attribute5**

```
public String attribute5
```

**attribute6**

```
public String attribute6
```

**attribute7**

public String attribute7

**attribute8**

public String attribute8

**attribute9**

public String attribute9

**attribute10**

public String attribute10

**attribute11**

public String attribute11

**attribute12**

public String attribute12

**attribute13**

public String attribute13

**attribute14**

public String attribute14

**attribute15**

public String attribute15

**automatic\_flag**

public String automatic\_flag

**benefit\_qty**

public BigDecimal benefit\_qty

**benefit\_uom\_code**

public String benefit\_uom\_code

**change\_reason\_code**

```
public String change_reason_code
```

**change\_reason\_text**

```
public String change_reason_text
```

**change\_sequence**

```
public String change_sequence
```

**charge\_subtype\_code**

```
public String charge_subtype_code
```

**charge\_type\_code**

```
public String charge_type_code
```

**cost\_id**

```
public BigDecimal cost_id
```

**created\_by**

```
public BigDecimal created_by
```

**creation\_date**

```
public Timestamp creation_date
```

**credit\_or\_charge\_flag**

```
public String credit_or_charge_flag
```

**estimated\_flag**

```
public String estimated_flag
```

**expiration\_date**

```
public Timestamp expiration_date
```

**inc\_in\_sales\_performance**

```
public String inc_in_sales_performance
```

**include\_on\_returns\_flag**

public String include\_on\_returns\_flag

**invoiced\_flag**

public String invoiced\_flag

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**list\_line\_no**

public String list\_line\_no

**modified\_from**

public BigDecimal modified\_from

**modified\_to**

public BigDecimal modified\_to

**modifier\_header\_id**

public BigDecimal modifier\_header\_id

**modifier\_level\_code**

public String modifier\_level\_code

**modifier\_line\_id**

public BigDecimal modifier\_line\_id

**modifier\_line\_type\_code**

public String modifier\_line\_type\_code



**modifier\_mechanism\_type\_code**

```
public String modifier_mechanism_type_code
```

**on\_invoice\_flag**

```
public String on_invoice_flag
```

**operand**

```
public BigDecimal operand
```

**operation\_code**

```
public String operation_code
```

**orig\_sys\_discount\_ref**

```
public String orig_sys_discount_ref
```

**parent\_adjustment\_id**

```
public BigDecimal parent_adjustment_id
```

**price\_adjustment\_id**

```
public BigDecimal price_adjustment_id
```

**price\_break\_type\_code**

```
public String price_break_type_code
```

**pricing\_group\_sequence**

```
public BigDecimal pricing_group_sequence
```

**pricing\_phase\_id**

```
public BigDecimal pricing_phase_id
```

**print\_on\_invoice\_flag**

```
public String print_on_invoice_flag
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

public BigDecimal program\_id

**program\_update\_date**

public Timestamp program\_update\_date

**proration\_type\_code**

public String proration\_type\_code

**qte\_line\_index**

public BigDecimal qte\_line\_index

**quote\_header\_id**

public BigDecimal quote\_header\_id

**quote\_line\_id**

public BigDecimal quote\_line\_id

**quote\_shipment\_id**

public BigDecimal quote\_shipment\_id

**range\_break\_quantity**

public BigDecimal range\_break\_quantity

**RCS\_ID**

public static final String RCS\_ID

**rebate\_payment\_system\_code**

public String rebate\_payment\_system\_code

**rebate\_transaction\_reference**

public String rebate\_transaction\_reference

**rebate\_transaction\_type\_code**

public String rebate\_transaction\_type\_code

**redeemed\_date**

```
public Timestamp redeemed_date
```

**redeemed\_flag**

```
public String redeemed_flag
```

**request\_id**

```
public BigDecimal request_id
```

**shipment\_index**

```
public BigDecimal shipment_index
```

**source\_system\_code**

```
public String source_system_code
```

**split\_action\_code**

```
public String split_action_code
```

**substitution\_attribute**

```
public String substitution_attribute
```

**tax\_code**

```
public String tax_code
```

**tax\_exempt\_flag**

```
public String tax_exempt_flag
```

**tax\_exempt\_number**

```
public String tax_exempt_number
```

**tax\_exempt\_reason\_code**

```
public String tax_exempt_reason_code
```

**update\_allowable\_flag**

```
public String update_allowable_flag
```

**update\_allowed**

public String update\_allowed

**updated\_flag**

public String updated\_flag

## 9.14.2 Constructors for Class PriceAdjustmentRecord

**PriceAdjustmentRecord**

public PriceAdjustmentRecord()

## 9.15 Class PriceAdjustmentRelationshipRecord

```
java.lang.Object >  
oracle.apps.ibe.shoppingcart.quote.PriceAdjustmentRelationshipRecord  
public class PriceAdjustmentRelationshipRecord
```

extends Object

Java wrapper class of the PL/SQL record type `Price_Adj_Rltship_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the view `ASO_PRICE_ADJ_RELATIONSHIPS_V`.

Stores the relationship between quote lines and price adjustments and also between price adjustments.

### 9.15.1 Variables for Class PriceAdjustmentRelationshipRecord

**adj\_relationship\_id**

public BigDecimal adj\_relationship\_id

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**last\_update\_date**

```
public Timestamp last_update_date
```

**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**operation\_code**

```
public String operation_code
```

**price\_adj\_index**

```
public BigDecimal price_adj_index
```

**price\_adjustment\_id**

```
public BigDecimal price_adjustment_id
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

```
public Timestamp program_update_date
```

**qte\_line\_index**

```
public BigDecimal qte_line_index
```

**quote\_line\_id**

```
public BigDecimal quote_line_id
```

**quote\_shipment\_id**

```
public BigDecimal quote_shipment_id
```

### **RCS\_ID**

```
public static final String RCS_ID
```

### **request\_id**

```
public BigDecimal request_id
```

### **rltd\_price\_adj\_id**

```
public BigDecimal rltd_price_adj_id
```

### **rltd\_price\_adj\_index**

```
public BigDecimal rltd_price_adj_index
```

### **shipment\_index**

```
public BigDecimal shipment_index
```

## **9.15.2 Constructors for Class PriceAdjustmentRelationshipRecord**

### **PriceAdjustmentRelationshipRecord**

```
public PriceAdjustmentRelationshipRecord()
```

## **9.16 Class PriceAttributeRecord**

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.PriceAttributeRecord
```

```
public class PriceAttributeRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type `Price_Attributes_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`.

The fields are based on the table `ASO_PRICE_ATTRIBUTES`.

Stores information on qualifiers and pricing attributes for which the corresponding price adjustment line qualifies.

### **9.16.1 Variables for Class PriceAttributeRecord**

#### **attribute1**

```
public String attribute1
```

**attribute2**

```
public String attribute2
```

**attribute3**

```
public String attribute3
```

**attribute4**

```
public String attribute4
```

**attribute5**

```
public String attribute5
```

**attribute6**

```
public String attribute6
```

**attribute7**

```
public String attribute7
```

**attribute8**

```
public String attribute8
```

**attribute9**

```
public String attribute9
```

**attribute10**

```
public String attribute10
```

**attribute11**

```
public String attribute11
```

**attribute12**

```
public String attribute12
```

**attribute13**

```
public String attribute13
```

**attribute14**

public String attribute14

**attribute15**

public String attribute15

**context**

public String context

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**flex\_title**

public String flex\_title

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**operation\_code**

public String operation\_code

**price\_attribute\_id**

public BigDecimal price\_attribute\_id

**pricing\_attribute1**

public String pricing\_attribute1



**pricing\_attribute2**

```
public String pricing_attribute2
```

**pricing\_attribute3**

```
public String pricing_attribute3
```

**pricing\_attribute4**

```
public String pricing_attribute4
```

**pricing\_attribute5**

```
public String pricing_attribute5
```

**pricing\_attribute6**

```
public String pricing_attribute6
```

**pricing\_attribute7**

```
public String pricing_attribute7
```

**pricing\_attribute8**

```
public String pricing_attribute8
```

**pricing\_attribute9**

```
public String pricing_attribute9
```

**pricing\_attribute10**

```
public String pricing_attribute10
```

**pricing\_attribute11**

```
public String pricing_attribute11
```

**pricing\_attribute12**

```
public String pricing_attribute12
```

**pricing\_attribute13**

```
public String pricing_attribute13
```

**pricing\_attribute14**

public String pricing\_attribute14

**pricing\_attribute15**

public String pricing\_attribute15

**pricing\_attribute16**

public String pricing\_attribute16

**pricing\_attribute17**

public String pricing\_attribute17

**pricing\_attribute18**

public String pricing\_attribute18

**pricing\_attribute19**

public String pricing\_attribute19

**pricing\_attribute20**

public String pricing\_attribute20

**pricing\_attribute21**

public String pricing\_attribute21

**pricing\_attribute22**

public String pricing\_attribute22

**pricing\_attribute23**

public String pricing\_attribute23

**pricing\_attribute24**

public String pricing\_attribute24

**pricing\_attribute25**

public String pricing\_attribute25

**pricing\_attribute26**

```
public String pricing_attribute26
```

**pricing\_attribute27**

```
public String pricing_attribute27
```

**pricing\_attribute28**

```
public String pricing_attribute28
```

**pricing\_attribute29**

```
public String pricing_attribute29
```

**pricing\_attribute30**

```
public String pricing_attribute30
```

**pricing\_attribute31**

```
public String pricing_attribute31
```

**pricing\_attribute32**

```
public String pricing_attribute32
```

**pricing\_attribute33**

```
public String pricing_attribute33
```

**pricing\_attribute34**

```
public String pricing_attribute34
```

**pricing\_attribute35**

```
public String pricing_attribute35
```

**pricing\_attribute36**

```
public String pricing_attribute36
```

**pricing\_attribute37**

```
public String pricing_attribute37
```

**pricing\_attribute38**

public String pricing\_attribute38

**pricing\_attribute39**

public String pricing\_attribute39

**pricing\_attribute40**

public String pricing\_attribute40

**pricing\_attribute41**

public String pricing\_attribute41

**pricing\_attribute42**

public String pricing\_attribute42

**pricing\_attribute43**

public String pricing\_attribute43

**pricing\_attribute44**

public String pricing\_attribute44

**pricing\_attribute45**

public String pricing\_attribute45

**pricing\_attribute46**

public String pricing\_attribute46

**pricing\_attribute47**

public String pricing\_attribute47

**pricing\_attribute48**

public String pricing\_attribute48

**pricing\_attribute49**

public String pricing\_attribute49

**pricing\_attribute50**

```
public String pricing_attribute50
```

**pricing\_attribute51**

```
public String pricing_attribute51
```

**pricing\_attribute52**

```
public String pricing_attribute52
```

**pricing\_attribute53**

```
public String pricing_attribute53
```

**pricing\_attribute54**

```
public String pricing_attribute54
```

**pricing\_attribute55**

```
public String pricing_attribute55
```

**pricing\_attribute56**

```
public String pricing_attribute56
```

**pricing\_attribute57**

```
public String pricing_attribute57
```

**pricing\_attribute58**

```
public String pricing_attribute58
```

**pricing\_attribute59**

```
public String pricing_attribute59
```

**pricing\_attribute60**

```
public String pricing_attribute60
```

**pricing\_attribute61**

```
public String pricing_attribute61
```

**pricing\_attribute62**

public String pricing\_attribute62

**pricing\_attribute63**

public String pricing\_attribute63

**pricing\_attribute64**

public String pricing\_attribute64

**pricing\_attribute65**

public String pricing\_attribute65

**pricing\_attribute66**

public String pricing\_attribute66

**pricing\_attribute67**

public String pricing\_attribute67

**pricing\_attribute68**

public String pricing\_attribute68

**pricing\_attribute69**

public String pricing\_attribute69

**pricing\_attribute70**

public String pricing\_attribute70

**pricing\_attribute71**

public String pricing\_attribute71

**pricing\_attribute72**

public String pricing\_attribute72

**pricing\_attribute73**

public String pricing\_attribute73

**pricing\_attribute74**

```
public String pricing_attribute74
```

**pricing\_attribute75**

```
public String pricing_attribute75
```

**pricing\_attribute76**

```
public String pricing_attribute76
```

**pricing\_attribute77**

```
public String pricing_attribute77
```

**pricing\_attribute78**

```
public String pricing_attribute78
```

**pricing\_attribute79**

```
public String pricing_attribute79
```

**pricing\_attribute80**

```
public String pricing_attribute80
```

**pricing\_attribute81**

```
public String pricing_attribute81
```

**pricing\_attribute82**

```
public String pricing_attribute82
```

**pricing\_attribute83**

```
public String pricing_attribute83
```

**pricing\_attribute84**

```
public String pricing_attribute84
```

**pricing\_attribute85**

```
public String pricing_attribute85
```

**pricing\_attribute86**

public String pricing\_attribute86

**pricing\_attribute87**

public String pricing\_attribute87

**pricing\_attribute88**

public String pricing\_attribute88

**pricing\_attribute89**

public String pricing\_attribute89

**pricing\_attribute90**

public String pricing\_attribute90

**pricing\_attribute91**

public String pricing\_attribute91

**pricing\_attribute92**

public String pricing\_attribute92

**pricing\_attribute93**

public String pricing\_attribute93

**pricing\_attribute94**

public String pricing\_attribute94

**pricing\_attribute95**

public String pricing\_attribute95

**pricing\_attribute96**

public String pricing\_attribute96

**pricing\_attribute97**

public String pricing\_attribute97



**pricing\_attribute98**

```
public String pricing_attribute98
```

**pricing\_attribute99**

```
public String pricing_attribute99
```

**pricing\_attribute100**

```
public String pricing_attribute100
```

**pricing\_context**

```
public String pricing_context
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

```
public Timestamp program_update_date
```

**qte\_line\_index**

```
public BigDecimal qte_line_index
```

**quote\_header\_id**

```
public BigDecimal quote_header_id
```

**quote\_line\_id**

```
public BigDecimal quote_line_id
```

**RCS\_ID**

```
public static final String RCS_ID
```

**request\_id**

```
public BigDecimal request_id
```

## 9.16.2 Constructors for Class PriceAttributeRecord

### PriceAttributeRecord

```
public PriceAttributeRecord()
```

## 9.17 Class Quote

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.Quote  
public class Quote
```

```
extends Object
```

A class that represents quotes which are used as shopping carts, Express Checkout carts, and checked out carts.

You use this class to perform operations like creation, loading, update, deletion, and sharing of a quote with all of its related information including quote header, quote lines, payment, shipment, tax, price adjustment, etc. All these information are stores in ASO (Oracle Order Capture) tables.

All the methods that perform read operations like `load`, `loadAll`, `loadSharees`, and `loadVersions` executes SQL through JDBC.

All the other methods that perform write operations are Java wrappers that call corresponding PL/SQL procedures through JDBC.

All the fields of the record classes like `HeaderRecord`, `LineRecord`, etc. are initialized with constant values that correspond to the PL/SQL constants `G_MISS_CHAR`, `G_MISS_NUM`, and `G_MISS_DATE` in the PL/SQL package `FND_API`.

The following general rules apply for all the methods:

- All the optional parameters accept null value.
- The following optional parameters are included in the methods that create a quote and can be used for sharing information (if the quote is not to be shared, they should be null):
  - `password`: the password to access the shared quote
  - `url`: a character string that specifies the url to access the shared quote.

An example:

```
String url = "http://" + request.getRemoteHost() + ":" +  
request.getServerPort() + "/" +  
DisplayManager.getTemplate("STORE_SCART_VIEW_LIST_P").getFileName() +
```

```

"?minisite=" + RequestCtx.getMinisiteId() +
"&Retrieve.x=Retrieve" +
"&dispFlag=sharee" +
"&sharedFlag=true" +
"&prevref=" + DisplayManager.getTemplate("STORE_SCART_VIEW_
LIST").getFileName() +
"&retSharTPassword" + java.net.URLEncoder.encode(password);

```

- e-mail addresses: e-mail addresses of the sharees
- privilege types: the privilege type granted to the sharees; the valid values are:
  - \* 'A': all privileges
  - \* 'F': feedback-only privilege
  - \* 'R': read-only privilege
- comment: text that is e-mailed to the sharees as a comment
- The parameter combineSameItem is used in methods like appendToAndShare, and merge that might have multiple lines for the same standard item. If it is 'Y', it will combine the lines into one and update the quantity with the sum of the quantities. The lines can be merges only if the inventory item ID and UOM code are same and the item type ID standard. If it is 'N', it will keep separate lines. If it is null, it will use the value of user profile 'TBE\_SC\_MERGE\_SHOPCART\_LINES'.
- Methods that create or update a quote have the optional parameters, price list ID and currency code that can be used to avoid the overhead of determining which price list to use to calculate the price. If you do not specify, Oracle Pricing will determine the best price list the user is qualified for.
- Methods that create or update a quote have the optional parameter, control record to indicate pricing related flags.

### 9.17.1 Variables for Class Quote

#### **COMBINED\_LINES**

```
public static final int COMBINED_LINES
```

Constant to have the quote to combine the lines of the same item.

### **controlRec**

```
public ControlRecord controlRec
```

Control record used for pricing.

This class is used in methods like `appendToAndShare`, `merge`, `replaceAndShare`, and `save` for pricing.

You should not set the field `line_pricing_event` if you want to get prices for the whole quote.

### **headerFreightChargeRec**

```
public FreightChargeRecord headerFreightChargeRec[]
```

Header level freight charge information.

### **headerPaymentRec**

```
public PaymentRecord headerPaymentRec[]
```

Header level payment information.

### **headerPriceAttrRec**

```
public PriceAttributeRecord headerPriceAttrRec[]
```

Header level price attributes.

### **headerRec**

```
public HeaderRecord headerRec
```

Quote header information.

### **headerShipmentRec**

```
public ShipmentRecord headerShipmentRec[]
```

Header level shipment information.

A header level shipment means the shipping information is the same for all the quote lines.

### **headerTaxDetailRec**

```
public TaxDetailRecord headerTaxDetailRec[]
```

Header level tax information.

### **lineAttrExtRec**

```
public LineAttributeExtRecord lineAttrExtRec[]
```

Stores attribute and value for quote line attributes not captured in lineRec and lineDetRec.

### **lineDetRec**

```
public LineDetailRecord lineDetRec[]
```

Service related attributes, model/option related attributes, and return related attributes of quote lines.

### **lineFreightChargeRec**

```
public FreightChargeRecord lineFreightChargeRec[]
```

Line level freight charge information.

### **linePaymentRec**

```
public PaymentRecord linePaymentRec[]
```

Line level payment information.

### **linePriceAttrRec**

```
public PriceAttributeRecord linePriceAttrRec[]
```

Line level price attributes.

### **lineRec**

```
public LineRecord lineRec[]
```

Quote line information.

### **lineRelRec**

```
public LineRelationshipRecord lineRelRec[]
```

Relationship between quote lines.

### **lineShipmentRec**

```
public ShipmentRecord lineShipmentRec[]
```

Line level shipment information.

A line level shipment means each line has its own shipping information.

### **lineTaxDetailRec**

```
public TaxDetailRecord lineTaxDetailRec[]
```

Line level tax information.

### **priceAdjAttrRec**

```
public PriceAdjustmentAttributeRecord priceAdjAttrRec[]
```

Price adjustment attributes.

### **priceAdjRec**

```
public PriceAdjustmentRecord priceAdjRec[]
```

Price adjustments.

### **priceAdjRelRec**

```
public PriceAdjustmentRelationshipRecord priceAdjRelRec[]
```

Relationship between quote lines and price adjustments and also between price adjustments.

### **quoteAccessRec**

```
public QuoteAccessRecord quoteAccessRec[]
```

The sharees' access control information for quotes.

### **RCS\_ID**

```
public static final String RCS_ID
```

Standard public final static String which is intialized with the usual RCS header used by ARCS.

**RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

Standard public final static boolean which is initialized by a call to `oracle.apps.fnd.common.VersionInfo.recordClassVersion`.

**SEPARATE\_LINES**

```
public static final int SEPARATE_LINES
```

Constant to have the quote to have separate lines for the same item.

**submitControlRec**

```
public SubmitControlRecord submitControlRec
```

Control record used while submitting a quote to order.

**USE\_PROFILE**

```
public static final int USE_PROFILE
```

Constant to let the user profile decide whether to combine the lines of the same item or to have separate lines for the same item.

## 9.17.2 Constructors for Class Quote

**Quote**

```
public Quote()
```

## 9.17.3 Methods for Class Quote

The following table is an index of Class Quote methods:

**Table 9–3 Method Index for Class Quote**

Method	Description
<a href="#">activate</a>	Activates a quote, that is, makes a quote as the active quote of the user account
<a href="#">appendToAndShare</a>	Appends a quote to another quote and saves sharees' information
<a href="#">authorizePayment</a>	Authorizes credit card payment

**Table 9–3 Method Index for Class Quote (Cont.)**

<b>Method</b>	<b>Description</b>
<code>delete</code>	Deletes the quote
<code>deleteAllLines</code>	Delete all the lines of the quote given the quote ID
<code>getQuoteID</code>	Gets the quote ID given the quote number and quote version
<code>getQuoteName</code>	Gets the quote name given the quote number and quote version
<code>getShareePrivilege</code>	Gets sharee's privilege for the quote
<code>isOrdered</code>	Checks if the quote is ordered
<code>isShippable</code>	Checks if the quote is shippable
<code>load</code>	Loads quote information
<code>loadAll</code>	Loads all quotes owned by the given account excluding the active quote
<code>loadSharees</code>	Retrieves the quote access control information
<code>loadVersions</code>	Loads quote versions of the same quote number
<code>merge</code>	Merges the active quote of the guest user account to the active quote of the registered user account
<code>replaceAndShare</code>	Replaces a quote to another quote and saves sharee information
<code>retrieveSharedQuote</code>	Retrieves a shared quote
<code>save</code>	Creates or updates a quote
<code>saveAsAndShare</code>	Saves a quote and saves sharee information
<code>share</code>	Shares a quote with sharees
<code>submit</code>	Submits a quote to turn it into an order

**activate**

```
public static BigDecimal activate(BigDecimal quoteID, Timestamp  
quoteLastUpdateDate, boolean keepOrigQuote)  
throws FrameworkException, QuoteException, SQLException
```

Activates a quote, that is, makes a quote as the active quote of the user account.

Active quotes have the name IBEACTIVECART.

**Parameters:** quoteID (quote ID)



`quoteLastUpdateDate` (The last update date of the quote header. Optional. If it is not null, a check is done to verify that there was no update to the quote header after the given last update date.)

`keepOrigQuote` (Indicates if the quote of `quoteID` is to be kept and a new quote with the same contents as the quote of `quoteID` with the name `IBEACTIVECART` is to be created. Use `true` to keep the quote `quoteID` and create a new quote; `false` just to rename the quote of `quoteID` to `IBEACTIVECART`.)

**Returns:** quote ID of the new active quote for the user account

### **appendToAndShare**

```
public static BigDecimal appendToAndShare(BigDecimal quoteID, Timestamp
quoteLastUpdateDate, BigDecimal toQuoteID, boolean createNewVersion, int
combineSameItem, String toQuotePassword, String url, String emailAddresses[],
String privilegeTypes[], String comment, BigDecimal priceListID, String
currencyCode, ControlRecord controlRec)
throws FrameworkException, QuoteException, SQLException
```

Appends a quote to another quote and saves sharees' information.

The optional parameters, `toQuotePassword`, `url`, `emailAddresses`, and `privilegeTypes`, are to share the quote.

The optional parameters, `priceListID` and `currencyCode` are to specify which price list to use to calculate the price.

The optional parameter `controlRec` is used for pricing.

**Parameters:** `quoteID` (ID of the appending quote)

`quoteLastUpdateDate` (Optional. The last update date of the quote header. If it is not null, a check is done to verify that there was no update to the quote header after the given last update date.)

`toQuoteID` (ID of the quote to which the quote `quoteID` is appended to)

`createNewVersion` (If `true`, it creates a new version of the quote `toQuoteID` and appends the quote `quoteID` to the quote `toQuoteID`. Otherwise, it appends the quote `quoteID` to the quote `toQuoteID`.)

`combineSameItem` (If `COMBINED_LINES`, it combines lines of the same item. If `SEPARATE_LINES`, it creates separate lines for the same item. If `USE_PROFILE` or any other values, it uses the user profile `IBE_SC_MERGE_SHOPCART_LINES` determine whether to combines lines or create separate lines.)

toQuotePassword (Optional. Password the sharees should use to access the quote toQuoteID)

url (Optional. URL to access the shared quote)

emailAddresses (Optional. E-mail addresses of the sharees)

privilegeTypes (Optional. Privilege types of the sharees)

priceListID (Optional. Price list ID)

currencyCode (Optional. Currency code)

controlRec (control information for pricing)

**Returns:** quote ID of the appended quote

### **authorizePayment**

```
public static CCTrxnOutRecord authorizePayment(BigDecimal quoteID, Timestamp
inQuoteLastUpdateDate, BigDecimal appID, String authType, Timestamp
outQuoteLastUpdateDate)
throws FrameworkException, QuoteException, SQLException
```

Authorizes credit card payment.

The payment record of the quote should be updated with the credit card information before calling this method.

The return status is set in the QuotePmtOutRecord which contains the return values from the Payment Server. If app\_id is null, we will default app ID of Oracle Order Capture. If authType is null, we will default to 'AUTHONLY'.

**Parameters:** quoteID (quote ID)

inQuoteLastUpdateDate (Optional. The last update date of the quote header. If it is not null, a check is done to verify that there was no update to the quote header after the given last update date.)

appID (Application ID registered in Oracle iPayment. If null, it defaults to the application ID of Oracle Order Capture.)

authType (Authoriazion type. 'AUTHONLY' to just authorize; 'AUTHCAPTURE' to authorize and capture)

outQuoteLastUpdateDate (the last update date of the quote after authorization)

**Returns:** record containing the result of the authorize operation

**delete**

```
public static void delete(BigDecimal quoteID, Timestamp quoteLastUpdateDate)
throws FrameworkException, QuoteException, SQLException
```

Deletes the quote.

**Parameters:** quoteID (quote ID)

quoteLastUpdateDate (Optional. The last update date of the quote header. If you do not pass null, a check is done to verify that there was no update to the quote header after the given last update date.)

**deleteAllLines**

```
public static Timestamp deleteAllLines(BigDecimal quoteID, Timestamp
quoteLastUpdateDate, BigDecimal shareeNum)
throws FrameworkException, QuoteException, SQLException
```

Delete all the lines of the quote given the quote ID

**Parameters:** quoteID (quote ID)

quoteLastUpdateDate (Optional. The last update date of the quote header. If you do not pass null, a check is done to verify that there was no update to the quote header after the given last update date.)

shareeNum (sharee number. If a user is deleting all the lines as a sharee, he/she should put his/her sharee number to see if he/she has the privilege to do so.)

**Returns:** last update date of the quote header record

**getQuoteID**

```
public static BigDecimal getQuoteID(BigDecimal quoteNumber, BigDecimal
quoteVersion)
throws FrameworkException, SQLException
```

Gets the quote ID given the quote number and quote version

**Parameters:** quoteNumber (quote number)

quoteVersion (quote version)

**Returns:** quote ID; null if the quote ID is not found

**getQuoteName**

```
public static String getQuoteName(BigDecimal quoteNumber, BigDecimal
```

quoteVersion)  
throws FrameworkException, SQLException

Gets the quote name given the quote number and quote version

**Parameters:** quoteNumber (quote number)

quoteVersion (quote version)

**Returns:** quote name; null if not found

### **getShareePrivilege**

public static String getShareePrivilege(BigDecimal quoteNumber, BigDecimal  
quoteVersion, BigDecimal shareeNumber, String password)  
throws FrameworkException, SQLException

Gets sharee's privilege for the quote. If the parameter password is not null, this method will only get the sharee's privilege if the password is validated.

**Parameters:** quoteNumber (quote number)

quoteVersion (quote version)

shareeNumber (sharee number)

password (quote password, optional)

**Returns:** sharee's privilege

### **isOrdered**

public boolean isOrdered()

Checks if the quote is ordered

**Returns:** true if the quote does not exist or it has been ordered; false otherwise

### **isShippable**

public static boolean isShippable(BigDecimal quoteID)  
throws FrameworkException, SQLException

Checks if the quote is shippable

**Parameters:** quoteID (quote ID)

**Returns:** true if at least one of the line items of the quote is shippable; false otherwise.

## load

```
public static Quote load(BigDecimal quoteID, BigDecimal partyID, BigDecimal
custAcctID, boolean loadLine, boolean loadLineDetail, boolean
loadHeaderPriceAttr, boolean loadLinePriceAttr, boolean loadHeaderPayment,
boolean loadLinePayment, boolean loadHeaderShipment, boolean loadLineShipment,
boolean loadHeaderTaxDetail, boolean loadLineTaxDetail, boolean loadLineRel,
boolean loadLineAttrExt, boolean includeOrdered)
throws FrameworkException, SQLException
```

Loads quote information

headerRec field is always loaded and all the other fields are loaded depending on the boolean flags like loadLine, loadLineDetail, etc. The quote to be loaded can be an active quote, a saved quote, or a contract quote.

If quoteID is not null, it loads a quote using quoteID.

If quoteID is null, it loads the active quote using partyID and custAcctID. The following fields are not loaded.

- headerRec
  - ffm\_request\_id
  - qte\_contract\_id
  - party\_name
- lineRec
  - operation\_code
  - pricing\_quantity\_uom
  - ffm\_content\_name
  - ffm\_document\_type
  - ffm\_media\_type
  - ffm\_media\_id
  - ffm\_content\_type
  - ffm\_user\_note
- lineDetRec
  - operation\_code
  - qte\_line\_index

- service\_ref\_qte\_line\_index
- return\_attribute\_category
- return\_reason\_code
- change\_reason\_code
- lineRelRec
  - operation\_code
  - qte\_line\_index
  - related\_qte\_line\_index
- lineAttrExtRec
  - qte\_line\_index
  - shipment\_index
  - quote\_header\_id
  - quote\_shipment\_id
  - operation\_code
- headerPaymentRec and linePaymentRec
  - operation\_code
  - qte\_line\_index
  - shipment\_index
- headerShipmentRec and lineShipmentRec
  - operation\_code
  - qte\_line\_index
  - ship\_quote\_price
  - pricing\_quantity
- headerTaxDetailRec and lineTaxDetailRec
  - operation\_code
  - qte\_line\_index
  - shipment\_index

**Parameters:** quoteID (quote ID which corresponds to the column quote\_header\_id in the table ASO\_QUOTE\_HEADERS\_ALL)

partyID (party ID)

custAcctID (customer account ID)

loadLine (use true to load lineRec field, false otherwise)

loadLineDetail (use true to load lineDetRec field, false otherwise)

loadHeaderPriceAttr (use true to load headerPriceAttrRec field, false otherwise)

loadLinePriceAttr (use true to load linePriceAttrRec field, false otherwise)

loadHeaderPayment (use true to load headerPaymentRec field, false otherwise)

loadLinePayment (use true to load linePaymentRec field, false otherwise)

loadHeaderShipment (use true to load headerShipmentRec field, false otherwise)

loadLineShipment (use true to load lineShipmentRec field, false otherwise)

loadHeaderTaxDetail (use true to load headerTaxDetailRec field, false otherwise)

loadLineTaxDetail (use true to load lineTaxDetailRec field, false otherwise)

loadLineRel (use true to load lineRelRec field, false otherwise)

loadLineAttrExt (use true to load lineAttrExtRec field, false otherwise)

includeOrdered (use true to indicate that ordered quote can be loaded, false otherwise)

**Returns:** quote object

### **loadAll**

```
public static Quote[] loadAll(BigDecimal partyID, BigDecimal custAcctID, boolean
includeAllVersions, boolean includeOrdered)
throws FrameworkException, SQLException
```

Loads all quotes owned by the given account excluding the active quote. Ordered by quote number in ascending order and by quote version in descending order.

**Parameters:** partyID (party ID)

custAcctID (customer account ID)

includeAllVersions (true if all the versions of quote number should be included, false if only the latest version of quote number should be included)

includeOrdered (true if the ordered quotes should be included, false otherwise)

**Returns:** Quote objects with headerRec fields loaded.

### loadSharees

```
public static QuoteAccessRecord[] loadSharees(BigDecimal quoteID)
throws FrameworkException, SQLException
```

Retrieves the quote access control information

**Parameters:** quoteID (quote ID)

**Returns:** an array of QuoteAccessRecord; one record for each sharee.

### loadVersions

```
public static Quote[] loadVersions(BigDecimal quoteNumber, boolean
includeOrdered)
throws FrameworkException, SQLException
```

Loads quote versions of the same quote number.

Only the field headerRec is loaded.

**Parameters:** quoteNumber (quote number)

includeOrdered (boolean flag to indicate if the ordered quotes should be included)

**Returns:** quote versions with the given quote number

### merge

```
public static BigDecimal merge(BigDecimal guestQuoteID, Timestamp
guestQuoteLastUpdateDate, String mode, int combineSameItem, BigDecimal
regUserPartyID, BigDecimal regUserCustAcctID, BigDecimal priceListID, String
currencyCode, ControlRecord controlRec)
throws FrameworkException, QuoteException, SQLException
```

Merges the active quote of the guest user account to the active quote of the registered user account.

The optional parameters, priceListID and currencyCode are to specify which price list to use to calculate the price.

The optional parameter controlRec is used for pricing.

**Parameters:** guestQuoteID (ID of the appending quote)



questQuoteLastUpdateDate (The last update date of the quote header. Optional parameter for concurrency control)

mode (The mode can have the value of "MERGE", "KEEP", or "REMOVE." "MERGE" is the default value and it merges the guest quote to the registered quote. "KEEP" makes the guest quote as the active quote in registered account. "REMOVE" removes the guest quote.)

combineSameItem (If COMBINED\_LINES, it combines lines of the same item. If SEPARATE\_LINES, it creates separate lines for the same item. If USE\_PROFILE or any other values, it uses the user profile IBE\_SC\_MERGE\_SHOPCART\_LINES determine whether to combines lines or create seperate lines.)

priceListID (the price list ID)

currencyCode (the currency code)

controlRec (the control information for pricing)

**Returns:** ID of the merged active quote of this registered account

### replaceAndShare

```
public static BigDecimal replaceAndShare(BigDecimal quoteID, Timestamp
quoteLastUpdateDate, BigDecimal replacedQuoteID, boolean createNewVersion,
String replacedQuotePassword, String url, String emailAddresses[], String
privilegeTypes[], String comment, BigDecimal priceListID, String currencyCode,
ControlRecord controlRec)
throws FrameworkException, QuoteException, SQLException
```

Replaces a quote to another quote and saves sharee information.

The optional parameters, replacedQuotePassword, url, emailAddresses, and privilegeTypes, are to share the quote.

The optional parameters, priceListID and currencyCode are to specify which price list to use to calculate the price.

The optional parameter controlRec is used for pricing.

**Parameters:** quoteID (ID of the replacing quote)

quoteLastUpdateDate (Optional. The last update date of the quote header. If you do not pass null, a check is done to verify that there was no update to the quote header after the given last update date.)

replacedQuoteID (the ID of the quote by which quoteID is replaced)

createNewVersion (true to create a new version of toQuoteID, false otherwise)

replacedQuotePassword (the password the sharees should use to access the quote replacedQuoteID)

url (URL to access the shared quote)

emailAddresses (e-mail addresses of the sharees)

privilegeTypes (privilege types of the sharees)

priceListID (price list ID)

currencyCode (currency code)

controlRec (the control information for pricing)

**Returns:** ID of the replaced quote

### **retrieveSharedQuote**

```
public static BigDecimal retrieveSharedQuote(BigDecimal quoteNumber, BigDecimal
quoteVersion, String quotePassword, BigDecimal shareePartyID, BigDecimal
shareeCustAcctID, BigDecimal shareeNumber, BigDecimal priceListID, String
currencyCode, ControlRecord controlRec)
throws FrameworkException, QuoteException, SQLException
```

Retrieves a shared quote.

Recalculates the quote price if the user profile 'TBE\_SC\_PRICE\_BASED\_ON\_OWNER' is "N".

The optional parameters, priceListID and currencyCode are to specify which price list to use to calculate the price.

The optional parameter controlRec is used for pricing.

**Parameters:** quoteNumber (quote number)

quoteVersion (quote version)

quotePassword (password to access the shared quote)

shareePartyID (party ID of the sharee)

shareeCustAcctID (customer account ID of the sharee)

shareeNumber (sharee number used to find the privilege of the sharee)

priceListID (Price list ID. If not null, the price engine uses this price list instead of searching for one)

currencyCode (currency code)

controlRec (control information for pricing)

**Returns:** shared quote ID

### save

```
public void save(BigDecimal shareePartyID, BigDecimal shareeCustAcctID,
    BigDecimal shareeNumber, int combineSameItem, boolean autoUpdateActiveQuote,
    boolean saveLine, boolean saveLineDetail, boolean saveHeaderPriceAttr, boolean
    saveLinePriceAttr, boolean savePriceAdj, boolean savePriceAdjAttr, boolean
    savePriceAdjRel, boolean saveHeaderPayment, boolean saveLinePayment, boolean
    saveHeaderShipment, boolean saveLineShipment, boolean saveHeaderFreight, boolean
    saveLineFreight, boolean saveHeaderTaxDetail, boolean saveLineTaxDetail, boolean
    saveLineRel, boolean saveLineAttrExt)
throws FrameworkException, QuoteException, SQLException
```

Creates or updates a quote.

If headerRec.quote\_header\_id is not null, it is used to update the quote.

If headerRec.quote\_header\_id is null and headerRec.quote\_name is 'TBEACTIVECART', it finds the active quote based on party ID and account ID, and updates it. If there is no current active quote, it creates a quote.

If headerRec.quote\_header\_id is null and headerRec.quote\_name is not 'TBEACTIVECART', it creates a quote using the name headerRec.quote\_name.

If the quote is a shared one, the parameters shareePartyId, shareeCustAcctId, and shareeNumber are used to check the privilege and to recalculate price based on sharee.

**Parameters:** combineSameItem (If COMBINED\_LINES, it combines lines of the same item. If SEPARATE\_LINES, it creates separate lines for the same item. If USE\_PROFILE or any other values, it uses the user profile IBE\_SC\_MERGE\_SHOPCART\_LINES determine whether to combines lines or create separate lines. This parameter does not affect configurable and service items)

autoUpdateActiveQuote (Used only when headerRec.quote\_header\_id is null and there is already an active quote in the database. If true, the update to this quote is applied in the active quote. If false, QuoteException is thrown.)

saveLine (use true to save lineRec, false otherwise)

saveHeaderPriceAttr (use true to save headerPriceAttrRec, false otherwise)

saveLinePriceAttr (use true to save linePriceAttrRec, false otherwise)

savePriceAdj (use true to save priceAdjRec, false otherwise)

savePriceAdjAttr (use true to save priceAdjAttrRec, false otherwise)  
savePriceAdjRel (use true to save priceAdjRelRec, false otherwise)  
saveHeaderPayment (use true to save headerPaymentRec, false otherwise)  
saveLinePayment (use true to save linePaymentRec, false otherwise)  
saveHeaderShipment (use true to save headerShipmentRec, false otherwise)  
saveLineShipment (use true to save lineShipmentRec, false otherwise)  
saveHeaderFreight (use true to save headerFreightChargeRec, false otherwise)  
saveLineFreight (use true to save lineFreightChargeRec, false otherwise)  
saveHeaderTaxDetail (use true to save headerTaxDetailRec, false otherwise)  
saveLineTaxDetail (use true to save lineTaxDetailRec, false otherwise)  
saveLineDetail (use true to save lineDetRec, false otherwise)  
saveLineRel (use true to save lineRelRec, false otherwise)  
saveLineAttrExt (use true to save lineAttrExtRec, false otherwise)

### **saveAsAndShare**

```
public static BigDecimal saveAsAndShare(BigDecimal quoteID, Timestamp
quoteLastUpdateDate, String newQuoteName, String newQuoteSourceCode, BigDecimal
partyID, BigDecimal custAcctID, String newQuotePassword, String url, String
emailAddresses[], String privilegeTypes[], String comment, BigDecimal
priceListID, String currencyCode, ControlRecord controlRec)
throws FrameworkException, QuoteException, SQLException
```

Saves a quote and saves sharee information.

The optional parameters, `newQuotePassword`, `url`, `emailAddresses`, and `privilegeTypes`, are to share the quote.

The optional parameters, `priceListID` and `currencyCode` are to specify which price list to use to calculate the price.

The optional parameter `controlRec` is used for pricing.

**Parameters:** `quoteID` (the quote ID)

`quoteLastUpdateDate` (Optional. The last update date of the quote header. If you do not pass null, a check is done to verify that there was no update to the quote header after the given last update date)

newQuoteName (the new quote name)  
newQuoteSourceCode (the new quote source code)  
partyID (party ID)  
custAcctID (customer account ID)  
newQuotePassword (password to access new quote as a sharee)  
url (URL to access the shared quote)  
emailAddresses (e-mail addresses of the sharees)  
privilegeTypes (privilege types of the sharees)  
priceListID (price list ID)  
currencyCode (currency code)  
controlRec (control information for pricing)  
**Returns:** quote ID of the saved quote

### **share**

```
public void share(BigDecimal quoteID, String url, String emailAddresses[],  
String privilegeTypes[], String comment)  
throws FrameworkException, QuoteException, SQLException
```

Shares a quote with sharees

**Parameters:** quoteID (quote ID)

url (URL)

emailAddresses (e-mail addresses, one for each sharee)

privilegeTypes (privilege types, one for each sharee)

### **submit**

```
public static OrderHeaderRecord submit(BigDecimal quoteID, Timestamp  
quoteLastUpdateDate, String salesRepAssistCode, String salesRepEmailAddr, String  
commentForSalesRep, BigDecimal shareePartyID, BigDecimal shareeCustAcctID,  
BigDecimal shareeNumber, SubmitControlRecord submitControlRec)  
throws FrameworkException, QuoteException, SQLException
```

Submits a quote to turn it into an order.

The parameters `salesRepAssistCode`, `salesRepEmailAddr`, and `commentForSalesRep` are used when the customer wants to send an e-mail to the sales representative with a comment.

The parameters `shareePartyID`, `shareeCustAcctID`, and `shareeNumber` need to be passed when a sharee wants to submit a quote.

**Parameters:** `quoteID` (quote ID)

`quoteLastUpdateDate` (Optional. The last update date of the quote header. If you do not pass null, a check is done to verify that there was no update to the quote header after the given last update date.)

`salesRepAssistCode` (Optional. The string that contains the reason code for the assistance of sales representative)

`salesRepEmailAddr` (Optional. The e-mail address of the sales representative)

`commentForSalesRep` (Optional. The comment for the sales representative)

`shareePartyID` (Optional. The party ID of the sharee who submits the quote)

`shareeCustAcctID` (Optional. The customer account ID of the sharee who submits the quote)

`shareeNumber` (Optional. The sharee number of the sharee who submits the quote)

`submitControlRec` (Optional. Submit control information that includes book flag, reserve flag, calculate price flag, and server ID. If null, the default values of 'F', 'F', 'F', and '-1' are used.)

**Returns:** order header information that includes order number, order header ID, order request ID, contract ID, and status

## 9.18 Class QuoteAccessRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.QuoteAccessRecord
public class QuoteAccessRecord
extends Object
```

The sharees' access control information for quotes. The fields are based on the table `IBE_SH_QUOTE_ACCESS`.

## 9.18.1 Variables for Class QuoteAccessRecord

### **created\_by**

public BigDecimal created\_by

### **creation\_date**

public Timestamp creation\_date

### **email\_contact\_address**

public String email\_contact\_address

### **last\_update\_date**

public Timestamp last\_update\_date

### **last\_update\_login**

public BigDecimal last\_update\_login

### **last\_updated\_by**

public BigDecimal last\_updated\_by

### **object\_version\_number**

public BigDecimal object\_version\_number

### **program\_application\_id**

public BigDecimal program\_application\_id

### **program\_id**

public BigDecimal program\_id

### **program\_update\_date**

public Timestamp program\_update\_date

### **quote\_header\_id**

public BigDecimal quote\_header\_id

### **quote\_sharee\_id**

public BigDecimal quote\_sharee\_id

**quote\_sharee\_number**

```
public BigDecimal quote_sharee_number
```

**RCS\_ID**

```
public static final String RCS_ID
```

**request\_id**

```
public BigDecimal request_id
```

**update\_privilege\_type\_code**

```
public String update_privilege_type_code
```

## 9.18.2 Constructors for Class QuoteAccessRecord

**QuoteAccessRecord**

```
public QuoteAccessRecord()
```

## 9.19 Class ShipmentRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.ShipmentRecord
```

```
public class ShipmentRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type Shipment\_Rec\_Type in the PL/SQL package ASO\_QUOTE\_PUB.

The fields are based on the view ASO\_SHIPMENTS\_V.

Stores shipping information for a quote at header or line level.

### 9.19.1 Variables for Class ShipmentRecord

**attribute\_category**

```
public String attribute_category
```

**attribute1**

```
public String attribute1
```



**attribute2**

```
public String attribute2
```

**attribute3**

```
public String attribute3
```

**attribute4**

```
public String attribute4
```

**attribute5**

```
public String attribute5
```

**attribute6**

```
public String attribute6
```

**attribute7**

```
public String attribute7
```

**attribute8**

```
public String attribute8
```

**attribute9**

```
public String attribute9
```

**attribute10**

```
public String attribute10
```

**attribute11**

```
public String attribute11
```

**attribute12**

```
public String attribute12
```

**attribute13**

```
public String attribute13
```

**attribute14**

public String attribute14

**attribute15**

public String attribute15

**created\_by**

public BigDecimal created\_by

**creation\_date**

public Timestamp creation\_date

**fob\_code**

public String fob\_code

**freight\_terms\_code**

public String freight\_terms\_code

**freight\_carrier\_code**

public String freight\_carrier\_code

**last\_update\_date**

public Timestamp last\_update\_date

**last\_update\_login**

public BigDecimal last\_update\_login

**last\_updated\_by**

public BigDecimal last\_updated\_by

**operation\_code**

public String operation\_code

**order\_line\_id**

public BigDecimal order\_line\_id

**packing\_instructions**

```
public String packing_instructions
```

**pricing\_quantity**

```
public BigDecimal pricing_quantity
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

```
public Timestamp program_update_date
```

**promise\_date**

```
public Timestamp promise_date
```

**qte\_line\_index**

```
public BigDecimal qte_line_index
```

**quantity**

```
public BigDecimal quantity
```

**quote\_header\_id**

```
public BigDecimal quote_header_id
```

**quote\_line\_id**

```
public BigDecimal quote_line_id
```

**RCS\_ID**

```
public static final String RCS_ID
```

**request\_date**

```
public Timestamp request_date
```

**request\_id**

public BigDecimal request\_id

**reservation\_id**

public BigDecimal reservation\_id

**reserved\_quantity**

public BigDecimal reserved\_quantity

**schedule\_ship\_date**

public Timestamp schedule\_ship\_date

**ship\_method\_code**

public String ship\_method\_code

**ship\_quote\_price**

public BigDecimal ship\_quote\_price

**ship\_partial\_flag**

public String ship\_partial\_flag

**ship\_set\_id**

public BigDecimal ship\_set\_id

**ship\_to\_address1**

public String ship\_to\_address1

**ship\_to\_address2**

public String ship\_to\_address2

**ship\_to\_address3**

public String ship\_to\_address3

**ship\_to\_address4**

public String ship\_to\_address4

**ship\_to\_city**

```
public String ship_to_city
```

**ship\_to\_contact\_first\_name**

```
public String ship_to_contact_first_name
```

**ship\_to\_contact\_last\_name**

```
public String ship_to_contact_last_name
```

**ship\_to\_contact\_middle\_name**

```
public String ship_to_contact_middle_name
```

**ship\_to\_country**

```
public String ship_to_country
```

**ship\_to\_country\_code**

```
public String ship_to_country_code
```

**ship\_to\_county**

```
public String ship_to_county
```

**ship\_to\_party\_id**

```
public BigDecimal ship_to_party_id
```

**ship\_to\_party\_name**

```
public String ship_to_party_name
```

**ship\_to\_party\_site\_id**

```
public BigDecimal ship_to_party_site_id
```

**ship\_to\_postal\_code**

```
public String ship_to_postal_code
```

**ship\_to\_province**

```
public String ship_to_province
```

**ship\_to\_state**

public String ship\_to\_state

**shipment\_id**

public BigDecimal shipment\_id

**shipment\_priority\_code**

public String shipment\_priority\_code

**shipping\_instructions**

public String shipping\_instructions

## 9.19.2 Constructors for Class ShipmentRecord

**ShipmentRecord**

public ShipmentRecord()

## 9.20 Class SubmitControlRecord

java.lang.Object > oracle.apps.ibe.shoppingcart.quote.SubmitControlRecord  
public class **SubmitControlRecord**

extends Object

Java wrapper class of the PL/SQL record type `Submit_Control_Rec_Type` in the PL/SQL package `ASO_QUOTE_PUB`. It is used to control the submit process.

It has the following 4 fields:

- `book_flag`: "F" is the default value
- `reserve_flag`: "F" is the default value
- `calculate_price`: "F" is the default value
- `server_id`: -1 is the default value

The `book_flag`, `reserve_flag`, and `calculate_price` fields can have the values "T" for true and "F" for false.

## 9.20.1 Variables for Class SubmitControlRecord

### **book\_flag**

```
public String book_flag
```

### **calculate\_price**

```
public String calculate_price
```

### **RCS\_ID**

```
public static final String RCS_ID
```

### **reserve\_flag**

```
public String reserve_flag
```

### **server\_id**

```
public BigDecimal server_id
```

## 9.20.2 Constructors for Class SubmitControlRecord

### **SubmitControlRecord**

```
public SubmitControlRecord()
```

## 9.21 Class TaxDetailRecord

```
java.lang.Object > oracle.apps.ibe.shoppingcart.quote.TaxDetailRecord
```

```
public class TaxDetailRecord
```

```
extends Object
```

Java wrapper class of the PL/SQL record type Tax\_Detail\_Rec\_Type in the PL/SQL package ASO\_QUOTE\_PUB.

The fields are based on the table ASO\_TAX\_DETAILS.

## 9.21.1 Variables for Class TaxDetailRecord

### **attribute\_category**

```
public String attribute_category
```

**attribute1**

public String attribute1

**attribute2**

public String attribute2

**attribute3**

public String attribute3

**attribute4**

public String attribute4

**attribute5**

public String attribute5

**attribute6**

public String attribute6

**attribute7**

public String attribute7

**attribute8**

public String attribute8

**attribute9**

public String attribute9

**attribute10**

public String attribute10

**attribute11**

public String attribute11

**attribute12**

public String attribute12



**attribute13**

```
public String attribute13
```

**attribute14**

```
public String attribute14
```

**attribute15**

```
public String attribute15
```

**created\_by**

```
public BigDecimal created_by
```

**creation\_date**

```
public Timestamp creation_date
```

**last\_update\_date**

```
public Timestamp last_update_date
```

**last\_update\_login**

```
public BigDecimal last_update_login
```

**last\_updated\_by**

```
public BigDecimal last_updated_by
```

**operation\_code**

```
public String operation_code
```

**orig\_tax\_code**

```
public String orig_tax_code
```

**program\_application\_id**

```
public BigDecimal program_application_id
```

**program\_id**

```
public BigDecimal program_id
```

**program\_update\_date**

public Timestamp program\_update\_date

**qte\_line\_index**

public BigDecimal qte\_line\_index

**quote\_header\_id**

public BigDecimal quote\_header\_id

**quote\_line\_id**

public BigDecimal quote\_line\_id

**quote\_shipment\_id**

public BigDecimal quote\_shipment\_id

**RCS\_ID**

public static final String RCS\_ID

**request\_id**

public BigDecimal request\_id

**shipment\_index**

public BigDecimal shipment\_index

**tax\_amount**

public BigDecimal tax\_amount

**tax\_code**

public String tax\_code

**tax\_date**

public Timestamp tax\_date

**tax\_detail\_id**

public BigDecimal tax\_detail\_id

**tax\_exempt\_flag**

```
public String tax_exempt_flag
```

**tax\_exempt\_number**

```
public String tax_exempt_number
```

**tax\_exempt\_reason\_code**

```
public String tax_exempt_reason_code
```

**tax\_rate**

```
public BigDecimal tax_rate
```

## 9.21.2 Constructors for Class TaxDetailRecord

**TaxDetailRecord**

```
public TaxDetailRecord()
```

## 9.22 Exceptions for Package oracle.apps.ibe.shoppingcart.quote

**SQLException**

Exception class to throw if database error occurs.

**FrameworkException**

Exception class to throw if error occurs while trying to get connection.

**Class ContractException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.shoppingcart.quote.ContractException  
public class ContractException
```

```
extends FrameworkException
```

Exception class to throw if error occurs during Contract class method.

## Variables for ContractException Class

### RCS\_ID

```
public static final String RCS_ID
```

### RCS\_ID\_RECORDED

```
public static final boolean RCS_ID_RECORDED
```

## Constructors for ContractException Class

### ContractException

```
public ContractException(String errorKey)
```

Construct an exception with the errorKey.

### ContractException

```
public ContractException(String errorKey, Object params[])
```

Construct an exception with the errorKey.

**Parameters:** params (an array of tokens for errorKey)

### ContractException

```
public ContractException(Exception e, String errorKey, Object params[])
```

Construct an Exception with the given exception and errorKey.

**Parameters:** e (the parent exception)

params (an array of tokens for errorKey)

### ContractException

```
public ContractException(Exception e, String errorKey, String param)
```

### ContractException

```
public ContractException(Exception e, String errorKey)
```

### ContractException

```
public ContractException(String err_msg, String errorKey, Object params[])
```

Construct an Exception with the errorKey and error message.

**Parameters:** params (an array of tokens for errorKey)

### **ContractException**

```
public ContractException(String err_msg, String errorKey, String param)
```

### **ContractException**

```
public ContractException(String err_msg, String errorKey)
```

### **ContractException**

```
public ContractException(int err_msg_count, String errorKey, Object params[])  
throws FrameworkException
```

Construct an Exception with the errorKey and errors occurred at PL/SQL level.

**Parameters:** err\_msg\_count (the number of messages to be returned from the PL/SQL error stack)

params (an array of tokens for the errorKey)

### **ContractException**

```
public ContractException(int err_msg_count, String errorKey, String param)  
throws FrameworkException
```

### **ContractException**

```
public ContractException(int err_msg_count, String errorKey)  
throws FrameworkException
```

### **Class QuoteException**

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.shoppingcart.quote.QuoteException  
public class QuoteException
```

```
extends FrameworkException
```

Exception class to throw if Quote class method action has already been performed by others or if there is an application error.

## Variables for QuoteException Class

### RCS\_ID

```
public static final String RCS_ID
```

### RCS\_ID\_RECORDED

```
public static final boolean RCS_ID_RECORDED
```

## Constructors for QuoteException Class

### QuoteException

```
public QuoteException(String errorKey)
```

Construct an exception with the errorKey.

### QuoteException

```
public QuoteException(String errorKey, Object params[])
```

Construct an exception with the errorKey.

**Parameters:** params (an array of tokens for errorKey )

### QuoteException

```
public QuoteException(Exception e, String errorKey, Object params[])
```

Construct an Exception with the given exception and errorKey.

**Parameters:** e (the parent exception)

params (an array of tokens for errorKey)

### QuoteException

```
public QuoteException(Exception e, String errorKey, String param)
```

### QuoteException

```
public QuoteException(Exception e, String errorKey)
```

### QuoteException

```
public QuoteException(String err_msg, String errorKey, Object params[])
```

Construct an Exception with the errorKey and error message.

**Parameters:** params (an array of tokens for errorKey)

### **QuoteException**

```
public QuoteException(String err_msg, String errorKey, String param)
```

### **QuoteException**

```
public QuoteException(String err_msg, String errorKey)
```

### **QuoteException**

```
public QuoteException(int err_msg_count, String errorKey, Object params[])  
throws FrameworkException
```

Construct an Exception with the errorKey and errors occurred at PL/SQL level.

**Parameters:** err\_msg\_count (the number of messages to be returned from the PL/SQL error stack)

params (an array of tokens for the errorKey)

### **QuoteException**

```
public QuoteException(int err_msg_count, String errorKey, String param)  
throws FrameworkException
```

### **QuoteException**

```
public QuoteException(int err_msg_count, String errorKey)  
throws FrameworkException
```





---

---

## Oracle iStore 11*i* Postsales APIs

This chapter contains the following information on the Oracle iStore 11*i* Postsales public class APIs:

- [Postsales API Class Summary](#)
- [Class AkCurrencyFormatter](#)
- [Class AkDateFormatter](#)
- [Class AkQuery](#)
- [Class AkQueryCondition](#)
- [Class AkRegion](#)
- [Class IbeAtpPvt](#)
- [Class Query](#)
- [Class QueryCondition](#)
- [Class QueryUtil](#)
- [Class QueryValidatorException](#)
- [Examples of Customizations with the Postsales APIs](#)

## 10.1 Postsales API Class Summary

APIs for the Oracle iStore 11i Postsales procedures are located in the package `oracle.apps.ibe.postsales`. The table below describes the classes briefly.

**Table 10–1 Postsales Class Summary**

Class Name	Description
Class <a href="#">AkCurrencyFormatter</a>	AkCurrencyFormatter implements the QueryFormatter interface for AkQuery objects.
Class <a href="#">AkDateFormatter</a>	AkDateFormatter implements the QueryFormatter interface for AkQuery objects.
Class <a href="#">AkQuery</a>	The AkQuery class extends the Query abstract class based on AK regions.
Class <a href="#">AkQueryCondition</a>	AkQueryCondition encapsulates a single condition in the WHERE clause of an @see AkQuery.
Class <a href="#">AkRegion</a>	The AkRegion class encapsulates AK regions for postsales (e.g. Order Tracker) queries.
Class <a href="#">IbeAtpPvt</a>	IbeAtpPvt is a Rosetta-generated Java wrapper for the IBE_ATP_PVT PL/SQL package.
Class <a href="#">Query</a>	The Query abstract class specifies the minimum functionality required for database queries on which the iStore postsales pages (e.g. Order Tracker) depend.
Class <a href="#">QueryCondition</a>	QueryCondition encapsulates a single condition in the WHERE clause of a @see Query.
Class <a href="#">QueryUtil</a>	The QueryUtil class contains basic utility methods for postsales, such as null handling methods, date validation methods, and methods to check if a given value is a valid number.
Class <a href="#">QueryValidatorException</a>	Exception class that extends the FrameworkException class and is thrown by all postsales interaction query methods.

## 10.2 Class AkCurrencyFormatter

```
java.lang.Object > oracle.apps.ibe.postsales.AkCurrencyFormatter
public final class AkCurrencyFormatter
extends Object
Implements QueryFormatter
```

AkCurrencyFormatter implements the QueryFormatter interface for AkQuery objects.

**See Also:** QueryFormatter, AkQuery

## 10.2.1 Variables for Class AkCurrencyFormatter

### RCS\_ID

```
public static final String RCS_ID
```

### RCS\_ID\_RECORDED

```
public static final boolean RCS_ID_RECORDED
```

### thisClass

```
public static final String thisClass
```

## 10.2.2 Methods for Class AkCurrencyFormatter

The following table is an index of Class AkCurrencyFormatter methods:

**Table 10–2 Method Index for Class AkCurrencyFormatter**

Method	Description
<a href="#">format</a>	Implements the format method of the QueryFormatter interface
<a href="#">getAkCurrencyFormatter</a>	Factory method for AkCurrencyFormatter objects
<a href="#">getItemName</a>	Returns the name of the region item which holds the amount to be formatted

### format

```
public final String format(ResultSet rs)
throws SQLException, FrameworkException
```

Implements the format method of the QueryFormatter interface. Given a JDBC result set, formats the amount in the amount column according to the currency code in the currency column (if specified), or the global currency code. Returns a string containing the formatted amount. Uses the formatNumber method of the oracle.apps.ibe.catalog.PriceObject class to do the work.

**Parameters:** rs (result set containing the row with the amount to be formatted)

**Returns:** a string containing the formatted amount (or null if amount is null)

**Throws:** SQLException if a database exception occurs while fetching from the result set

**Throws:** FrameworkException if the amount cannot be formatted

### **getAkCurrencyFormatter**

```
public static final AkCurrencyFormatter getAkCurrencyFormatter(Query q, String  
currencyItemName, String amountItemName)  
throws FrameworkException
```

Factory method for AkCurrencyFormatter objects. Returns a formatter object that formats the amount in the specified region item using the currency code in the specified region item. If such a formatter already exists and the object cache is enabled, returns the cached object; otherwise creates a new object.

**Parameters:** q (query whose results are to be formatted)

currencyItemName (name of region item specifying the currency code)

amountItemName (name of region item containing the amount)

**Returns:** an AkCurrencyFormatter object (possibly a cached copy)

**Throws:** FrameworkException if either region item name is invalid

### **getAkCurrencyFormatter**

```
public static final AkCurrencyFormatter getAkCurrencyFormatter(Query q, String  
amountItemName)  
throws FrameworkException
```

Factory method for AkCurrencyFormatter objects. Returns a formatter object that formats the amount in the specified region item using the currency code from the cookie. If such a formatter already exists and the object cache is enabled, returns the cached object; otherwise creates a new object. Note that this method will only provide useful formatter objects in a single- currency store where the currency code stored in the cookie is valid for all amounts displayed.

**Parameters:** q (query whose results are to be formatted)

amountItemName (name of region item containing the amount)

**Returns:** an AkCurrencyFormatter object (possibly a cached copy)

**Throws:** FrameworkException if the amount item name is invalid

**getItemName**

```
public final String getItemName()
```

Returns the name of the region item which holds the amount to be formatted.

**Returns:** name of AK region item containing the amount to be formatted

## 10.3 Class AkDateFormatter

```
java.lang.Object > oracle.apps.ibe.postsales.AkDateFormatter
```

```
public class AkDateFormatter
```

```
extends Object
```

```
implements QueryFormatter
```

AkDateFormatter implements the QueryFormatter interface for AkQuery objects.

**See Also:** QueryFormatter, AkQuery

### 10.3.1 Variables for Class AkDateFormatter

**RCS\_ID**

```
public static final String RCS_ID
```

**RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

**thisClass**

```
public static final String thisClass
```

### 10.3.2 Methods for Class AkDateFormatter

The following table is an index of Class AkDateFormatter methods:

**Table 10–3 Method Index for Class AkDateFormatter**

Method	Description
<a href="#">format</a>	Implements the format method of the QueryFormatter interface
<a href="#">getAkDateFormatter</a>	Factory method for AkDateFormatter objects

**Table 10–3 Method Index for Class AkDateFormatter (Cont.)**

Method	Description
<a href="#">getItemName</a>	Returns the name of the region item which holds the date to be formatted

**format**

```
public final String format(ResultSet rs)
throws SQLException, FrameworkException
```

Implements the format method of the QueryFormatter interface. Given a JDBC result set, formats the date in the date column according to the stored date format mask. Returns a string containing the formatted date.

**Parameters:** rs (result set containing the row with the date to be formatted)

**Returns:** a string containing the formatted date (or null if date is null)

**Throws:** SQLException if a database exception occurs while fetching from the result set

**Throws:** FrameworkException if the date cannot be formatted

**getAkDateFormatter**

```
public static final AkDateFormatter getAkDateFormatter(Query q, String
dateItemName)
throws FrameworkException
```

Factory method for AkDateFormatter objects. Returns a formatter object that formats the date in the specified region item using the default date format from the cookie. If such a formatter already exists and the object cache is enabled, returns the cached object; otherwise creates a new object.

**Parameters:** q (query whose results are to be formatted)

dateItemName (name of region item containing the date)

**Returns:** an AkDateFormatter object (possibly a cached copy)

**Throws:** FrameworkException if either region item name is invalid

**getItemName**

```
public final String getItemName()
```

Returns the name of the region item which holds the date to be formatted.

**Returns:** name of AK region item containing the date to be formatted

## 10.4 Class AkQuery

```
java.lang.Object > oracle.apps.ibe.postsales.Query >
oracle.apps.ibe.postsales.AkQuery
public final class AkQuery
```

extends Query

The AkQuery class extends the Query abstract class based on AK regions.

**See Also:** QueryFormatter, QueryValidator, QueryUtil

### 10.4.1 Variables for Class AkQuery

#### **RCS\_ID**

```
public static final String RCS_ID
```

#### **RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

#### **thisClass**

```
public static final String thisClass
```

### 10.4.2 Constructors for Class AkQuery

#### **AkQuery**

```
public AkQuery()
```

Constructor.

### 10.4.3 Methods for Class AkQuery

The following table is an index of Class AkQuery methods:

**Table 10–4 Method Index for Class AkQuery**

Method	Description
<a href="#">addCondition</a>	Adds a condition to the WHERE clause of the query

**Table 10–4 Method Index for Class AkQuery (Cont.)**

<b>Method</b>	<b>Description</b>
<a href="#">addFormatter</a>	Registers a QueryFormatter object for the specified query item
<a href="#">connect</a>	Connects to the database and initializes the named query
<a href="#">disconnect</a>	Disconnects from the database and cleans up the query
<a href="#">execute</a>	Executes the query
<a href="#">fromURL</a>	Reconstructs the state of a query from URL parameters generated by an HTML form or a call to toURL()
<a href="#">getBatchSize</a>	Returns the maximum number of records to be displayed on a single page
<a href="#">getColumnIndex</a>	Returns the index of the database view column on which the named query item is based
<a href="#">getConditions</a>	Returns a vector of all QueryCondition objects added to the query via the addCondition() methods
<a href="#">getDateFormat</a>	Returns the Oracle date format mask (e.g. DD-MON-YYYY) used by the query
<a href="#">getInvoiceId</a>	
<a href="#">getItemIndex</a>	Returns the index of the query item (i.e. AK region item or view column) with the specified name
<a href="#">getItemLabel</a>	Returns the display label of the query item with the specified parameter, suitable for use as a column heading when rendering the query result table on the page
<a href="#">getItemName</a>	Returns the name of the query item (i.e. AK region item or view column) with the specified index
<a href="#">getName</a>	Returns a string that uniquely identifies the region (along with the responsibility ID, application ID, and language code used when connecting to the region)
<a href="#">getNumItems</a>	Returns the number of items (i.e. fields) in a record returned by the query
<a href="#">getNumRowsFetched</a>	Returns the total number of rows fetched by the last call to execute()
<a href="#">getNumRowsShown</a>	Returns the actual number of records to be displayed on the current page
<a href="#">getStartRow</a>	Returns the index of the first row to be displayed on the current page



**Table 10–4 Method Index for Class AkQuery (Cont.)**

Method	Description
<a href="#">getValue</a>	Returns the string representation of the value in the specified cell of the query result table
<a href="#">isDisplayable</a>	Returns true if the specified query item may be displayed
<a href="#">isQueryable</a>	Returns true if the specified query item may be searched on
<a href="#">isSelectable</a>	Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause
<a href="#">resetConditions</a>	Removes all conditions from the WHERE clause of the query
<a href="#">setBatchSize</a>	Sets the batch size (i.e. the maximum number of records shown per page) to the specified value
<a href="#">setOrderByColumn</a>	Sets the ORDER BY clause of the query to use the specified item and sort direction
<a href="#">setStartRow</a>	Sets the start row to the specified row for execution
<a href="#">showQueryConditions</a>	Returns a properly formatted HTML option list of search conditions
<a href="#">showQueryOperators</a>	Returns a properly formatted HTML option list of search operators
<a href="#">toURL</a>	Externalizes the current state of the query as a URL parameter string of the form "param1=val1¶m2=val2&...¶mX=valX"

**addCondition**

```
public final void addCondition(int itemIndex, String operatorCode, String value)
throws FrameworkException
```

Adds a condition to the WHERE clause of the query. Successive calls to `addCondition()` add new conditions to the WHERE clause in order. The condition is of the form , e.g. CUST\_ACCOUNT\_ID = 1001. The specified item must be one for which `isSelectable()` returns true. The specified operator must be one of the valid operators returned by `showQueryOperators()`. For queries based on AK regions, valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. The value may be any free-form String, but it must be convertible to the specified item's datatype. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

operatorCode (valid binary operator code)

value (value against which item is to be compared)

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** addCondition in class Query

**See Also:** addCondition, isSelectable, showQueryOperators, connect

### **addCondition**

```
public final void addCondition(String itemName, String operatorCode, String value)
throws FrameworkException
```

Adds a condition to the WHERE clause of the query. Successive calls to addCondition() add new conditions to the WHERE clause in order. The condition is of the form , e.g. CUST\_ACCOUNT\_ID = 1001. The specified item must be one for which isSelectable() returns true. The specified operator must be one of the valid operators returned by showQueryOperators(). For queries based on AK regions, valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. The value may be any free-form String, but it must be convertible to the specified item's datatype. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

operatorCode (valid binary operator code)

value (value against which item is to be compared)

**Throws:** FrameworkException

if itemName is not found

**Overrides:** addCondition in class Query

**See Also:** addCondition, isSelectable, showQueryOperators, connect

### **addCondition**

```
public final void addCondition(String condition)
```

Appends the specified string to the WHERE clause of the query as-is, without performing any syntactic or semantic verification. The programmer must ensure that the argument is a valid condition given the specifics of the query. If this method is called before connect(), the result is undefined.

**Parameters:** condition (any valid condition to be appended to the WHERE clause)

**Overrides:** addCondition in class Query

**See Also:** addCondition, addCondition, connect

### **addFormatter**

```
public final void addFormatter(QueryFormatter formatter)
throws FrameworkException
```

Registers a QueryFormatter object for the specified query item. As it fetches data from the database, the execute() method invokes the appropriate QueryFormatter objects to format specific query items in the desired way.

**Parameters:** formatter (the QueryFormatter object to be registered)

itemIndex (index of the query item to be formatted)

**Overrides:** addFormatter in class Query

### **connect**

```
public final void connect(String name)
throws FrameworkException
```

Connects to the database and initializes the named query. Classes extending the Query abstract class based on AK regions should interpret the name parameter as the name of an AK region; classes based directly on database views should interpret name as the name of a database view. Names are case-sensitive. The programmer must ensure that queries initialized by calling connect() are properly cleaned up by calling disconnect() when finished.

**Parameters:** name (name of the query, e.g. AK region name or view name)

**Throws:** FrameworkException if an error occurs while connecting to the database and/or initializing the query

**Overrides:** connect in class Query

**See Also:** disconnect

### **disconnect**

```
public final void disconnect()
```

Disconnects from the database and cleans up the query. The programmer must ensure that queries initialized by calling `connect()` are properly cleaned up by calling `disconnect()` when finished.

**Overrides:** `disconnect` in class `Query`

**See Also:** `connect`

### **execute**

```
public final void execute()  
throws FrameworkException, SQLException
```

Executes the query.

**Throws:** `FrameworkException` if an error occurred while fetching and/or formatting the query results

**Overrides:** `execute` in class `Query`

### **fromURL**

```
public final void fromURL(ServletRequest request)  
throws FrameworkException, QueryValidatorException
```

Reconstructs the state of a query from URL parameters generated by an HTML form or a call to `toURL()`. This method must be called after calling `connect()`.

**Parameters:** `request` (the HTTP request containing the URL parameters)

**Throws:** `FrameworkException` if the data in the URL parameters is corrupt and/or some error occurred while attempting to initialize the query object

**Overrides:** `fromURL` in class `Query`

### **getBatchSize**

```
public final int getBatchSize()
```

Returns the maximum number of records to be displayed on a single page. If this method is invoked before `connect()`, it returns 0.

**Returns:** maximum number of rows per page

**Overrides:** `getBatchSize` in class `Query`

**See Also:** `connect`

### **getColumnIndex**

```
public final int getColumnIndex(String itemName)
throws FrameworkException
```

Returns the index of the database view column on which the named query item is based. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** database view column index of query item

**Throws:** FrameworkException if no item with the specified name is found

**Overrides:** getColumnIndex in class Query

**See Also:** connect

### **getConditions**

```
public final Vector getConditions()
```

Returns a vector of all QueryCondition objects added to the query via the addCondition() methods. Used by classes implementing the QueryValidator interface.

**Returns:** vector containing all QueryConditions for this query

**Overrides:** getConditions in class Query

**See Also:** addCondition, addCondition, QueryValidator

### **getDateFormat**

```
public final String getDateFormat()
```

Returns the Oracle date format mask (e.g. DD-MON-YYYY) used by the query. If this method is invoked before connect(), the result is undefined.

**Returns:** date format mask used by the query

**Overrides:** getDateFormat in class Query

**See Also:** connect

### **getInvoiceId**

```
public String getInvoiceId(String lineId)
throws FrameworkException, SQLException
```

### **getItemIndex**

```
public final int getItemIndex(String itemName)
throws FrameworkException
```

Returns the index of the query item (i.e. AK region item or view column) with the specified name. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** index of query item

**Throws:** `FrameworkException` if no item with the specified name is found

**Overrides:** `getItemIndex` in class `Query`

**See Also:** `getItemName`, `connect`

### **getItemLabel**

```
public final String getItemLabel(int itemIndex)
throws FrameworkException
```

Returns the display label of the query item with the specified index, suitable for use as a column heading when rendering the query result table on the page. Item indexes are zero-based. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** item label (i.e. column heading)

**Throws:** `FrameworkException` if `itemIndex` is invalid

**Overrides:** `getItemLabel` in class `Query`

**See Also:** `getItemLabel`, `connect`

### **getItemLabel**

```
public final String getItemLabel(String itemName)
throws FrameworkException
```

Returns the display label of the query item with the specified name, suitable for use as a column heading when rendering the query result table on the page. Item names are case-sensitive. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** item label (i.e. column heading)

**Throws:** FrameworkException if no item with the specified name is found

**Overrides:** getItemLabel in class Query

**See Also:** getItemLabel, connect

### **getItemName**

```
public final String getItemName(int itemIndex)
throws FrameworkException
```

Returns the name of the query item (i.e. AK region item or view column) with the specified index. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** name of query item

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** getItemName in class Query

**See Also:** getItemIndex, connect

### **getName**

```
public final String getName()
```

Returns a string that uniquely identifies the region (along with the responsibility ID, application ID, and language code used when connecting to the region). Used for caching purposes.

**Returns:** string uniquely identifying the region

**Overrides:** getName in class Query

**See Also:** connect

### **getNumItems**

```
public final int getNumItems()
```

Returns the number of items (i.e. fields) in a record returned by the query. Classes extending the Query abstract class based on AK regions should return the number

of region items; classes based directly on database views should return the number of view columns. If this method is invoked before `connect()`, it returns 0.

**Returns:** number of query items

**Overrides:** `getNumItems` in class `Query`

### **getNumRowsFetched**

```
public final int getNumRowsFetched()
```

Returns the total number of rows fetched by the last call to `execute()`. Depending on the way the query statement is constructed, this value may be greater than or equal to the value returned by `getNumRowsShown()`. If this method is invoked before `execute()`, it returns 0.

**Returns:** total number of rows fetched

**Overrides:** `getNumRowsFetched` in class `Query`

**See Also:** `getNumRowsShown`, `execute`

### **getNumRowsShown**

```
public final int getNumRowsShown()
```

Returns the actual number of records to be displayed on the current page. This number is always guaranteed to be less than or equal to the value returned by `getBatchSize()`. If this method is invoked before `execute()`, it returns 0.

**Returns:** number of rows shown on current page

**Overrides:** `getNumRowsShown` in class `Query`

**See Also:** `getBatchSize`, `execute`

### **getStartRow**

```
public final int getStartRow()
```

Returns the index of the first row to be displayed on the current page. Row indexes are zero-based, so the index of the first row on the page displaying records 11-20 is actually 10. If this method is invoked before `connect()`, it returns 0.

**Overrides:** `getStartRow` in class `Query`

**See Also:** `connect`



**getValue**

```
public final String getValue(int rowIndex, int itemIndex)
throws FrameworkException
```

Returns the string representation of the value in the specified cell of the query result table. `rowIndex` is zero-based and must be strictly less than the value returned by `getNumRowsShown()`. `itemIndex` is zero-based and must be strictly less than the value returned by `getNumItems()`. Any formatting performed by `QueryFormatters` registered with the query will have already taken place and will be reflected in the value returned by `getValue()`. If this method is invoked before `execute()`, the result is undefined.

**Parameters:** `rowIndex` (row index of the cell to be displayed)

`itemIndex` (column index of the cell to be displayed)

**Returns:** value of the specified table cell

**Throws:** `FrameworkException` if either index is invalid

**Overrides:** `getValue` in class `Query`

**See Also:** `getValue`, `getNumRowsShown`, `getNumItems`, `execute`

**getValue**

```
public final String getValue(int rowIndex, String itemName)
throws FrameworkException
```

Returns the string representation of the value in the specified cell of the query result table. `rowIndex` is zero-based and must be strictly less than the value returned by `getNumRowsShown()`. `itemName` is case-sensitive and must be one of the item names returned by `getItemName()`. Any formatting performed by `QueryFormatters` registered with the query will have already taken place and will be reflected in the value returned by `getValue()`. If this method is invoked before `execute()`, the result is undefined.

**Parameters:** `rowIndex` (row index of the cell to be displayed)

`itemName` (item name of the cell to be displayed)

**Returns:** value of the specified table cell

**Throws:** `FrameworkException` if `rowIndex` is invalid and/or `itemName` is not found

**Overrides:** `getValue` in class `Query`

**See Also:** `getValue`, `getNumRowsShown`, `getItemName`, `execute`

### **isDisplayable**

```
public final boolean isDisplayable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling `getValue()`. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** true if the item should be displayed

**Throws:** `FrameworkException` if `itemIndex` is invalid

**Overrides:** `isDisplayable` in class `Query`

**See Also:** `isDisplayable`, `connect`

### **isDisplayable**

```
public final boolean isDisplayable(String itemName)
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling `getValue()`. Item names are case-sensitive. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** true if the item should be displayed

**Throws:** `FrameworkException` if `itemName` is not found

**Overrides:** `isDisplayable` in class `Query`

**See Also:** `isDisplayable`, `connect`

### **isQueryable**

```
public final boolean isQueryable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** true if the item can be searched on

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** isQueryable in class Query

**See Also:** isQueryable, connect

### isQueryable

```
public final boolean isQueryable(String itemName)
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item names are case-sensitive. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** true if the item can be searched on

**Throws:** FrameworkException

if itemName is not found

**Overrides:** isQueryable in class Query

**See Also:** isQueryable, connect

### isSelectable

```
public final boolean isSelectable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause. Classes extending the Query abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item indexes are zero-based. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** true if the item is based on a database object

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** isSelectable in class Query

**See Also:** isSelectable, isQueryable, connect

### **isSelectable**

```
public final boolean isSelectable(String itemName)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause. Classes extending the query abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item names are case-sensitive. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** true if the item is based on a database object

**Throws:** FrameworkException if itemName is not found

**Overrides:** isSelectable in class Query

**See Also:** isSelectable, isQueryable, connect

### **resetConditions**

```
public final void resetConditions()
```

Removes all conditions from the WHERE clause of the query. If this method is called before connect(), the result is undefined.

**Overrides:** resetConditions in class Query

**See Also:** addCondition, addCondition, connect

### **setBatchSize**

```
public final void setBatchSize(int batchSize)
throws FrameworkException
```

Sets the batch size (i.e. the maximum number of records shown per page) to the specified value. batchSize must be a non-negative integer.

**Parameters:** batchSize (new number of rows per page)

**Throws:** FrameworkException if batchSize is not a non-negative integer

**Overrides:** setBatchSize in class Query

### setOrderByColumn

```
public final void setOrderByColumn(int itemIndex, boolean isAscending)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item and sort direction. The item must be one for which isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

isAscending (true if the sort direction should be ascending)

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** setOrderByColumn in class Query

**See Also:** setOrderByColumn, isSelectable, connect

### setOrderByColumn

```
public final void setOrderByColumn(String itemName, boolean isAscending)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item and sort direction. The item must be one for which isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (name of query item)

isAscending (true if the sort direction should be ascending)

**Throws:** FrameworkException if itemName is not found

**Overrides:** setOrderByColumn in class Query

**See Also:** setOrderByColumn, isSelectable, connect

### setOrderByColumn

```
public final void setOrderByColumn(int itemIndex)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item. If the ORDER BY clause is already using this item, reverses the sort direction. If the ORDER BY clause is empty or is using a different item, sets the ORDER BY clause to use this item and sets the sort direction to ascending by default. The item must be one for which

isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Throws:** FrameworkException if itemIndex is invalid

**Overrides:** setOrderByColumn in class Query

**See Also:** setOrderByColumn, isSelectable, connect

### **setOrderByColumn**

```
public final void setOrderByColumn(String itemName)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item. If the ORDER BY clause is already using this item, reverses the sort direction. If the ORDER BY clause is empty or is using a different item, sets the ORDER BY clause to use this item and sets the sort direction to ascending by default. The item must be one for which isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Throws:** FrameworkException if itemName is not found

**Overrides:** setOrderByColumn in class Query

**See Also:** setOrderByColumn, isSelectable, connect

### **setStartRow**

```
public final void setStartRow(int rowIndex)
throws FrameworkException
```

Sets the start row to the specified row for execution. rowIndex must be a non-negative integer.

**Parameters:** rowIndex (index of the start row)

**Throws:** FrameworkException if rowIndex is not a non-negative integer

**Overrides:** setStartRow in class Query

### **showQueryConditions**

```
public final String showQueryConditions()
```

Returns a properly formatted HTML option list of search conditions. Each query item for which `isQueryable()` returns true represents a valid search condition. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. If `preset` is non-NULL and specifies one of the values in the list, the corresponding option will be automatically selected. If this method is called before `connect()`, the result is undefined.

**Parameters:** `preset` (preset value for the option list)

**Returns:** a list of OPTION tags specifying valid query criteria

**Overrides:** `showQueryConditions` in class `Query`

**See Also:** `isQueryable`, `connect`

### **showQueryOperators**

```
public final String showQueryOperators()  
throws FrameworkException
```

Returns a properly formatted HTML option list of search operators. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. If `preset` is non-NULL and specifies one of the values in the list, the corresponding option will be automatically selected. For queries based on AK regions, the valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. If this method is called before `connect()`, the result is undefined.

**Parameters:** `preset` (preset value for the option list)

**Returns:** a list of OPTION tags specifying valid query operators

**Overrides:** `showQueryOperators` in class `Query`

**See Also:** `connect`

### **toURL**

```
public final String toURL()
```

Externalizes the current state of the query as a URL parameter string of the form "param1=val1¶m2=val2&...¶mX=valX". When creating hyperlinks from one query display page to another, programmers must ensure that the string returned by `toURL()` is appended to the URL parameters of the link.

**Overrides:** `toURL` in class `Query`

## 10.5 Class AkQueryCondition

```
java.lang.Object > oracle.apps.ibe.postsales.QueryCondition >  
oracle.apps.ibe.postsales.AkQueryCondition  
public final class AkQueryCondition  
extends QueryCondition
```

AkQueryCondition encapsulates a single condition in the WHERE clause of an @see AkQuery. The WHERE clause of an AkQuery is a conjunction of zero or more AkQueryConditions. Note that AkQueryCondition does not support disjunctions of conditions or joins between database objects.

**See Also:** AkQuery, Region, AkRegionItem

### 10.5.1 Variables for Class AkQueryCondition

#### **\_itemIndex**

```
protected int _itemIndex
```

#### **\_operatorCode**

```
protected String _operatorCode
```

#### **RCS\_ID**

```
public static final String RCS_ID
```

#### **RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

#### **thisClass**

```
public static final String thisClass
```

#### **\_value**

```
protected String _value
```

### 10.5.2 Constructors for Class AkQueryCondition

#### **AkQueryCondition**

```
public AkQueryCondition(int itemIndex, String operatorCode, String value)
```



Constructs a AkQueryCondition instance and initializes it with the values provided.

**Parameters:** itemIndex (zero-based index of the query item)  
operatorCode (SQL binary operator)  
value (value to compare against)

### 10.5.3 Methods for Class AkQueryCondition

The following table is an index of Class AkQueryCondition methods:

**Table 10–5 Method Index for Class AkQueryCondition**

Method	Description
<a href="#">getItemIndex</a>	Returns the item index
<a href="#">getOperatorCode</a>	Returns the operator code
<a href="#">getValue</a>	Returns the value

#### **getItemIndex**

```
public final int getItemIndex()
```

Returns the item index

**Overrides:** getItemIndex in class QueryCondition

#### **getOperatorCode**

```
public final String getOperatorCode()
```

Returns the operator code

**Overrides:** getOperatorCode in class QueryCondition

#### **getValue**

```
public final String getValue()
```

Returns the value

**Overrides:** getValue in class QueryCondition

## 10.6 Class AkRegion

```
java.lang.Object > oracle.apps.ibe.postsales.AkRegion  
public final class AkRegion
```

```
extends Object
```

The AkRegion class encapsulates AK regions for post-sales (e.g. Order Tracker) queries.

### 10.6.1 Variables for Class AkRegion

#### **RCS\_ID**

```
public static final String RCS_ID
```

#### **RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

#### **thisClass**

```
public static final String thisClass
```

### 10.6.2 Methods for Class AkRegion

The following table is an index of Class AkRegion methods:

**Table 10–6 Method Index for Class AkRegion**

<b>Method</b>	<b>Description</b>
<a href="#">getAkRegion</a>	Returns an AkRegion object instance based on the region name, the responsibility ID, the application ID, and the language code
<a href="#">getColumnIndex</a>	
<a href="#">getColumnName</a>	
<a href="#">getColumnType</a>	
<a href="#">getItemIndex</a>	Returns the index of the AK region item with the specified name
<a href="#">getItemLabel</a>	Returns the display label of the region item with the specified index, suitable for use as a column heading when rendering the query result table on the page

**Table 10–6 Method Index for Class AkRegion (Cont.)**

Method	Description
<a href="#">getItemName</a>	Returns the name of the AK region item with the specified index
<a href="#">getNumItems</a>	Returns the total number of region items in the AK region
<a href="#">getRegionName</a>	Returns the name of the AK region
<a href="#">getViewName</a>	Returns the name of the view on which the AK region is based
<a href="#">isDisplayable</a>	Returns true if the specified query item may be displayed
<a href="#">isQueryable</a>	Returns true if the specified query item may be searched on
<a href="#">isSelectable</a>	Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause
<a href="#">showQueryConditions</a>	Returns a properly formatted HTML option list of search conditions based on queryable AK region items
<a href="#">showQueryOperators</a>	Returns a properly formatted HTML option list of search operators

**getAkRegion**

```
public static final AkRegion getAkRegion(OracleConnection conn, String
regionName, int respId, int appId, String langCode)
throws SQLException, FrameworkException
```

Returns an AkRegion object instance based on the region name, the responsibility ID, the application ID, and the language code

**getColumnIndex**

```
public final int getColumnIndex(int itemIndex)
throws FrameworkException
```

**getColumnIndex**

```
public final int getColumnIndex(String itemName)
throws FrameworkException
```

**getColumnName**

```
public final String getColumnName(int itemIndex)
throws FrameworkException
```

### **getColumnName**

```
public final String getColumnName(String itemName)
throws FrameworkException
```

### **getColumnType**

```
public final String getColumnType(int itemIndex)
throws FrameworkException
```

### **getColumnType**

```
public final String getColumnType(String itemName)
throws FrameworkException
```

### **getItemIndex**

```
public final int getItemIndex(String itemName)
throws FrameworkException
```

Returns the index of the AK region item with the specified name. Item indexes are zero-based; item names are case-sensitive.

**Parameters:** itemName (name of AK region item)

**Returns:** index of AK region item

**Throws:** FrameworkException if no item with the specified name is found

**See Also:** getItemName

### **getItemLabel**

```
public final String getItemLabel(int itemIndex)
throws FrameworkException
```

Returns the display label of the region item with the specified index, suitable for use as a column heading when rendering the query result table on the page. Item indexes are zero-based.

**Parameters:** itemIndex (index of region item)

**Returns:** item label (i.e. column heading)

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** getItemLabel, connect

**getItemLabel**

```
public final String getItemLabel(String itemName)
throws FrameworkException
```

Returns the display label of the region item with the specified name, suitable for use as a column heading when rendering the query result table on the page. Item names are case-sensitive.

**Parameters:** itemName (name of region item)

**Returns:** item label (i.e. column heading)

**Throws:** FrameworkException if no item with the specified name is found

**See Also:** getItemLabel, connect

**getItemName**

```
public final String getItemName(int itemIndex)
throws FrameworkException
```

Returns the name of the AK region item with the specified index. Item indexes are zero-based; item names are case-sensitive.

**Parameters:** itemIndex (index of AK region item)

**Returns:** name of AK region item

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** getItemIndex

**getNumItems**

```
public final int getNumItems()
```

Returns the total number of region items in the AK region

**Returns:** number of AK region items

**getRegionName**

```
public final String getRegionName()
```

Returns the name of the AK region

**Returns:** name of AK region

### **getViewName**

```
public final String getViewName()
```

Returns the name of the view on which the AK region is based

**Returns:** name of database view

### **isDisplayable**

```
public final boolean isDisplayable(int itemIndex)  
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling `getValue()`. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** true if the item should be displayed

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `isDisplayable`, `connect`

### **isDisplayable**

```
public final boolean isDisplayable(String itemName)  
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling `getValue()`. Item names are case-sensitive. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** true if the item should be displayed

**Throws:** `FrameworkException` if `itemName` is not found

**See Also:** `isDisplayable`, `connect`

### **isQueryable**

```
public final boolean isQueryable(int itemIndex)  
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** true if the item can be searched on

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `isQueryable`, `connect`

### **isQueryable**

```
public final boolean isQueryable(String itemName)
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item names are case-sensitive. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** true if the item can be searched on

**Throws:** `FrameworkException` if `itemName` is not found

**See Also:** `isQueryable`, `connect`

### **isSelectable**

```
public final boolean isSelectable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a `SELECT` or `ORDER BY` clause. Classes extending the `Query` abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** true if the item is based on a database object

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `isSelectable`, `isQueryable`, `connect`

**isSelectable**

```
public final boolean isSelectable(String itemName)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause. Classes extending the query abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item names are case-sensitive. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** true if the item is based on a database object

**Throws:** FrameworkException if itemName is not found

**See Also:** isSelectable, isQueryable, connect

**showQueryConditions**

```
public final String showQueryConditions()
```

Returns a properly formatted HTML option list of search conditions based on queryable AK region items. Each queryable region item is assumed to represent a valid search condition. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. If preset is non-NULL and specifies one of the values in the list, the corresponding option will be automatically selected.

**Parameters:** preset (preset value for the option list)

**Returns:** a list of OPTION tags specifying valid query criteria

**See Also:** connect

**showQueryOperators**

```
public final String showQueryOperators()
```

Returns a properly formatted HTML option list of search operators. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. If preset is non-NULL and specifies one of the values in the list, the corresponding option will be automatically selected. For queries based on AK regions, the valid operators are AIS, BNOT, CCONTAIN,



DSTART, EEND, FGREATER, and GLESS. If this method is called before connect(), the result is undefined.

**Returns:** a list of OPTION tags specifying valid query operators

## 10.7 Class IbeAtpPvt

```
java.lang.Object > oracle.apps.ibe.postsales.IbeAtpPvt
public class IbeAtpPvt
```

extends Object

IbeAtpPvt is a Rosetta-generated Java wrapper for the IBE\_ATP\_PVT PL/SQL package. It contains a single static class (AtpLineTyp), which is a wrapper for the PL/SQL type IBE\_ATP\_PVT.Atp\_Line\_Typ, as well as a static method (CheckAvailability), which is a wrapper to call the PL/SQL procedure IBE\_ATP\_PVT.Check\_Availability.

### 10.7.1 Variables for Class IbeAtpPvt

#### **RCS\_ID**

```
public static final String RCS_ID
```

### 10.7.2 Constructors for Class IbeAtpPvt

#### **IbeAtpPvt**

```
public IbeAtpPvt()
```

### 10.7.3 Methods for Class IbeAtpPvt

The following table is an index of Class IbeAtpPvt methods:

**Table 10–7 Method Index for Class IbeAtpPvt**

Method	Description
<a href="#">checkAvailability</a>	This is a Rosetta-generated method to call the PL/SQL procedure IBE_ATP_PVT.Check_Availability
<a href="#">writeOrSkip</a>	

### checkAvailability

```
public static void checkAvailability(OracleConnection _connection, BigDecimal p_
quote_header_id, String p_date_format, String p_lang_code, String x_error_
flag[], String x_error_message[], IbeAtpPvt. AtpLineTyp x_atp_line_tbl[][])
throws SQLException
```

This is a Rosetta-generated method to call the PL/SQL procedure IBE\_ATP\_PVT.Check\_Availability. The parameter `x_atp_line_tbl` is a nested array of `IbeAtpPvt.AtpLineTyp` objects. It must be declared by the caller as `IbeAtpPvt.AtpLineTyp[][] x_atp_line_tbl = new IbeAtpPvt.AtpLineTyp[1][[]]`. For each line, the following fields must be populated: `quote_line_id`, `organization_id`, `inventory_item_id`, `quantity`, and `uom_code`. The `request_date` parameter is optional; if null or the empty string, the API assumes SYSDATE. The `checkAvailability()` method populates the following fields for each quote line: `request_date_quantity` and `available_date`.

**Parameters:** `_connection` (database connection--IN)

`p_quote_header_id` (shopping cart ID--IN)

`p_date_format` (SQL date format string--IN)

`p_lang_code` (user's language--IN)

`x_error_flag` (Y if an error occurred, N otherwise--OUT)

`x_error_message` (description of the error--OUT)

`x_atp_line_tbl` (array of quote lines--IN/OUT)

**Throws:** `SQLException` if a database or SQL error occurs

### writeOrSkip

```
protected static boolean writeOrSkip(StringBuffer sb, boolean isGMiss, boolean
anyWritten, int numSkipped, String argName)
```

## 10.8 Class Query

```
java.lang.Object > oracle.apps.ibe.postsales.Query
```

```
public abstract class Query
```

```
extends Object
```

The `Query` abstract class specifies the minimum functionality required for database queries on which the iStore post-sales pages (e.g. Order Tracker) depend.

See Also: [QueryFormatter](#), [QueryValidator](#), [QueryUtil](#)

## 10.8.1 Variables for Class Query

### RCS\_ID

```
public static final String RCS_ID
```

### RCS\_ID\_RECORDED

```
public static final boolean RCS_ID_RECORDED
```

## 10.8.2 Constructors for Class Query

### Query

```
public Query()
```

## 10.8.3 Methods for Class Query

The following table is an index of Class Query methods:

**Table 10–8 Method Index for Class Query**

Method	Description
<a href="#">addCondition</a>	Adds a condition to the WHERE clause of the query
<a href="#">addFormatter</a>	Registers a QueryFormatter object for the specified query item
<a href="#">connect</a>	Connects to the database and initializes the named query
<a href="#">disconnect</a>	Disconnects from the database and cleans up the query
<a href="#">execute</a>	Executes the query
<a href="#">fromURL</a>	Reconstructs the state of a query from URL parameters generated by an HTML form or a call to toURL()
<a href="#">getBatchSize</a>	Returns the maximum number of records to be displayed on a single page
<a href="#">getColumnIndex</a>	Returns the index of the database view column on which the named query item is based
<a href="#">getConditions</a>	Returns a vector of all QueryCondition objects added to the query via the addCondition() methods

**Table 10–8 Method Index for Class Query (Cont.)**

<b>Method</b>	<b>Description</b>
<a href="#">getDateFormat</a>	Returns the Oracle date format mask (e.g. DD-MON-YYYY) used by the query
<a href="#">getItemIndex</a>	Returns the index of the query item (i.e. AK region item or view column) with the specified name
<a href="#">getItemLabel</a>	Returns the display label of the query item with the specified index, suitable for use as a column heading when rendering the query result table on the page
<a href="#">getItemName</a>	Returns the name of the query item (i.e. AK region item or view column) with the specified index
<a href="#">getName</a>	Returns the name of the query
<a href="#">getNumItems</a>	Returns the number of items (i.e. fields) in a record returned by the query
<a href="#">getNumRowsFetched</a>	Returns the total number of rows fetched by the last call to execute()
<a href="#">getNumRowsShown</a>	Returns the actual number of records to be displayed on the current page
<a href="#">getStartRow</a>	Returns the index of the first row to be displayed on the current page
<a href="#">getValue</a>	Returns the string representation of the value in the specified cell of the query result table
<a href="#">isDisplayable</a>	Returns true if the specified query item may be displayed
<a href="#">isQueryable</a>	Returns true if the specified query item may be searched on
<a href="#">isSelectable</a>	Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause
<a href="#">resetConditions</a>	Removes all conditions from the WHERE clause of the query
<a href="#">setBatchSize</a>	Sets the batch size (i.e. the maximum number of records shown per page) to the specified value
<a href="#">setOrderByColumn</a>	Sets the ORDER BY clause of the query to use the specified item and sort direction
<a href="#">setStartRow</a>	Sets the start row to the specified row for execution
<a href="#">showQueryConditions</a>	Returns a properly formatted HTML option list of search conditions

**Table 10–8 Method Index for Class Query (Cont.)**

Method	Description
<a href="#">showQueryOperators</a>	Returns a properly formatted HTML option list of search operators
<a href="#">toURL</a>	Externalizes the current state of the query as a URL parameter string of the form "param1=val1¶m2=val2&...¶mX=valX"

**addCondition**

```
public abstract void addCondition(int itemIndex, String operatorCode, String
value)
throws FrameworkException
```

Adds a condition to the WHERE clause of the query. Successive calls to `addCondition()` add new conditions to the WHERE clause in order. The condition is of the form `, e.g. CUST_ACCOUNT_ID = 1001`. The specified item must be one for which `isSelectable()` returns true. The specified operator must be one of the valid operators returned by `showQueryOperators()`. For queries based on AK regions, valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. The value may be any free-form String, but it must be convertible to the specified item's datatype. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

`operatorCode` (valid binary operator code)

`value` (value against which item is to be compared)

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `addCondition`, `isSelectable`, `showQueryOperators`, `connect`

**addCondition**

```
public abstract void addCondition(String itemName, String operatorCode, String
value)
throws FrameworkException
```

Adds a condition to the WHERE clause of the query. Successive calls to `addCondition()` add new conditions to the WHERE clause in order. The condition is of the form `, e.g. CUST_ACCOUNT_ID = 1001`. The specified item must be one for which `isSelectable()` returns true. The specified operator must be one of the valid operators returned by `showQueryOperators()`. For queries based on AK regions,

valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. The value may be any free-form String, but it must be convertible to the specified item's datatype. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

operatorCode (valid binary operator code)

value (value against which item is to be compared)

**Throws:** FrameworkException if itemName is not found

**See Also:** addCondition, isSelectable, showQueryOperators, connect

### **addCondition**

```
public abstract void addCondition(String condition)
```

Appends the specified string to the WHERE clause of the query as-is, without performing any syntactic or semantic verification. The programmer must ensure that the argument is a valid condition given the specifics of the query. If this method is called before connect(), the result is undefined.

**Parameters:** condition (any valid condition to be appended to the WHERE clause)

**See Also:** addCondition, addCondition, connect

### **addFormatter**

```
public abstract void addFormatter(QueryFormatter formatter)
throws FrameworkException
```

Registers a QueryFormatter object for the specified query item. As it fetches data from the database, the execute() method invokes the appropriate QueryFormatter objects to format specific query items in the desired way.

**Parameters:** formatter (the QueryFormatter object to be registered)

itemIndex (index of the query item to be formatted)

### **connect**

```
public abstract void connect(String name)
throws FrameworkException
```

Connects to the database and initializes the named query. Classes extending the Query abstract class based on AK regions should interpret the name parameter as

the name of an AK region; classes based directly on database views should interpret name as the name of a database view. Names are case-sensitive. The programmer must ensure that queries initialized by calling `connect()` are properly cleaned up by calling `disconnect()` when finished.

**Parameters:** name (name of the query, e.g. AK region name or view name)

**Throws:** `FrameworkException` if an error occurs while connecting to the database and/or initializing the query

**See Also:** `disconnect`

### **disconnect**

```
public abstract void disconnect()
```

Disconnects from the database and cleans up the query. The programmer must ensure that queries initialized by calling `connect()` are properly cleaned up by calling `disconnect()` when finished.

**See Also:** `connect`

### **execute**

```
public abstract void execute()  
throws FrameworkException, SQLException
```

Executes the query

**Throws:** `FrameworkException` if an error occurred while fetching and/or formatting the query results

### **fromURL**

```
public abstract void fromURL(ServletRequest request)  
throws FrameworkException
```

Reconstructs the state of a query from URL parameters generated by an HTML form or a call to `toURL()`. This method must be called after calling `connect()`.

**Parameters:** request (the HTTP request containing the URL parameters)

**Throws:** `FrameworkException` if the data in the URL parameters is corrupt and/or some error occurred while attempting to initialize the query object

### **getBatchSize**

```
public abstract int getBatchSize()
```

Returns the maximum number of records to be displayed on a single page. If this method is invoked before `connect()`, it returns 0.

**Returns:** maximum number of rows per page

**See Also:** `connect`

### **getColumnIndex**

```
public abstract int getColumnIndex(String itemName)
throws FrameworkException
```

Returns the index of the database view column on which the named query item is based. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** database view column index of query item

**Throws:** `FrameworkException` if no item with the specified name is found

**See Also:** `connect`

### **getConditions**

```
public abstract Vector getConditions()
```

Returns a vector of all `QueryCondition` objects added to the query via the `addCondition()` methods. Used by classes implementing the `QueryValidator` interface.

**Returns:** vector containing all `QueryConditions` for this query

**See Also:** `addCondition`, `addCondition`, `QueryValidator`

### **getDateFormat**

```
public abstract String getDateFormat()
```

Returns the Oracle date format mask (e.g. `DD-MON-YYYY`) used by the query. If this method is invoked before `connect()`, the result is undefined.

**Returns:** date format mask used by the query

**See Also:** `connect`



**getItemIndex**

```
public abstract int getItemIndex(String itemName)
throws FrameworkException
```

Returns the index of the query item (i.e. AK region item or view column) with the specified name. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** index of query item

**Throws:** `FrameworkException` if no item with the specified name is found

**See Also:** `getItemName`, `connect`

**getItemLabel**

```
public abstract String getItemLabel(int itemIndex)
throws FrameworkException
```

Returns the display label of the query item with the specified index, suitable for use as a column heading when rendering the query result table on the page. Item indexes are zero-based. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** item label (i.e. column heading)

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `getItemLabel`, `connect`

**getItemLabel**

```
public abstract String getItemLabel(String itemName)
throws FrameworkException
```

Returns the display label of the query item with the specified name, suitable for use as a column heading when rendering the query result table on the page. Item names are case-sensitive. If this method is invoked before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** item label (i.e. column heading)

**Throws:** FrameworkException if no item with the specified name is found

**See Also:** getItemLabel, connect

### **getItemName**

```
public abstract String getItemName(int itemIndex)
throws FrameworkException
```

Returns the name of the query item (i.e. AK region item or view column) with the specified index. Item indexes are zero-based; item names are case-sensitive. If this method is invoked before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** name of query item

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** getItemIndex, connect

### **getName**

```
public abstract String getName()
```

Returns the name of the query. Classes extending the Query abstract class based on AK regions should return a string that uniquely identifies the region; classes based directly on database views should return a string that uniquely identifies the view. If this method is invoked before connect(), the result is undefined.

**Returns:** name of the query

**See Also:** connect

### **getNumItems**

```
public abstract int getNumItems()
```

Returns the number of items (i.e. fields) in a record returned by the query. Classes extending the Query abstract class based on AK regions should return the number of region items; classes based directly on database views should return the number of view columns. If this method is invoked before connect(), it returns 0.

**Returns:** number of query items

### **getNumRowsFetched**

```
public abstract int getNumRowsFetched()
```

Returns the total number of rows fetched by the last call to `execute()`. Depending on the way the query statement is constructed, this value may be greater than or equal to the value returned by `getNumRowsShown()`. If this method is invoked before `execute()`, it returns 0.

**Returns:** total number of rows fetched

**See Also:** `getNumRowsShown`, `execute`

### **getNumRowsShown**

```
public abstract int getNumRowsShown()
```

Returns the actual number of records to be displayed on the current page. This number is always guaranteed to be less than or equal to the value returned by `getBatchSize()`. If this method is invoked before `execute()`, it returns 0.

**Returns:** number of rows shown on current page

**See Also:** `getBatchSize`, `execute`

### **getStartRow**

```
public abstract int getStartRow()
```

Returns the index of the first row to be displayed on the current page. Row indexes are zero-based, so the index of the first row on the page displaying records 11-20 is actually 10. If this method is invoked before `connect()`, it returns 0.

**See Also:** `connect`

### **getValue**

```
public abstract String getValue(int rowIndex, int itemIndex)  
throws FrameworkException
```

Returns the string representation of the value in the specified cell of the query result table. `rowIndex` is zero-based and must be strictly less than the value returned by `getNumRowsShown()`. `itemIndex` is zero-based and must be strictly less than the value returned by `getNumItems()`. Any formatting performed by `QueryFormatters` registered with the query will have already taken place and will be reflected in the value returned by `getValue()`. If this method is invoked before `execute()`, the result is undefined.

**Parameters:** `rowIndex` (row index of the cell to be displayed)

itemIndex (column index of the cell to be displayed)

**Returns:** value of the specified table cell

**Throws:** FrameworkException if either index is invalid

**See Also:** getValue, getNumRowsShown, getNumItems, execute

### getValue

```
public abstract String getValue(int rowIndex, String itemName)
throws FrameworkException
```

Returns the string representation of the value in the specified cell of the query result table. rowIndex is zero-based and must be strictly less than the value returned by getNumRowsShown(). itemName is case-sensitive and must be one of the item names returned by getItemName(). Any formatting performed by QueryFormatters registered with the query will have already taken place and will be reflected in the value returned by getValue(). If this method is invoked before execute(), the result is undefined.

**Parameters:** rowIndex (row index of the cell to be displayed)

itemName (item name of the cell to be displayed)

**Returns:** value of the specified table cell

**Throws:** FrameworkException if rowIndex is invalid and/or itemName is not found

**See Also:** getValue, getNumRowsShown, getItemName, execute

### isDisplayable

```
public abstract boolean isDisplayable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling getValue(). Item indexes are zero-based. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** true if the item should be displayed

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** isDisplayable, connect

**isDisplayable**

```
public abstract boolean isDisplayable(String itemName)
throws FrameworkException
```

Returns true if the specified query item may be displayed. This is purely a convenience feature; the displaying JSP may decide whether or not to check if a particular item is displayable before calling `getValue()`. Item names are case-sensitive. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** true if the item should be displayed

**Throws:** `FrameworkException` if `itemName` is not found

**See Also:** `isDisplayable`, `connect`

**isQueryable**

```
public abstract boolean isQueryable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item indexes are zero-based. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Returns:** true if the item can be searched on

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `isQueryable`, `connect`

**isQueryable**

```
public abstract boolean isQueryable(String itemName)
throws FrameworkException
```

Returns true if the specified query item may be searched on. Item names are case-sensitive. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Returns:** true if the item can be searched on

**Throws:** `FrameworkException` if `itemName` is not found

**See Also:** `isQueryable`, `connect`

**isSelectable**

```
public abstract boolean isSelectable(int itemIndex)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause. Classes extending the Query abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item indexes are zero-based. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

**Returns:** true if the item is based on a database object

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** isSelectable, isQueryable, connect

**isSelectable**

```
public abstract boolean isSelectable(String itemName)
throws FrameworkException
```

Returns true if the specified query item is based on a database object and thus may be included in a SELECT or ORDER BY clause. Classes extending the query abstract class based on AK regions should return true if the specified AK region item is based on an object attribute; classes based directly on database views should return true if the specified query item is based on a view column. In any case, all queryable items must also be selectable, though the reverse is not necessarily true. Item names are case-sensitive. If this method is called before connect(), the result is undefined.

**Parameters:** itemName (name of query item)

**Returns:** true if the item is based on a database object

**Throws:** FrameworkException if itemName is not found

**See Also:** isSelectable, isQueryable, connect

**resetConditions**

```
public abstract void resetConditions()
```

Removes all conditions from the WHERE clause of the query. If this method is called before connect(), the result is undefined.

**See Also:** addCondition, addCondition, connect

### setBatchSize

```
public abstract void setBatchSize(int batchSize)
throws FrameworkException
```

Sets the batch size (i.e. the maximum number of records shown per page) to the specified value. batchSize must be a non-negative integer.

**Parameters:** batchSize (new number of rows per page)

**Throws:** FrameworkException if batchSize is not a non-negative integer

### setOrderByColumn

```
public abstract void setOrderByColumn(int itemIndex, boolean isAscending)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item and sort direction. The item must be one for which isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (index of query item)

isAscending (true if the sort direction should be ascending)

**Throws:** FrameworkException if itemIndex is invalid

**See Also:** setOrderByColumn, isSelectable, connect

### setOrderByColumn

```
public abstract void setOrderByColumn(String itemName, boolean isAscending)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item and sort direction. The item must be one for which isSelectable() returns true. If this method is called before connect(), the result is undefined.

**Parameters:** itemIndex (name of query item)

isAscending (true if the sort direction should be ascending)

**Throws:** FrameworkException if itemName is not found

**See Also:** `setOrderByColumn`, `isSelectable`, `connect`

### **setOrderByColumn**

```
public abstract void setOrderByColumn(int itemIndex)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item. If the ORDER BY clause is already using this item, reverses the sort direction. If the ORDER BY clause is empty or is using a different item, sets the ORDER BY clause to use this item and sets the sort direction to ascending by default. The item must be one for which `isSelectable()` returns true. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemIndex` (index of query item)

**Throws:** `FrameworkException` if `itemIndex` is invalid

**See Also:** `setOrderByColumn`, `isSelectable`, `connect`

### **setOrderByColumn**

```
public abstract void setOrderByColumn(String itemName)
throws FrameworkException
```

Sets the ORDER BY clause of the query to use the specified item. If the ORDER BY clause is already using this item, reverses the sort direction. If the ORDER BY clause is empty or is using a different item, sets the ORDER BY clause to use this item and sets the sort direction to ascending by default. The item must be one for which `isSelectable()` returns true. If this method is called before `connect()`, the result is undefined.

**Parameters:** `itemName` (name of query item)

**Throws:** `FrameworkException` if `itemName` is not found

**See Also:** `setOrderByColumn`, `isSelectable`, `connect`

### **setStartRow**

```
public abstract void setStartRow(int rowIndex)
throws FrameworkException
```

Sets the start row to the specified row for execution. `rowIndex` must be a non-negative integer.

**Parameters:** `rowIndex` (index of the start row)



**Throws:** FrameworkException if rowIndex is not a non-negative integer

### showQueryConditions

```
public abstract String showQueryConditions()
```

Returns a properly formatted HTML option list of search conditions. Each query item for which `isQueryable()` returns true represents a valid search condition. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. If this method is called before `connect()`, the result is undefined.

**Returns:** a list of OPTION tags specifying valid query criteria

**See Also:** `isQueryable`, `connect`

### showQueryOperators

```
public abstract String showQueryOperators()  
throws FrameworkException
```

Returns a properly formatted HTML option list of search operators. The return value is a string consisting of OPTION tags, suitable to be included between a SELECT tag and its closing tag. For queries based on AK regions, the valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER, and GLESS. If this method is called before `connect()`, the result is undefined.

**Returns:** a list of OPTION tags specifying valid query operators

**Throws:** FrameworkException if there is an error while retrieving valid operators (e.g. from the database)

**See Also:** `connect`

### toURL

```
public abstract String toURL()
```

Externalizes the current state of the query as a URL parameter string of the form "param1=val1¶m2=val2&...¶mX=valX". When creating hyperlinks from one query display page to another, programmers must ensure that the string returned by `toURL()` is appended to the URL parameters of the link.

## 10.9 Class QueryCondition

```
java.lang.Object > oracle.apps.ibe.postsales.QueryCondition
```

```
public abstract class QueryCondition
```

```
extends Object
```

QueryCondition encapsulates a single condition in the WHERE clause of a @see Query. The WHERE clause of a Query is a conjunction of zero or more QueryConditions. Note that QueryCondition does not support disjunctions of conditions or joins between database objects at all.

**See Also:** Query, Region, AkRegionItem

## 10.9.1 Variables for Class QueryCondition

### **RCS\_ID**

```
public static final String RCS_ID
```

### **RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

## 10.9.2 Constructors for Class QueryCondition

### **QueryCondition**

```
public QueryCondition()
```

## 10.9.3 Methods for Class QueryCondition

The following table is an index of Class QueryCondition methods:

**Table 10–9 Method Index for Class QueryCondition**

<b>Method</b>	<b>Description</b>
<a href="#">getItemIndex</a>	
<a href="#">getOperatorCode</a>	
<a href="#">getValue</a>	

### **getItemIndex**

```
public abstract int getItemIndex()
```

**getOperatorCode**

```
public abstract String getOperatorCode()
```

**getValue**

```
public abstract String getValue()
```

## 10.10 Class QueryUtil

```
java.lang.Object > oracle.apps.ibe.postsales.QueryUtil
public final class QueryUtil
    extends Object
```

### 10.10.1 Variables for Class QueryUtil

**RCS\_ID**

```
public static final String RCS_ID
```

**RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

### 10.10.2 Constructors for Class QueryUtil

**QueryUtil**

```
public QueryUtil()
```

### 10.10.3 Methods for Class QueryUtil

The following table is an index of Class QueryUtil methods:

*Table 10–10 Method Index for Class QueryUtil*

Method	Description
<a href="#">boolVal</a>	
<a href="#">doubleVal</a>	
<a href="#">intVal</a>	
<a href="#">isNonNullString</a>	

**Table 10–10 Method Index for Class QueryUtil (Cont.)**

Method	Description
<a href="#">isNullString</a>	
<a href="#">nonNullString</a>	
<a href="#">stringVal</a>	
<a href="#">validDate</a>	
<a href="#">validNumber</a>	

**boolVal**

```
public static final boolean boolVal(String paramVal, boolean defaultVal)
```

**doubleVal**

```
public static final double doubleVal(String paramVal, double defaultVal)
throws FrameworkException
```

**intVal**

```
public static final int intVal(String paramVal, int defaultVal)
throws FrameworkException
```

**isNonNullString**

```
public static final boolean isNonNullString(String s)
```

**isNullString**

```
public static final boolean isNullString(String s)
```

**nonNullString**

```
public static final String nonNullString(String s)
```

**stringVal**

```
public static final String stringVal(String paramVal, String defaultVal)
throws FrameworkException
```

**validDate**

```
public static final boolean validDate(String dateVal, String dateFormat)
```

**validNumber**

```
public static final boolean validNumber(String numVal)
```

## 10.11 Class QueryValidatorException

```
java.lang.Object > java.lang.Throwable > java.lang.Exception >  
oracle.apps.jtf.base.resources.FrameworkException >  
oracle.apps.ibe.postsales.QueryValidatorException  
public class QueryValidatorException  
extends FrameworkException
```

### 10.11.1 Variables for Class QueryValidatorException

**RCS\_ID**

```
public static final String RCS_ID
```

**RCS\_ID\_RECORDED**

```
public static final boolean RCS_ID_RECORDED
```

**thisClass**

```
public static final String thisClass
```

### 10.11.2 Constructors for Class QueryValidatorException

**QueryValidatorException**

```
public QueryValidatorException()
```

**QueryValidatorException**

```
public QueryValidatorException(String s)
```

**QueryValidatorException**

```
public QueryValidatorException(String errorKey, Object params[])
```

Construct an exception with the errorKey.

**Parameters:** params (an array of tokens for errorKey)

## 10.12 Examples of Customizations with the Postsales APIs

Oracle iStore 11i's Order Tracker uses the AK regions for storing the meta-data related to the display of the Orders, Invoices, Shipments and Payments information. There are corresponding Java objects that retrieve the information from these AK regions and display them in Oracle iStore 11i JSPs. All of the Java objects are in the `$IBE_TOP/java/postsales` directory.

### 10.12.1 AkQuery.java

AkQuery.java is the main Java object used by all JSPs. Following are examples of how you can customize Oracle iStore 11i using methods of the Order Tracker Java object AkQuery.java.

All of the examples assume `q` as an instantiated object of AkQuery.

#### 10.12.1.1 The `connect` Method

Following is an example of how to use the `connect` method of the object AkQuery.java.

##### **connect**

This method connects to the database and initializes the named query. The parameter to this method is the name of the AK region.

**Example** `IBE_ORD_SUM_R` is the AK region for displaying the Order Summary data.

To initialize this region and retrieve all of the data related to this region, you can invoke the `connect` method as follows:

```
q.connect("IBE_ORD_SUM_R");
```

#### 10.12.1.2 The `addCondition` Method

Following are examples of how to use the `addCondition` method of the object AkQuery.java.

##### **addCondition(itemName,operatorCode,value)**

Adds a condition to the WHERE clause of the query. Successive calls to `addCondition()` add new conditions to the WHERE clause in order. The condition is of the form:

<itemName> <operator> <value>

**ItemName:** The specified item must be an AK region item that is an Object Attribute.

**Operator Code:** For queries based on AK regions, valid operators are AIS, BNOT, CCONTAIN, DSTART, EEND, FGREATER and GLESS.

**Value:** The value may be any free-form String, but it must be convertible to the specified item's data type.

**Example** To add a where clause with `Cust_account_id = Customer ID` from the cookie, for the Order Summary page, you can invoke the `addCondition` method as follows:

```
q.addCondition("IBE_OS_CUST_ACCOUNT_ID", "AIS",
RequestCtx.getAccountId().toString());
```

where `IBE_OS_CUST_ACCOUNT_ID` is the region item name which represents `Cust_account_id`.

### **addCondition((String condition)**

Appends the specified string to the WHERE clause of the query as is, without performing any syntactic or semantic verification. The programmer must ensure that the argument is a valid condition given the specifics of the query.

**Example** An additional column has been added to the view and there is no AK region item specified for that column. If you want to use the column in the where clause, you can use the above method. There is a column called "Order\_category\_code" in the view `IBE_ORDER_SUM_V` for which an AK region item has not been defined. If you want to use the `Order_category_code` in the where clause, then you can invoke the `addCondition` method as follows:

```
q.addCondition("Order_category_code = 'RETURN' ");
```

#### **10.12.1.3 The addFormatter Method**

Following are examples of how to use the `addFormatter` method of the object `AkQuery.java`.

### **addFormatter**

Registers a QueryFormatter object for the specified query item. A QueryFormatter object could be a DateFormatter or CurrencyFormatter. For more details, refer to the the API documentation for the QueryFormatter, AkDateFormatter and AkCurrencyFormatter classes.

**Example** Currency Formatting: IBE\_OH\_ORDER\_TOTAL is an AK region item that corresponds to the amount column, which requires currency formatting. You can invoke the addFormatter method as follows:

```
q.addFormatter(AkCurrencyFormatter.getAkCurrencyFormatter(q, "IBE_OH_ORDER_TOTAL"));
```

**Example** Date Formatting: IBE\_OS\_ORDER\_DATE is an AK region item that corresponds to the Date column, which requires date formatting. You can invoke the addFormatter method as follows:

```
q.addFormatter(AkDateFormatter.getAkDateFormatter(q, "IBE_OS_ORDER_DATE"));
```

#### **10.12.1.4 The setOrderByColumn Method**

Following is an example of how to use the setOrderByColumn method of the object AkQuery.java.

### **setOrderByColumn**

Sets the ORDER BY clause of the query to use the specified item and sort direction. The item should be an AK region item. The sort direction should be true for Ascending order and false for Descending order.

**Example** In the Order Summary page, if the query should have an Order By with Order Number in Descending direction, you can invoke the setOrderByColumn method as follows:

```
q.setOrderByColumn("IBE_OS_ORDER_NUMBER", false);
```

where IBE\_OS\_ORDER\_NUMBER is an AK region item for the Order\_number column.



---

---

## Diagnostics and Troubleshooting

This section contains instructions on error corrections and workarounds for problems that you may encounter in configuration or administration of Oracle iStore 11*i*. Topics include:

- [Java Applet Warning Workaround](#)
- [Error ORA-29868 While Executing amviccn.sql](#)
- [Display Manager Errors](#)
- [Catalog and Pricing Errors](#)
- [Shopping List Errors](#)
- [Search Errors](#)
- [Postsales Errors](#)
- [Potential Issues Installing Oracle8i interMedia Text Version 8.1.7](#)
- [Reporting Issues](#)

## 11.1 Java Applet Warning Workaround

Use the following procedure to remove the yellow bar displaying a java applet warning.

1. Uninstall JInitiator from your client machine: in Windows 95/98/NT, go to **Start > Settings > Control Panel > Add/Remove Programs**. Find Oracle JInitiator 1.1.7.27 Export, and click **Add/Remove**. Make sure JInitiator is removed successfully.

2. Delete the identitydb.obj file from your client machine (it is usually in the parent directory of your JInitiator installation, i.e., C:\Program Files\Oracle\).

3. From your client machine, FTP to the server.

```
cd <COMMON_TOP>/html
```

where <COMMON\_TOP> is the actual value (i.e. /u04/viscomn). You cannot use environment variables in FTP. Make sure you are using binary transfer mode: bin.

4. Download oajinit.exe to your client machine:

```
get oajinit.exe
```

5. Change to <APPL\_TOP>/admin, where <APPL\_TOP> is the actual value (i.e. /u02/visappl).

6. Download the certificate file appltop.cer:

```
get appltop.cer
```

7. Close the FTP session.

8. On your client machine, launch a command prompt session, and execute oajinit.exe from the command line. Wait until JInitiator is successfully installed.

9. Change to the directory in which JInitiator is installed (typically C:\Program Files\Oracle\Oracle JInitiator 1.1.7.27 Export).

10. Change to the bin subdirectory.

11. Copy the appltop.cer file into the bin directory.

12. Execute the following command exactly as shown:

```
./javakey.exe ic appltop appltop.cer
```

13. If your browser was open, close and restart it.
14. Log in to Oracle Forms. The yellow bar should not appear.

## 11.2 Error ORA-29868 While Executing amviccn.sql

When running the database driver and executing amviccn.sql, you may receive the following error message:

```
ORA-29868: cannot issue DDL on a domain index marked as LOADING.
```

This error causes adpatch to halt.

To correct this error, use the following procedure.

---

---

**Note:** For instructions on how to use the AutoPatch utility, see *Maintaining Oracle Applications, Release 11i*.

---

---

### Steps

1. Open another window or telnet session.
2. Set up the environment as usual.
3. Change to the \$AMV\_TOP/patch/1306413/amv/patch/115/sql directory.
4. Edit the file amviccn.sql to change the line

```
dbms_sql.parse(curs, 'DROP INDEX '||dropIndex.index_name, dbms_sql.native);
```

to

```
dbms_sql.parse(curs, 'DROP INDEX '||dropIndex.index_name||' FORCE',  
dbms_sql.native);
```

That is, add the FORCE clause to the DROP INDEX command.

5. Save the file amviccn.sql.
6. Use the adctrl utility to restart the failed worker by performing the following procedure:
  - a. Execute adctrl.
  - b. Accept defaults for all prompts until you get to the Main Menu.
  - c. Choose **1. Show Worker Status** to identify the failed worker.

- d. Press **Return** to continue.
  - e. Choose **2. Tell Worker To Restart A Failed Job** and enter the ID number of the failed worker.
  - f. Choose **1. Show Worker Status** again to monitor the status of the job.
7. If all else fails, stop and restart adpatch.

## 11.3 Display Manager Errors

If a stack trace or error message indicates that a Display Manager API caused an error, check the top line of the exception Java stack trace for the last class and method that was called and caused the error.

Display Manager uses two sets of log files:

- FWSYS, system log files that are used by JTF and are not used by Applications.
- ibe.log.run, the Application log files that contain the sequence of actions recorded by Oracle iStore 11i.

### 11.3.1 Display Manager Error Messages

Use the following fixes and workarounds for the Display Manager error messages listed below.

#### **TemplateNotFoundException or MediaNotFoundException**

This error occurs when either `DisplayManager.getURL()` or `DisplayManager.getTemplate(accessName)`, `DisplayManager.getMedia(accessName)` methods or used.

- Verify that the template or media was created in the Admin Console by searching for it by access name in the Admin Console.
- Reboot the server.
- Check the log file for diagnostic messages.

#### **NullPointerException (Template or Multimedia Object is Null)**

This error occurs when methods `Item.getTemplateFileName()` or `Section.getTemplateFileName()` methods or `Display Manager.getSectionMedia(...)` are made.

This error indicates that a product or section mapping to a template or multimedia was not defined and that a store level default was also not defined.

- Verify that the mapping for the product or section was specified in the admin console.
- If the category default is in use, check whether a mapping has been defined at the category level for the requested display style.
- Verify that the specified display styles/multimedia components are actually available.
- Verify that the server was bounced after the mapping was created.
- Check the log file for diagnostic messages.

**New Template (or Multimedia) Has Been Associated with a Product/Category/Section, But Won't Show Up at Runtime.**

- Verify that the association was created successfully in the admin console.
- Verify that the specified display styles or multimedia components are available.
- Verify that the server was bounced after the association was created.
- Check the log file for diagnostic messages.

## 11.4 Catalog and Pricing Errors

To troubleshoot Catalog and Pricing errors, use the checkpoints or the following fixes for specific error messages.

### 11.4.1 Checkpoints

- View the JSP source in the Web browser and look for a stack trace.
- Check the log file for a stack trace.
- If you know the JSP or Java method in which the error occurred, search the log file for debug statements from the JSP or method.
- If a Web store page does not appear, verify whether the problem is with the database, Apache server, connection, or JTF/AOL as follows:
  1. Go to the following URL:

`http://<host>:<apache port>/html/jtfmain.htm`

This page has links to test the basic functionalities that Oracle iStore 11i depends on.

If the page itself does not appear, the problem is with the database connection.

2. Follow the instructions on the page and use the links to check for environment problems.
- If an Oracle iMarketing posting does not show up in the Oracle iStore 11i catalog, check the following points:
    - Oracle iMarketing is installed.
    - The profile option IBE: Use iMarketing Postings is set to Yes.
    - The centralized posting JSP (ibecpstg.jsp or ibeCCtpPostingI.jsp) is modified to set the correct posting ID value.
    - After modifying the centralized posting JSP, the following pages were removed from the `_pages` directory and recompiled: `ibec*.java`, `ibescdch.java`, `ibeCCt*.java`, `ibeCScdViewA.java`, `ibec*.class`, `ibescdch.class`, `ibeCCt*.class`, and `ibeCScdViewA.class`.
    - If Oracle iStore 11i has successfully displayed other types of Oracle iMarketing postings, the problem may be due to an Oracle iMarketing setup issue or other Oracle iMarketing problem.

## 11.4.2 Specific Error Messages

Use the following fixes and workarounds for the error messages listed below.

### No Items in the Catalog

- Check the following profiles:
  - ASO: Product Organization
  - IBE: Item Validation Organization
  - MO: Operating Unit
- Check the Item setup for items that should appear in the catalog. They must have the following setup:
  - `web_status='PUBLISHED'`
  - `start_date_active` is NULL or `<=SYSDATE`

- `end_date_active` is NULL or `>=SYSDATE`
- The item's primary unit of measure is in `MTL_UNITS_OF_MEASURE_VL`.
- The item is available in the user's organization. A user only sees items in his or her inventory organization. You can check the Oracle iStore 11i log file for "Organization Id" to see the value of the user's Inventory Organization ID.
- If there are configurable items in the catalog, confirm that their components' item setups are also correct. Otherwise, their descriptions will not appear in the Oracle iStore 11i shopping cart.

### Items Do Not Have Prices

- Check the following profiles:
  - IBE: Pricing Event -- Before Shopping Cart
  - IBE: Request Type to get a Price
  - IBE: Use Price list associated with Specialty Store
- Confirm Items are in the minisite price list.
- Confirm that the minisite has the correct price lists for walk-in and registered B2B and B2C users.
- Confirm that the price list has item prices for the correct UOMs.
- Confirm that the price is set up correctly in Oracle Pricing.
- Check the log file for the pricing API that is being called (`Item.getListAndBestPrices`). Confirm the values that Oracle iStore 11i passes to the Pricing engine: price list ID, currency code, inventory item IDs, UOM codes, price request type, and pricing event. Oracle iStore 11i only passes party ID and account ID for user-specific pricing. Check the values that Oracle Pricing returns: status code and status text.
- Status Code from Pricing Engine
  - UPDATED
  - DUPLICATE\_PRICE\_LIST
  - INVALID\_UOM
  - IPL: Invalid price list ID, that is, a non-existent price list.

- If the profile option IBE: Use Price list associated with Specialty Store is set to No, check the following setups:
  - Set the profile option ASO: OM Defaulting to Yes at the desired level: site, application, responsibility, or user.
  - Log in to Oracle Forms as SYSADMIN. Choose the Oracle Pricing Manager responsibility. Select **Setup > Event Phases**. Query for all event phases. Set Search Flag to Yes for the pricing event "LINE" in all phases.
  - Log in to Oracle Forms as SYSADMIN. Choose the Oracle Pricing Manager responsibility. Select **Price Lists > Price List Setup**. Search for the price list in question. Make sure the precedence at the product level is set correctly so that Oracle Pricing can resolve to a single price list.

For example, if there are only two price lists defined, Price List A and Price List B, and all users qualify for both price lists, and if both price lists contain Product X with precedence 220, then Oracle Pricing returns an error because it cannot choose a single price list. If the Product X precedence is 100 in Price List A and 220 in Price List B, Oracle Pricing can resolve to Price List A.

- To confirm that the defaulting row is set up correctly in Oracle Order Management, log in to Oracle Forms as SYSADMIN and choose the Order Management Super User responsibility. Select **Setup > Rules > Defaulting**. Search using the criteria of Application = Oracle Order Management and Entity = Order Header. Select Order Type from the Attributes section and click on **Defaulting Rules**. Add an entry in the Default Sourcing Rules with the following information:
  - Sequence = 1
  - Source Type = Constant Value
  - Order Type = Standard

### **“Add to Cart” Buttons Do Not Appear**

“Add to Cart” buttons are not displayed for items that are not orderable on the Web and for items that do not have defined prices.

- Check that the item setup has `orderable_on_web_flag = 'Y'` in Oracle Inventory.
- Check the price setup.
- Confirm that the item’s BOM item type is not OPTION CLASS.



- If the item's BOM item type is MODEL, the item must have the Oracle Configurator UI set up.

## 11.5 Shopping List Errors

To troubleshoot Shopping List, perform the following checks:

- Run `ctx_output.start_log("log")`
- Check the log file for errors in the program flow.
- Check whether data is stored or modified in the database table.
- Run SQL from the log files.
- Run PL/SQL scripts if a problem has occurred in the PL/SQL layer.

## 11.6 Search Errors

To troubleshoot Search errors, use the following checkpoints:

- Check the Search log file by executing `ctx_output.start_log('log')`.
- Check that Concurrent Manager is up and running.
- If searches are inaccurate, verify that `interMedia` was set up correctly. Refer to [Section 11.8, "Potential Issues Installing Oracle8i interMedia Text Version 8.1.7"](#).
- Look for data stored in the indexed search column.
- If you add multiple items that do not appear in the search table, re-run the Concurrent Manager program `Store Search Insert`. This program populates the search table in Oracle `iStore 11i` with product information from the inventory tables.

---

---

**Note:** The search function goes offline while `Store Search Insert` is running, which takes about forty-five minutes.

---

---

- Verify that the `listener.ora` and `tnsnames` entries are correct so that the callout to the `.dll` can be made. For UNIX, refer to [Section 11.8, "Potential Issues Installing Oracle8i interMedia Text Version 8.1.7"](#).

For Windows NT, refer to the following example.

```
listener.ora
```

```
*****
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP)(HOST = sthattil-pc)(PORT = 1521))
      )
    )
    (DESCRIPTION =
      (PROTOCOL_STACK =
        (PRESENTATION = GIOP)
        (SESSION = RAW)
      )
      (ADDRESS = (PROTOCOL = TCP)(HOST = sthattil-pc)(PORT = 2481))
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = E:\Oracle\Ora81)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = ORCL)
      (ORACLE_HOME = E:\oracle\ora81)
      (SID_NAME = ORCL)
    )
  )
)

*****
tnsnames.ora
*****
EXTPROC_CONNECTION_DATA.US.ORACLE.COM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC0))
    )
    (CONNECT_DATA =
      (SID = PLSExtProc)
      (PRESENTATION = RO)
    )
  )
)
```

```
)  
)
```

If the code is correct, refresh the dr\$libx so it can find the .dll to create the index (my oractxx8.dll is in my \$ORACLE\_HOME\bin directory). The command to recreate the library is:

```
create or replace library dr$libx as 'e:\oracle\ora81\bin\oractxx8.dll';
```

Start the ctxsrv and re-index.

## 11.7 Postsales Errors

To fix issues with Postsales, first verify that the following prerequisites were done.

- All the views in Postsales are “VALID” in the database.
- Regions exist in Apps. Use the developer responsibility AK Developer or Apps for the Web Manager.
- Shipments, invoices and payments were created by the merchant through ERP applications before trying to view them in Oracle iStore 11i.

### 11.7.1 Order Summary Page Records Out of Sequence

The workaround is to use Search to locate an order.

## 11.8 Potential Issues Installing Oracle8i interMedia Text Version 8.1.7

Use the following procedures to troubleshoot problems installing Oracle8i interMedia Text Version 8.1.7.

### 11.8.1 Manually Installing ctxsys Data Dictionary

Data Dictionary Installation interMedia Text is integrated with the Oracle Database Creation Assistant (DBCA) so the ctxsys data dictionary should be installed when using this tool. If the ctxsys data dictionary does not install, use the following procedure to install it manually.

### Prerequisites

- The interMedia Text files are installed.
- The database does not have a ctxsys user.
- The current directory is `$/ctx/admin`.
- You can `sqlplus internal`.

### Steps

1. Create the ctxsys user and pass it the ctxsys password, default tablespace, and temporary tablespace as arguments.

```
sqlplus internal @dr0csys <password> <def_tblspc> <tmp_tblspc>
```

2. Install the data dictionary:

```
sqlplus ctxsys/<password> @dr0inst <ctxx_library>
```

The argument is the full path to the ctxx library, for instance:

```
sqlplus ctxsys/<password> @dr0inst $ORACLE_HOME/ctx/lib/libctxx8.so
```

3. Install appropriate language-specific default preferences. There are more than forty scripts in `$/ctx/admin/defaults` that create language-specific default preferences. They are named in the form `drdefXX.sql`, where `XX` is the language code (from the Server Reference Manual).

For instance, to install the US defaults:

```
sqlplus ctxsys/<password> @defaults/drdefus.sql
```

interMedia Text should now be installed and working.

## 11.8.2 Post-Installation Setup

If this database was an existing ConText site, make sure to remove `text_enable` from the `init.ora`. It is no longer used in Oracle8i, and will actually prevent Oracle8i from operating properly. You will get errors such as “Cannot find package DR\_REWRITE.”

Ensure that the Net8 listener is running and is configured to invoke external procedures. A brief description of the process is below, and complete details are in *Oracle8i Server Administrator's Guide*.

## Steps

1. Add an entry to the tnsnames.ora:

```
extproc_connection_data =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = ipc)
      (KEY = DBSID))
    (CONNECT_DATA = (SID = ep_agt1)))
```

DBSID is the database SID. ep\_agt1 can be named anything.

extproc\_connection\_data should not be changed.

2. Add the following to the listener SID\_LIST:

```
SID_DESC = (SID_NAME = ep_agt1)
  (ORACLE_HOME = /oracle)
  (ENVS = LD_LIBRARY_PATH=/oracle/ctx/lib)
  (PROGRAM = extproc)
```

ep\_agt1 matches the CONNECT\_DATA SID for extproc\_connection\_data in the tnsnames.ora. The PROGRAM section tells the Net8 listener to start the external procedure process. The ENVS section, which is shown here for UNIX, will ensure that the environment includes ?/ctx/lib in LD\_LIBRARY\_PATH. This is needed so that indexing can use the INSO filters.

3. Since the extproc\_connection\_data ADDRESS section specifies ipc, make sure that the ADDRESS\_LIST of listener.ora accepts ipc connections.

A quick way to test the Net8 configuration is to do:

```
exec ctx_output.start_log('log')
```

from SQL\*Plus. If the setup was not performed correctly, you get the error,

```
DRG-50704: Net8 listener is not running or cannot start external procedures.
```

To troubleshoot this error, check the following possible causes:

- listener is not running.
- listener.ora is not configured for extproc.
- tnsnames.ora is not configured for extproc.
- listener does not accept ipc connections.

## 11.9 Reporting Issues

Use the following guidelines to report problems and file bugs.

Check store functionality in the following order:

1. Check whether all server processes (Oracle, TNS listener, WebDB listener, Forms server, Reports server, Apache, etc.) are up and running.

2. Check JTF login by going to:

```
http://<host>:<apache port>/html/jtfllogin.jsp
```

and try to log in as SYSADMIN. If login fails, the problem is with JTF. File a bug on JTF (product code 481).

3. Check the Oracle iStore 11i Merchant UI by logging in to:

```
http://<host>:<apache port>/OA_HTML/jtfllogin.jsp
```

with a store manager user name. (See [Section 7.2, "Setting Up Store Manager User Accounts"](#) for more information on creating the user name.) If login fails, the problem is with the Merchant UI. File a bug on Oracle iStore 11i (product code 384, component SPCLTYSTR).

4. Check the Customer UI by going to:

```
http://<host>:<apache port>/OA_HTML/ibeCZzpHome.jsp?minisite=<minisite ID>
```

and seeing if the store comes up. If not, file a bug on Oracle iStore 11i (product code 384). Use the error message as guidance for which component to specify in the bug.

---

---

**Note:** Make sure you have set up the guest user account before checking the Customer UI. See [Section 7.3, "Setting Up the Guest User Account"](#) for more information.

---

---

5. If the store comes up but there are problems adding items to the shopping cart and/or placing orders, use the Oracle Forms UI to check if Order Capture is working. Log in as SYSADMIN, select Order Capture Sales Manager responsibility, select Order Capture, and enter a quote. (Refer to *Oracle Order Capture Concepts and Procedures, Release 11i* for further details.) If Order Capture is not working, file a bug on Oracle Order Capture (product code 769). If Order Capture is working, file a bug on Oracle iStore 11i (product code 384, component SHPCRT).

---

---

## Oracle iStore 11i Roles and Responsibilities

This appendix lists the Oracle Forms, Oracle CRM Applications, and Oracle iStore 11i Customer UI roles and responsibilities necessary to implement and administer Oracle iStore 11i.

You can create new users and responsibilities, and assign responsibilities as needed. See *Oracle Applications System Administrator's Guide, Release 11i*, *Oracle CRM Foundation Implementation Guide, Release 11i*, and *Oracle CRM Foundation Concepts and Procedures, Release 11i* for more information.

Topics include:

- [Oracle Forms Roles and Responsibilities](#)
- [Oracle CRM Applications Roles and Responsibilities](#)
- [Oracle iStore 11i Customer UI Roles and Responsibilities](#)

## A.1 Oracle Forms Roles and Responsibilities

Access Oracle Forms by navigating to:

`http://<host>:<apache port>/`

and clicking on **Apps Logon Links > VIS Logon** through the Forms cartridge (UNIX). Log in with the appropriate user name to perform the specified tasks.

The following table summarizes the roles and responsibilities necessary to perform setup and administrative tasks for Oracle iStore 11i in Oracle Forms.

**Table A-1 Oracle Forms Roles and Responsibilities**

Role	Responsibility	Tasks
SYSADMIN	AK Developer	Define regions in Apps to troubleshoot Postsales errors. See <a href="#">Section 11.7, "Postsales Errors"</a> for details.
SYSADMIN	Application Developer	<p>Modify messages to change the text in the bins in the Customer UI. See <a href="#">Section 5.4.1, "Creating Template Source Files"</a> for details.</p> <p>Set up descriptive flexfields to appear on item detail pages. See <a href="#">Section 5.9.3, "Adding Item Descriptive Flexfields"</a> for details.</p> <p>Set up comment flexfields to appear on the checkout page. See <a href="#">Section 5.11.3, "Specifying Flexfields At the Checkout Page"</a> for details.</p> <p>Set up System Administrator Access in the IBE_DEFAULT_PAYMENT_TERM_ID profile option, during the process of setting site-level profile options as SYSADMIN with the System Administrator responsibility. See <a href="#">Section 7.15, "Setting Up Site-Level Profile Options"</a> for details.</p>
SYSADMIN	Apps for the Web Manager	Define regions in Apps to troubleshoot Postsales errors. See <a href="#">Section 11.7, "Postsales Errors"</a> for details.



**Table A-1 Oracle Forms Roles and Responsibilities (Cont.)**

<b>Role</b>	<b>Responsibility</b>	<b>Tasks</b>
SYSADMIN	Inventory responsibility for the Master Inventory Organization	<p>Publish items in Inventory to make them available for sale in the specialty stores. See <a href="#">Section 2.3.5, "Setting Up Product Items in Oracle Inventory"</a>, <a href="#">Section 4.5, "Setting Up the Product Catalog"</a> and <a href="#">Section 5.9.1, "Modifying the Product Catalog"</a> for details.</p> <p>Specify in Inventory if customers can order decimal quantities of an item in the specialty stores. See <a href="#">Section 5.11.2, "Allowing Decimal Quantities for Items"</a> for details.</p>
SYSADMIN	iStore Concurrent Programs Responsibility,	<p>Run the concurrent programs iStore Search Insert and iStore Section Search Refresh when setting up product searches. See <a href="#">Section 5.10, "Setting Up Product Searches"</a> for details.</p> <p>Run the concurrent program iStore - One Click Consolidation to submit Express Checkout orders. See <a href="#">Section 7.13, "Setting Up Concurrent Program Manager"</a> for details.</p>
SYSADMIN	Oracle Pricing Manager	Set up Oracle Pricing. See <a href="#">Section 2.3.12, "Setting Up Oracle Pricing"</a> for details.
SYSADMIN	Order Capture Sales Manager	<p>Set up Order Capture. See <a href="#">Section 7.16, "Setting Up Order Capture in Forms"</a> for details.</p> <p>Check to make sure Order Capture is working when troubleshooting problems with adding items to shopping carts or placing orders. See <a href="#">Section 11.9, "Reporting Issues"</a> for details.</p>
SYSADMIN	Order Management Super User	<p>Set up Order Management. See <a href="#">Section 7.14, "Setting Up Order Management in Forms"</a> for details.</p> <p>Set up Web-enabled shipping methods. See <a href="#">Section 7.14.1, "Setting Up Web-Enabled Shipping Methods"</a> for details.</p>
SYSADMIN	Receivables Manager	Set up Accounts Receivable for credit card payments in Oracle iStore 11i. See <a href="#">Section 5.11.4, "Setting Up Credit Card Payments in Oracle iStore 11i"</a> for details.

**Table A-1 Oracle Forms Roles and Responsibilities (Cont.)**

Role	Responsibility	Tasks
SYSADMIN	System Administrator	<p>Set profile option BOM: Configurator URL of UI Manager for the IBE_CUSTOMER responsibility. See <a href="#">Section 2.4.3, "Setting Up Oracle Configurator"</a> for details.</p> <p>Enter values for the descriptive flexfields that appear on item detail pages when set up by SYSADMIN using the Application Developer responsibility. See <a href="#">Section 5.9.3, "Adding Item Descriptive Flexfields"</a> for details.</p> <p>Set up credit card payments in Oracle iStore 11i. See <a href="#">Section 5.11.4, "Setting Up Credit Card Payments in Oracle iStore 11i"</a> for details.</p> <p>Set up Oracle iStore 11i store manager user accounts. See <a href="#">Section 7.2, "Setting Up Store Manager User Accounts"</a> for details.</p> <p>Set up the Oracle iStore 11i guest user account. See <a href="#">Section 7.3, "Setting Up the Guest User Account"</a> for details.</p> <p>Set Foundation (JTF) profile options. See <a href="#">Section 7.5, "Setting Foundation (JTF) Profile Options"</a> for details.</p> <p>Set Oracle iStore (IBE) profile options. See <a href="#">Section 7.6, "Setting Oracle iStore (IBE) Profile Options"</a> for details.</p> <p>Set Order Capture (ASO) profile options. See <a href="#">Section 7.8, "Setting Order Capture (ASO) Profile Options"</a> for details.</p> <p>Set Multi Organization (MO) profile options. See <a href="#">Section 7.10, "Setting Multi Organization (MO) Profile Options"</a> for details.</p> <p>Assign the iStore Concurrent Programs Responsibility to a user. See <a href="#">Section 7.13, "Setting Up Concurrent Program Manager"</a> for details.</p> <p>Check the status of concurrent program requests. See <a href="#">Section 7.13, "Setting Up Concurrent Program Manager"</a> for details.</p> <p>Set up site-level profile options. See <a href="#">Section 7.15, "Setting Up Site-Level Profile Options"</a> for details.</p>

## A.2 Oracle CRM Applications Roles and Responsibilities

Access the Oracle CRM Applications login page at:

`http://<host>:<apache port>/OA_HTML/jtfflogin.jsp`

Log in with the appropriate user name to perform the specified tasks.

The following table summarizes the roles and responsibilities necessary to perform setup and administrative tasks for Oracle iStore 11i in Oracle CRM Applications.

**Table A-2 Oracle CRM Applications Roles and Responsibilities**

Role	Responsibility	Tasks
<Store Manager User Account>	IBE_ADMINISTRATOR Logging in with this responsibility launches the Oracle iStore 11i Merchant UI.	<p>Set up and modify specialty stores. See <a href="#">Section 4.3, "Creating Specialty Stores"</a> for details.</p> <p>Set up and modify the hierarchy. See <a href="#">Section 4.4, "Creating the Hierarchy"</a> and <a href="#">Section 5.6, "Modifying the Hierarchy"</a> for details.</p> <p>Set up and modify the product catalog. See <a href="#">Section 4.5, "Setting Up the Product Catalog"</a> and <a href="#">Section 5.9.1, "Modifying the Product Catalog"</a> for details.</p> <p>Set up store appearance by managing multimedia, multimedia components, templates, and display styles at item, category, section, and store level. See <a href="#">Chapter 5</a> for more information.</p> <p>Set up product relationships. See <a href="#">Section 4.5.1, "Using Seeded Relationship Types"</a> and <a href="#">Section 5.7, "Creating Relationships"</a> for details.</p> <p>Manage the product and section caches. See <a href="#">Section 6.5, "Managing the Cache"</a> for details.</p>

**Table A-2 Oracle CRM Applications Roles and Responsibilities (Cont.)**

Role	Responsibility	Tasks
SYSADMIN	Logging in with this role launches the Oracle CRM System Administrator Console.	<p>Set up default roles and responsibilities for Oracle iStore 11i customers. See <a href="#">Section 6.3, "Setting Up Oracle iStore 11i Customer Types"</a> for details.</p> <p>Create B2B user roles, approve customer users for the specialty stores, and modify customer user data. See <a href="#">Section 6.4, "Setting Up B2B Users"</a> for details.</p> <p>Set up JTF properties. See <a href="#">Section 7.4, "Setting Up JTF Properties"</a> for details.</p>

### A.3 Oracle iStore 11i Customer UI Roles and Responsibilities

Customers are also assigned roles and responsibilities, which customize their Web store experience. At least one customer responsibility is assigned to each customer name when the name is approved during user registration. You can use the seeded IBE\_CUSTOMER responsibility and create other customer responsibilities too.

In a multiple operating unit environment, you need to create a customer responsibility for each operating unit if you want a customer to access only items from the Inventory Organization specific to each operating unit.

In the Merchant UI, you can specify a list of the customer responsibilities that are supported by a given specialty store. You can select these from all existing responsibilities. You can also specify for a given specialty store whether Oracle iStore 11i checks the customer's responsibilities and grants access only if the customer has an assigned responsibility that is supported by the specialty store.

If multiple responsibilities are supported by a specialty store, a customer who logs in to the store must choose one responsibility for that session. The responsibility uniquely identifies the operating unit against which any orders placed during the session will be booked. The responsibility is assigned only for the current session.

If you set up a specialty store to check the customer's responsibility, the customer can choose only from the responsibilities that have been assigned to him or her during registration. If the specialty store is not set up to check the customer's responsibility, then the customer can choose any supported responsibility in any specialty store that does not check the customer's assigned responsibilities.

The following table summarizes the responsibilities seeded by Oracle iStore 11i for Web store customers to use in the Customer UI, located at:

---

`http://<host>:<apache port>/OA_HTML/ibeCZzpHome.jsp?minisite=<minisite ID>`

**Table A-3 Oracle iStore 11i Customer UI Roles and Responsibilities**

<b>Role</b>	<b>Responsibility</b>	<b>Tasks</b>
<Guest User Account>	IBE_CUSTOMER	The guest user name is assigned to every customer who browses the store without registering. You must assign the IBE_CUSTOMER responsibility or another customer responsibility as a default for guest users. See <a href="#">Section 7.3, "Setting Up the Guest User Account"</a> for details.
<Customer User Account>	IBE_CUSTOMER	The IBE_CUSTOMER responsibility is assigned by default to every registered customer.



---

---

# Index

## A

---

APIs, 5-14

    Catalog, 8-2

    oracle.apps.ibe.catalog package, 8-2

    oracle.apps.ibe.postsales package, 10-2

    oracle.apps.ibe.shoppingcart.quote package, 9-3

    Postsales, 10-2

    Shopping Cart Quote, 9-3

Available to Promise (ATP), 2-27

## B

---

B2B user permissions, 6-12

    IBE\_USER\_ADMIN, 6-9

B2B user roles, 6-9

    creating, 6-12

    seeded values, 6-9

B2B users, 6-9

    approving, 6-15

    default customer responsibility, 6-8

    default customer role, 6-8

    modifying role assignments, 6-20

    modifying user data, 6-17

    organization accounts, 6-15

    Role Management, 6-11

    user administration, 6-9

    User Management, 6-10

B2C users

    default customer responsibility, 6-8

## C

---

cache management, 6-22

    product cache, 6-23, 6-24

    purging entire cache, 6-23

    section cache, 6-23

Cascading Style Sheets, 5-12

categories, 5-27

    assigning display styles, 5-28

    assigning multimedia, 5-28

    assigning templates, 5-27

concurrent programs

    checking program status, 7-27

    iStore - One Click Consolidation, 7-26

    iStore Search Insert, 7-25

    iStore Section Search Refresh, 7-26

contracts, 2-29

cookies, 7-31

credit card payments, 5-48

customers

    roles and responsibilities, A-6

## D

---

dependency

    verification, 3-1

dependency requirements, 2-5

display styles

    calling templates, 5-19

    cataloging, 5-19

    seeded values, 5-19

## E

---

error messages

    MediaNotFoundException, 11-4

    NullPointerException, 11-4

- ORA-29868, 11-3
- TemplateNotFoundException, 11-4
- errors
  - catalog, 11-5
  - Display Manager, 11-4
  - Oracle8i interMedia ctxsys data
    - dictionary, 11-11
  - Oracle8i interMedia post-installation, 11-12
  - Postsales, 11-11
  - pricing, 11-5
  - product search, 11-9
  - shopping list, 11-9
- Express Checkout
  - processing orders, 7-26, 7-27

---

## H

- hardware requirements, 1-4
- hierarchy
  - creating, 4-8
  - modifying, 5-20
  - moving sections, 5-23

---

## I

- installation
  - verification, 3-1

---

## J

- Java applet warning workaround, 11-2
- JavaServer Pages, 5-10
- JSPs, 5-10
  - caching, 5-11
  - Cascading Style Sheets, 5-12
  - naming conventions, 5-11
  - product searches, 5-42
  - source code directory, 5-11
- JTF properties, 7-7

---

## L

- languages, 2-9

---

## M

- Merchant UI, 4-3
- multimedia
  - cataloging, 5-3
  - creating media source files, 5-3
  - customizing, 5-3
  - naming, 5-4
- multimedia components
  - cataloging, 5-7
  - seeded values, 5-7

---

## O

- operating units
  - responsibilities, A-6
  - setting up multiple, 2-13, 2-19, 7-24, A-6
- Oracle 8i interMedia, 11-11
- Oracle Advanced Inbound, 2-26
- Oracle Advanced Supply Chain Planning, 2-27
- Oracle Application Object Library (AOL), 2-5
  - languages, 2-9
  - service items, 2-10
- Oracle Configurator, 2-27
- Oracle Contracts, 2-29
- Oracle Forms
  - roles and responsibilities, A-2
  - setting up Oracle iStore user accounts, 7-2
  - setting up Oracle Order Capture, 7-30
  - setting up Oracle Order Management, 7-28
- Oracle General Ledger, 2-5, 2-12
- Oracle Human Resources, 2-5
- Oracle Inventory, 2-5, 5-22
  - service items, 2-15
  - serviceable items, 2-14
  - setting up, 2-12
  - setting up for product search, 5-36
  - setting up for the product catalog, 4-14, 5-29
- Oracle iPayment, 2-32
  - setting up for credit card payments, 5-49
- Oracle Order Capture
  - setting up, 2-17
  - setting up in Oracle Forms, 7-30
- Oracle Order Management, 2-6, 5-46
  - setting up, 2-18



- setting up checkout page flexfields, 5-47
  - setting up in Oracle Forms, 7-28
- Oracle Pricing, 5-46
  - service items, 2-21
  - serviceable items, 2-21
  - setting up, 2-19
- Oracle Receivables
  - setting up, 2-17
  - setting up for credit card payments, 5-49
- Oracle Shipping, 2-33
- Oracle Workflow, 2-33

## P

---

- password
  - resetting customer password, 6-21
- product catalog
  - modifying, 5-28
  - Oracle Inventory prerequisites, 4-14, 5-29
  - setting up, 4-14
- product searches
  - changing type, 5-39
  - Oracle Inventory prerequisites, 5-36
  - profile options, 5-36
  - program units, 5-42
  - search index tables, 5-41
  - search table, 5-35
  - search tables, 5-43
  - searchable attributes, 5-35, 5-43
  - setting up, 5-35
  - setting up category-level, 5-37
  - setting up section-level, 5-38
  - stopwords, 5-44
- products
  - adding item descriptive flexfields, 5-34
  - assigning multimedia, 5-32
  - assigning templates, 5-32
  - category-level presentation, 5-27
  - creating images for, 5-33
  - publishing, 4-16, 5-30
- profile options
  - AOL, 7-20
  - Foundation (JTF), 7-8
  - multi organization (MO), 7-23, 7-24
  - Oracle iStore (IBE), 7-9

- Oracle Order Capture (ASO), 7-21
- site level, 7-29

## R

---

- relationship rules, 5-22, 5-24
  - creating mapping rules, 5-25
  - creating SQL rules, 5-26
- relationship types, 5-24
  - adding relationship rules, 5-25
  - creating, 5-25
  - seeded values, 4-18
- relationships
  - creating, 5-24
- reporting problems, 11-14
- requirements
  - hardware, 1-4
  - Oracle applications, 2-5
  - software, 1-5
- responsibilities, A-1
  - creating, 7-3, 7-6
  - customers, A-6
  - Oracle Forms, A-2
- roles, A-1
  - customers, A-6
  - Oracle Forms, A-2

## S

---

- search index tables, 5-41
- seeded values
  - B2B user roles, 6-9
  - display styles, 5-19
  - multimedia components, 5-7
  - relationship types, 4-18
- service items, 2-10, 2-15, 2-21
- serviceable items, 2-14, 2-21
- shipping methods
  - enabling for Web, 7-29
- shopping cart
  - checkout page flexfields, 5-47
  - credit card payments, 5-48
  - decimal quantities, 5-46
  - UOM conversions, 5-46
  - software requirements, 1-5

- specialty store
  - administration, 6-2
  - creating, 4-2, 4-4
  - customization, 5-2
  - globalization, 4-4
  - root section, 4-5, 4-8, 5-22, 5-24
  - testing the storefront, 4-20

## T

---

- templates
  - assigning presentation levels, 5-15
  - Cascading Style Sheets, 5-12
  - cataloging, 5-14, 5-15
  - creating template source files, 5-10
  - customizing, 5-10
  - JSP caching, 5-11
  - JSP naming conventions, 5-11
  - JSP source code directory, 5-11
  - modifying seeded source files, 5-12
  - naming, 5-14
- testing the store, 4-20

## U

---

- users
  - creating, 7-3, 7-6
  - guest account, 7-3
  - store manager account, 7-2